Computer Methods in Applied Mechanics and Engineering 432 (2024) 117397

Contents lists available at ScienceDirect

Comput. Methods Appl. Mech. Engrg.

journal homepage: www.elsevier.com/locate/cma

KAN-ODEs: Kolmogorov–Arnold network ordinary differential equations for learning dynamical systems and hidden physics

Benjamin C. Koenig[1], Suyong Kim[1], Sili Deng [*]

Department of Mechanical Engineering, Massachusetts Institute of Technology, 77 Massachusetts Ave., Cambridge, 02139, MA, USA

ARTICLE INFO

Dataset link: https://github.com/DENG-MIT/KAN-ODEs

Keywords:
Kolmogorov–Arnold networks
Partial differential equations
Dynamical systems
Machine learning
Model discovery
Interpretable networks

ABSTRACT

Kolmogorov–Arnold networks (KANs) as an alternative to multi-layer perceptrons (MLPs) are a recent development demonstrating strong potential for data-driven modeling. This work applies KANs as the backbone of a neural ordinary differential equation (ODE) framework, generalizing their use to the time-dependent and temporal grid-sensitive cases often seen in dynamical systems and scientific machine learning applications. The proposed KAN-ODEs retain the flexible dynamical system modeling framework of Neural ODEs while leveraging the many benefits of KANs compared to MLPs, including higher accuracy and faster neural scaling, stronger interpretability and generalizability, and lower parameter counts. First, we quantitatively demonstrated these improvements in a comprehensive study of the classical Lotka–Volterra predator–prey model. We then showcased the KAN-ODE framework's ability to learn symbolic source terms and complete solution profiles in higher-complexity and data-lean scenarios including wave propagation and shock formation, the complex Schrödinger equation, and the Allen–Cahn phase separation equation. The successful training of KAN-ODEs, and their improved performance compared to traditional Neural ODEs, implies significant potential in leveraging this novel network architecture in myriad scientific machine learning applications for discovering hidden physics and predicting dynamic evolution.

1. Introduction

Dynamical system modeling is a key part of many branches of engineering and science. Traditionally, governing equations (often partial differential equations (PDEs)) have been derived through extensive observations and careful selection of model equations. These processes, heavily relying on expert knowledge, have gradually been augmented and sometimes replaced with machine learning techniques by leveraging the data-fitting power in neural networks [1–4]. Despite the potential to reduce the need for detailed expert knowledge, these data-driven modeling approaches are often subject to tradeoffs between the amount of prior knowledge embedded in the model versus the interpretability of the model: purely data-driven approaches tend to lack interpretability, and conversely, interpretable methods typically require substantial knowledge or enforcement of preexisting governing laws [1]. To mitigate this issue, there have been extensive efforts to develop novel modeling schemes that blend data-driven modeling with traditional dynamical system tools, including sparse identification of nonlinear dynamics (SINDy) [5], physics-informed neural network (PINNs) [6], and neural ordinary differential equations (Neural ODEs) [7]. Such studies that bridge this gap have the potential to not only infer high-accuracy models from data but also provide useful insights into the dynamics of what are often unknown physical processes.

[*] Corresponding author.
E-mail addresses: bckoenig@mit.edu (B.C. Koenig), suyong@mit.edu (S. Kim), silideng@mit.edu (S. Deng).
[1] B.C.K. and S.K. contributed equally to this work.

https://doi.org/10.1016/j.cma.2024.117397
Received 18 July 2024; Received in revised form 9 September 2024; Accepted 13 September 2024
Available online 19 September 2024
0045-7825/© 2024 Elsevier B.V. All rights are reserved, including those for text and data mining, AI training, and similar technologies.

Neural ODEs have been developed to learn a continuous evolution of states [7–9], as opposed to the discrete sequences learned by residual networks [10,11]. Chen et al. coupled black box neural networks used as gradient getters with standard ODE solvers in order to learn the relationship between the current state of a system and its gradient [7]. This mapping gives Neural ODEs inherent grid and timescale flexibility, allowing for substantial interpolation and generalization of learned models compared with typical fixed grid neural network approaches. Neural ODEs do not require prior knowledge of a system in order to infer dynamical models from large datasets thanks to their use of black box multi-layer perceptrons (MLPs). However, on the flip side of this benefit, they typically deliver models with thousands to millions of parameters, making any deeper interpretation or extraction of physical insights difficult.

In contrast to Neural ODEs, physics-informed neural networks (PINNs) leverage the governing equations of the dynamical system, wherever available, in the loss function to train a traditional (i.e. not ODE based) neural network that maps the time and spatial coordinates directly to the state variables [6,12,13]. This physics-embedded loss function allows for training that aims to fit both the data and the known governing equations without having to directly solve what are typically expensive PDEs. These networks are significantly interpretable thanks to the governing equations used as a soft constraint in the training cycle, giving physical meaning to trends and patterns that emerge from the training process. The inclusion of the governing equations as a soft constraint also allows PINNs to train with relatively small datasets, which is a useful benefit in many scientific and engineering applications with high costs of data acquisition [6]. Other frameworks have also been developed using a physics-based structure similar to PINN. The auto-regressive dense encoder–decoder model [14] encodes physics into a time-integration network scheme, while the physics-informed convolutional-recurrent network [15] runs PDE solutions through convolutional filters with physical penalty terms and hard-encoded initial and boundary conditions to efficiently develop models. Other works have also investigated the use of hard constraints with PINN-type models for initial and boundary conditions with strong results [16,17]. PINN-type loss functions have additionally been included in Neural ODE frameworks [18]. Further combinations exist in the area of physics-based Neural ODEs, such as Chemical Reaction Neural Networks [19], which hard constrain Neural ODE frameworks to directly learn governing equation parameters from real [20,21] and noisy [22,23] experimental data. A downside shared throughout these numerous physics-based techniques is their requirement of strong prior knowledge of the system's governing laws. In cases where such knowledge is not available, other methods are required.

The SINDy approach [4,5,24,25] tackles dynamical system modeling via sparse regression. Given training data and a set of candidate functions (polynomials, trigonometric functions, constant functions, etc.), it carries out sparse regression to parsimoniously select only the candidate functions needed to describe trends present in the data. The use of a presumed set of candidate functions gives SINDy models significant interpretability over pure black box methods like MLPs or Neural ODEs. It also requires less of the detailed physical knowledge behind the system dynamics as in PINNs and other physics-based methods, and less data with a reduced model search space compared to Neural ODEs.

While most advances in data-driven modeling lie in MLPs and polynomial basis approaches, Liu et al. recently proposed Kolmogorov–Arnold networks (KANs) as an alternative to MLP-based networks [26]. Instead of learning weights and biases given fixed activation functions as in MLPs, the activation functions themselves are learnable in KANs with gridded basis functions and associated trainable scaling factors. Major advantages of KANs over MLPs are increased interpretability thanks to the direct learning of activation functions, faster convergence to low training loss with significantly fewer parameters thanks to their faster neural scaling laws, and the possibility of postprocessing via sparse regression to deliver human-readable expressions. With this potential, recent efforts have been dedicated to extending KANs for time-series predictions based on residual networks by directly mapping training data onto future testing windows [27,28]. However, such a discrete time-step approach has a limited ability to predict the states where various time scales and data collection windows are involved. This research gap naturally motivates our work to embed KANs into Neural ODE-type dynamical system solvers, allowing us to resolve these drawbacks while preserving the interpretability and inference potential of KANs.

In this study, we introduce the possibility of learning interpretable and modular ODE and PDE models for dynamical systems without prior knowledge or functional form assumptions by leveraging KANs as gradient-getters within the concept of Neural ODEs. In doing so, we propose a combined framework that benefits from the high-accuracy, parameter-lean, and strongly interpretable KAN structure coupled with the general-purpose, grid-independent, and solver-flexible Neural ODE framework. A qualitative depiction of the comparison between the currently proposed KAN-ODE method and the other discussed data-driven approaches for dynamical system modeling is shown in Fig. 1. While the current state of the art is generally situated on the diagonal of this plot, where increased prior knowledge fed into a machine learning method delivers better interpretability, we place KAN-ODEs slightly below the diagonal, thanks to their potential to deliver interpretable activations and even symbolic relationships without requiring prior knowledge or presumed functional forms in the training cycle. The ODE-based formulation is also inherently flexible, allowing for variable timesteps depending on the gradient landscape as well as the application of the KAN-ODEs as components of a larger-scale model, for example as the source term in a nonlinear transport PDE. We demonstrated the promise of KAN-ODEs through a comprehensive comparison against standard Neural ODEs, as well as an array of tests at different scales and levels of complexity including a predator–prey ODE, wave propagation and shock formation PDEs, the complex Schrödinger equation, and a phase separation PDE. We found that KAN-ODEs succeeded in all of these cases and were able to reconstruct complete temporal profiles even when given sparse training data. KAN-ODEs with a small number of parameters were able to converge with remarkable accuracy, and sparsity or symbolic relationships imposed during the training cycle or during postprocessing were able to give KAN-ODEs substantial generalization capability. Overall, we found that this encoding of KANs into the Neural ODE framework is a promising step toward small-data and low-knowledge yet interpretable machine learning for dynamical system modeling.

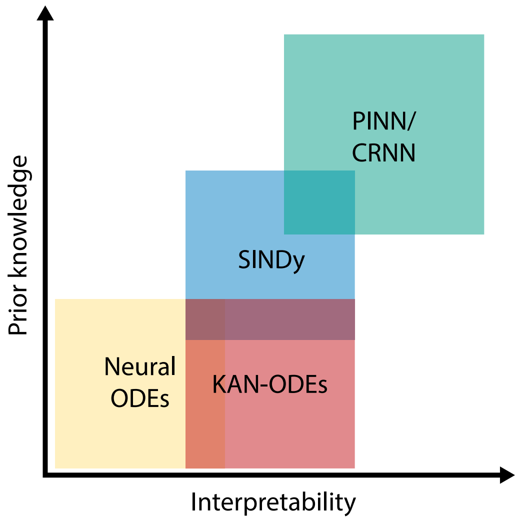

Fig. 1. Qualitative depiction of KAN-ODE's general capability compared to similar machine learning techniques.

2. Methodology

The formulation proposed here leverages KANs and the neural network-ODE solver concept of Neural ODEs. Section 2.1 introduces the concept of a KAN. Then, we describe how we couple KANs to differentiable ODE solvers to model dynamical systems in Section 2.2.

2.1. Kolmogorov-Arnold networks as gradient evaluators

In contrast to MLPs based on the universal approximation theorem [29], KANs [26] are based on the Kolmogorov–Arnold representation theorem (KAT) [30], which states that any multivariate function $f(\mathbf{x}) : [0,1]^n \to \mathbb{R}$ that is continuous and smooth can be represented by a finite sum of continuous univariate functions,

$$f(\mathbf{x}) = f(x_1, \dots, x_n) = \sum_{q=1}^{2n+1} \Phi_q \left(\sum_{p=1}^{n} \phi_{q,p}(x_p) \right). \tag{1}$$

Here, $\phi_{q,p} : [0,1] \to \mathbb{R}$ are univariate functions, while $\Phi_q : \mathbb{R} \to \mathbb{R}$ composes these univariate functions to reconstruct $f(\mathbf{x})$. The implementation in this work defines Φ_q as trainable weights to sum the univariate basis functions. Once these basis functions are chosen, Eq. (1) exactly describes a 2-layer KAN with a hidden layer width of $2n+1$ (the upper bound of the outer summation in Eq. (1)) and an output dimension of 1. The KAT is extended to a general KAN structure with L layers of arbitrary width by exploiting the hierarchical structure of Eq. (1) [26].

$$\mathbf{y} = \mathrm{KAN}(\mathbf{x}) = \left(\Phi_{L-1} \circ \Phi_{L-2} \circ \cdots \circ \Phi_1 \circ \Phi_0\right)(\mathbf{x}). \tag{2}$$

$$\Phi_l = \begin{pmatrix} \phi_{l,1,1}(\cdot) & \phi_{l,1,2}(\cdot) & \cdots & \phi_{l,1,n_l}(\cdot) \\ \phi_{l,2,1}(\cdot) & \phi_{l,2,2}(\cdot) & \cdots & \phi_{l,2,n_l}(\cdot) \\ \vdots & \vdots & & \vdots \\ \phi_{l,n_{l+1},1}(\cdot) & \phi_{l,n_{l+1},2}(\cdot) & \cdots & \phi_{l,n_{l+1},n_l}(\cdot) \end{pmatrix}, \tag{3}$$

where ϕ is the activation function, l is the index of a layer, and n_l and n_{l+1} denote the numbers of nodes in the lth and $(l+1)$th layers, respectively. While Liu et al. [26] originally proposed a B-spline for the gridded basis functions that comprise the various activations ϕ, this study adopts the more computationally efficient Gaussian radial basis functions (RBFs) ψ as proposed by Li [31], such that

$$\phi_{l,\alpha,\beta}(\mathrm{x}) = \sum_{i=1}^{N} w^{\psi}_{l,\alpha,\beta,i} \cdot \psi\left(||\mathrm{x} - c_i||\right) + w^{b}_{l,\alpha,\beta} \cdot b(\mathrm{x}), \tag{4}$$

$$\psi(r) = \exp(-\frac{r^2}{2h^2}), \tag{5}$$

where N is the number of gridpoints, $w^{\psi}_{l,\alpha,\beta,i}$ and $w^{b}_{l,\alpha,\beta}$ are the trainable weights for the RBF function $\psi(r)$ and the base activation function $b(\mathrm{x})$, respectively (superscripts indicating which function the weights apply to), the subscripts $[\alpha, \beta]$ indicate the index of the matrix in Eq. (3) within a KAN layer l that these weights apply to, c_i is the center of the ith RBF function, r is the distance of x from the center, and h is a spreading parameter defined as the gridpoint spacing. The inputs to each layer are normalized to be on the $[-1, 1]$ gridded RBF domain as in [32], which avoids the costly alternative technique of [26] that requires periodically re-gridding the RBF networks in each layer to match the input domain. In addition to fully gridded RBF basis functions, the inputs and outputs of each layer are also directly connected with Swish residual activation functions such that $b(\mathrm{x}) = \mathrm{x} \cdot \mathrm{sigmoid}(\mathrm{x})$ [33].

Eq. (4) therefore defines two input pathways for each layer: the gridded RBF pathway (first term), and the direct (non-gridded) residual activation pathway (second term). The RBF function connects each input node with each output node using a number of basis functions equal to the size of the grid. The number of parameters required for the RBF pathway of a single layer is therefore equal to the product of the input dimension, output dimension, and grid size (following as well from $w^{\psi}_{l,\alpha,\beta,i}$, where for a single layer w^{ψ}_{l} the remaining three indices cover the input, output, and grid indices). Similarly, the residual activation pathway, which directly connects inputs to outputs without a grid, simply requires parameters equal to the product of the input and output dimensions (likewise see $w^{b}_{l,\alpha,\beta}$, which includes the input and output indices α and β but not the grid index i). For example, in a 2-layer KAN with a 2D input, 1D output, 10D hidden layer, and 5-point grid, the numbers of parameters are $2 \times 5 \times 10 + 2 \times 10$ for the input $\rightarrow$ hidden layer, and $10 \times 5 \times 1 + 10 \times 1$ for the hidden $\rightarrow$ output layer, respectively, resulting in 180 parameters in total.

Expanded mathematical formulations for KANs with multivariate outputs become increasingly bulky and cumbersome, and are available in the extensive [26]. For convenience, we use the notation $[n_l, n_{l+1}, N]$ to represent an lth KAN layer for the remainder of this work, where n_l is the input dimension, n_{l+1} is the output dimension, and N is the grid size. For example, a 2-input and 1-output KAN with a 10-node hidden layer and a 5-point grid is represented as "[2, 10, 5], [10, 1, 5]" for the connections between the input and hidden layer, and the hidden and output layer, respectively.

2.2. KAN-ordinary differential equations

A dynamical system is generally expressed by an ODE system such that

$$\frac{d\mathbf{u}}{dt} = \mathbf{g}(\mathbf{u}, t), \tag{6}$$

where $\mathbf{u} \in \mathbb{R}^n$ is a vector of the state variables and $\mathbf{g}$ is the system equation. Chen et al. [7] previously proposed neural ordinary differential equations with the help of the universal approximation theorem (UAT) and the *adjoint* sensitivity method to construct a model NN such that NN $\approx \mathbf{g}$. We similarly adopt the recently developed KAN as an approximator for a dynamical system by leveraging the benefits of the Kolmogorov–Arnold representation theorem (KAT). The proposed KAN-ODEs can be expressed as

$$\frac{d\mathbf{u}}{dt} = \mathrm{KAN}\left(\mathbf{u}(t), \theta\right), \tag{7}$$

where θ is the collection of all trainable weights in the KAN. Here, the input and output dimensions of the KAN are fixed as the dimension of the state variables $\mathbf{u}$, and it is used as a gradient getter. Eq. (7) can be solved using a traditional ODE solver from an initial time t_0 to a desired time t such that

$$\mathbf{u}(t) = \mathbf{u}_0 + \int_{t_0}^{t} \mathrm{KAN}\left(\mathbf{u}(\tau), \theta\right) d\tau, \tag{8}$$

where $\mathbf{u}_0$ is the initial condition at t_0.

To train the KAN model, we define the loss function as the mean squared error

$$\mathcal{L}(\theta) = \mathrm{MSE}\left(\mathbf{u}^{\mathrm{KAN}}(t, \theta), \mathbf{u}^{\mathrm{obs}}(t)\right) = \frac{1}{N}\sum_{i=1}^{N} ||\mathbf{u}^{\mathrm{KAN}}\left(t_i, \theta\right) - \mathbf{u}^{\mathrm{obs}}\left(t_i\right)||^2, \tag{9}$$

where $\mathbf{u}^{\mathrm{KAN}}$ is the prediction, $\mathbf{u}^{\mathrm{obs}}$ is the observation, and N is the total number of time steps. To optimize $\mathcal{L}$, we require gradients with respect to θ. Generally, two options are available: the *forward* sensitivity method [8,34] and the *adjoint* sensitivity method [7,35]. In this paper, the *adjoint* sensitivity method is adopted to efficiently handle a potentially large Kolmogorov–Arnold network [7,8,35] thanks to its more advantageous scaling with model size. By introducing an adjoint state variable $\boldsymbol{\omega}$, the augmented system equation can be derived, which allows for gradient backpropagation through the ODE integrators.

$$\frac{d}{dt}\begin{bmatrix} \mathbf{z} \\ \boldsymbol{\omega} \\ \frac{\partial \mathcal{L}}{\partial \theta} \end{bmatrix} = -\begin{bmatrix} 1 & \boldsymbol{\omega}^T & \boldsymbol{\omega}^T \end{bmatrix}\begin{bmatrix} \mathbf{g} & \frac{\partial \mathbf{g}}{\partial \mathbf{z}} & \frac{\partial \mathbf{g}}{\partial \theta} \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}, \tag{10}$$

where $\mathbf{z}(t) = \mathbf{u}(-t)$, $\mathbf{g} = $ KAN, and T denotes the transpose. Detailed derivations of the adjoint equations (Eq. (10)) can be found in [7,36–38]. After Eq. (10) is computed, θ is updated using a gradient descent method. A schematic depicting the overall training cycle is provided in Fig. 2. This study implements KAN-ODEs in the Julia scientific machine learning ecosystem including packages such as DifferentialEquations.jl [39], Lux.jl [40], KomolgorovArnold.jl [32], and Zygote.jl. Unless otherwise specified, we employed the ODE integrator of `Tsit5` (Tsitouras 5/4 Runge–Kutta method [41]) and the optimizer of `ADAM` [42]. The KANs used in this work were constructed with RBF basis functions and Swish residual activation functions.

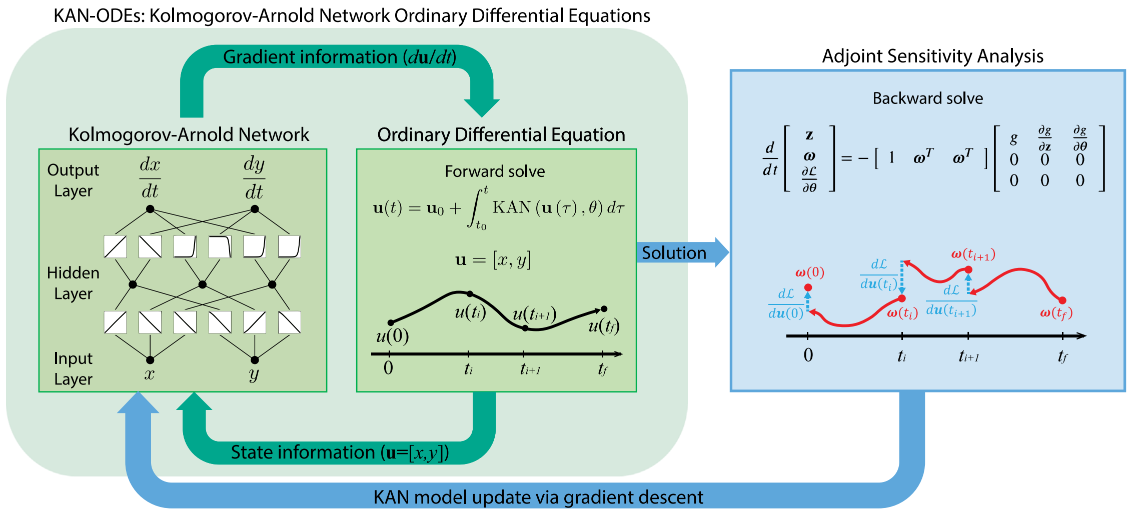
KAN-ODEs: Kolmogorov-Arnold Network Ordinary Differential Equations
Gradient information ($d\mathbf{u}/dt$)
Kolmogorov-Arnold Network
Output Layer
Hidden Layer
Input Layer
$\frac{dx}{dt}$ $\frac{dy}{dt}$ x y
Ordinary Differential Equation
Forward solve
$\mathbf{u}(t) = \mathbf{u}_0 + \int_{t_0}^{t} \mathrm{KAN}(\mathbf{u}(\tau), \theta)\, d\tau$
$\mathbf{u} = [x, y]$
$u(0)$ $u(t_i)$ $u(t_{i+1})$ $u(t_f)$
Solution
Adjoint Sensitivity Analysis
Backward solve
State information ($\mathbf{u}$=[x,y])
KAN model update via gradient descent

Fig. 2. Schematic depicting the overall training cycle of a KAN-ODE system. The loop in green leverages a KAN as a temporal gradient getter for the state vector to solve the ODE forward. Once a solution is generated, the blue loop computes the gradient of the loss function via the adjoint method to update the KAN activation functions. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

3. Experiments

We present five inference examples in this work to demonstrate the broad capability of KAN-ODEs, grouped into three subsections. We first validated our methodology and compared its performance against the standard MLP-based Neural ODEs in Section 3.1, where the dynamics of the Lotka–Volterra predator–prey model were inferred. There we also comprehensively explored the neural scaling benefits of KAN-ODEs, as well as the efficient representations that can be inferred via sparsification and pruning. Then, we tackled a one-dimensional PDE in Section 3.2, using KAN-ODEs to infer a symbolic source term in a wave propagation simulation. This example demonstrates the flexibility of KAN-ODEs to be used as submodels in higher complexity simulations, as well as their capability to directly extract symbolic models from data. Finally, we demonstrated the scalability and inference power of KAN-ODEs as standalone models in Section 3.3 by inferring complete data-driven solutions to PDEs using limited samples. Examples there include shock wave formation governed by the Burgers' equation and quantum wave function evolution governed by the Schrödinger equation, with an additional phase separation case using the Allen–Cahn equation provided in Appendix C.

3.1. KAN-ODEs vs. Neural ODEs: Extensive comparison via Lotka–Volterra equations

In this section, we targeted the Lotka–Volterra predator–prey model shown in Eq. (11). We replicated the simulation parameters used in the Neural ODE study of [9]. Namely, we used $\alpha = 1.5$, $\beta = 1$, $\gamma = 1$, and $\delta = 3$ with an initial condition of $\mathbf{u}_0 = [x_0, y_0] = [1, 1]$ and a time span of $t \in [0, 14]$ s. The temporal grid used for data generation, training, and loss evaluation had a spacing of 0.1 s. The `Tsit5` ODE integrator was used to solve the governing equations for the true ODE trajectory (i.e. the "ground truth" data used to train and validate the KAN-ODEs). The first 3.5 s of this ground truth trajectory was used to train the KAN-ODEs (Fig. 3(A)) via Eqs. (9) and (10), while the remainder of the time history (3.5 s to 14 s) was withheld to validate the ability of the KAN-ODEs to forecast the states at unseen times. While the training data is reported at 0.1 s increments, the `Tsit5` adaptive solver used with the KAN-ODEs requires much finer fidelity in its gradient accuracy. We tested KAN-ODEs with the various architectures shown in Table 1. The inputs to each KAN layer in this example were normalized to be on the $[-1, 1]$ range required by the RBF networks via the hyperbolic tangent function, as in [32].

$$\begin{aligned}\frac{dx}{dt} &= \alpha x - \beta xy,\\ \frac{dy}{dt} &= \gamma xy - \delta y.\end{aligned} \tag{11}$$

Loss profiles from training a 240-parameter KAN-ODE system of shape [2, 10, 5], [10, 2, 5] as a representative case are shown in Fig. 3(B1). Strong convergence is observed at or even before 10^4 epochs into training, while convergence into the 10^{-7} range of MSE loss occurs toward 10^5 epochs. The prediction of the coverged KAN-ODEs is shown in Fig. 3(A), where the excellent agreement in the testing window ($t \in [3.5, 14]$) is nearly indistinguishable from the agreement in the training window ($t \in [0, 3.5]$). We remark again that the trained KAN-ODEs are reconstructing both the seen training data (i.e. the ground truth ODE solution from 0 s to 3.5 s) and unseen testing data (3.5 s to 14 s) given only the initial condition ($\mathbf{u}_0 = [1, 1]$), and can deliver output values on any temporal grid, thanks to its coupling to an ODE solver.

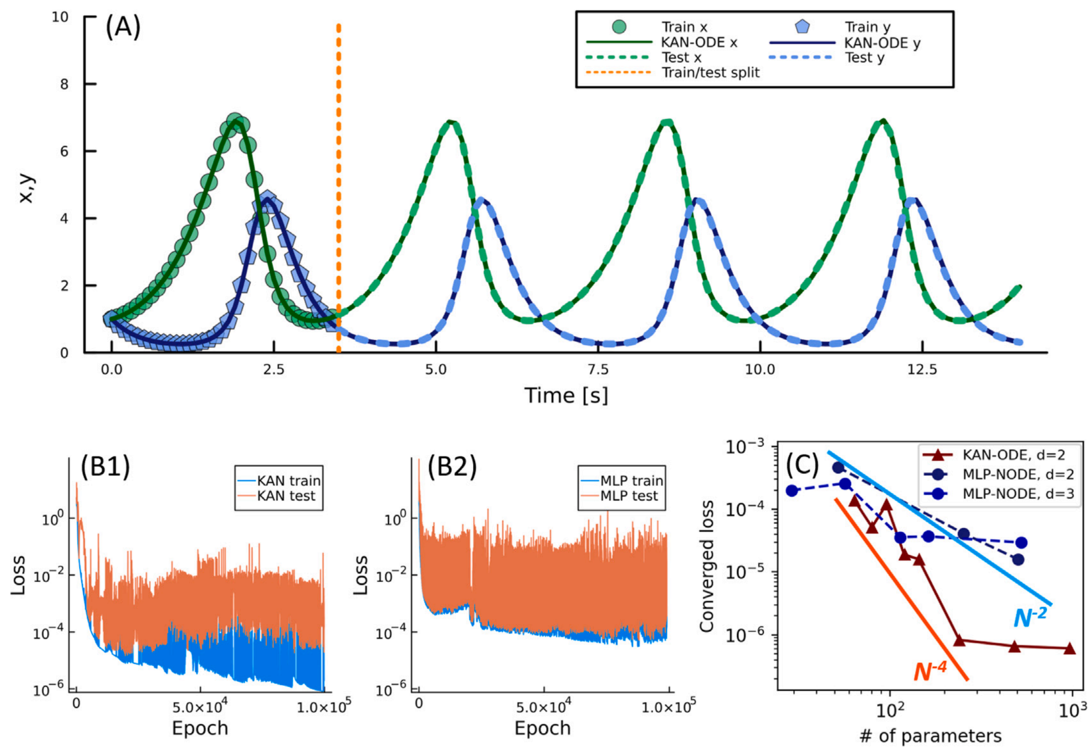

Fig. 3. Comparison between KAN-ODEs and Neural ODEs for Lotka–Volterra predator–prey model. (A) Synthetic data (training and testing) and reconstruction via KAN-ODEs. (B) Loss profile during training for (B1) KAN-ODEs and (B2) MLP-based Neural ODEs of comparable sizes (240 and 252, respectively). (C) Comparison of error between converged KAN-ODEs and Neural ODEs using different model sizes, and two MLP depths ($d = 2$ and $d = 3$). Neural scaling rates of N^{-2} and N^{-4} plotted for comparison, as per the theory in [26].

To contextualize these results, benchmark tests were performed to compare the KAN-ODEs with standard, MLP-based Neural ODEs. We draw comparisons in terms of accuracy, convergence speeds, and neural scaling in Section 3.1.1; and interpretability and generalization in Section 3.1.2. In Section 3.1.2, we additionally discuss the impacts of sparsified and symbolic KAN-ODEs.

3.1.1. Benchmark tests and neural scaling behavior

The MLP size and training procedure were taken from the Neural ODE work of [9] to avoid biasing the results in favor of the KAN-ODEs. Namely, an MLP with a single hidden layer comprising 50 nodes and the hyperbolic tangent activation function replaced the KAN in Eq. (7). This MLP contained 252 trainable parameters (refer to the bold architecture in Table 1), which is comparable (and in fact slightly larger) than the 240 used in the reported KAN-ODEs, facilitating fair comparison. Other relevant training details such as the train/test time windows, temporal grid, Lotka–Volterra equation parameters, and ODE solver remained identical to the KAN-ODEs example, which itself was modeled as directly as possible after the Neural ODE efforts of [9].

In this one-to-one comparison using a data generation and training regime borrowed from a well-cited Neural ODE work, our proposed KAN-ODEs beat the performance of the Neural ODEs in all key metrics. Inspection of the loss profiles of Fig. 3(B1–B2) shows that the KAN-ODEs converged to a substantially smaller training loss of 8.3×10^{-7} over 10^5 epochs compared to the similarly-sized Neural ODEs, which reached only 3×10^{-5} in the same number of epochs. Both loss profiles exhibit significant oscillatory behavior while still successfully training, which we theorize may be due to the data-lean, large-batch regime studied here (i.e. a single dataset, all time steps of which are optimized in one batch). These oscillations, which do not appear to affect the final result, are smaller in the training cycle of the KAN-ODEs than in that of the Neural ODEs. The KAN-ODEs qualitatively appear to overfit more than the Neural ODEs, especially in the later epochs, by inspection of the difference between training and testing losses. Quantitatively, however, they still beat the Neural ODEs in accuracy to the unseen testing data, indicating an objective improvement in performance when switching from the Neural ODEs to the KAN-ODEs. Even in the last 10% of the training cycle (epochs $90,000$–$100,000$), where the KAN-ODEs experienced a notable difference between training and testing losses, they still achieved an average testing loss of 6.8×10^{-3}, with a minimum value in that range of 1.9×10^{-5}. The Neural ODEs in this same range averaged a much larger 1.4×10^{-2}, with a minimum value of 5.4×10^{-5}. If training for the KAN-ODEs was stopped early at 2×10^4 epochs when the testing error just began to creep upward, these statistics would tend even more strongly in their favor. At this early stopping point, little to no indication of overfitting is observed when comparing its own training and testing losses, and all loss metrics still outperform those of the Neural ODEs (even when the latter are trained to the full 1×10^5 epochs). The KAN-ODEs-based solution of Fig. 3(A) can

Table 1
Lotka–Volterra tests with Neural ODEs and KAN-ODEs of varying size. Bold entries are studied further in Fig. 3.

	Depth	Layer width	Grid size	Activation function	No. Params	Train loss
Neural ODE (MLP)	2	10	N/a	tanh	52	4.7×10^{-4}
	2	**50**	N/a	**tanh**	**252**	$\mathbf{4.1 \times 10^{-5}}$
	2	100	N/a	tanh	502	1.6×10^{-5}
	3	3	N/a	tanh	29	2.0×10^{-4}
	3	5	N/a	tanh	57	2.6×10^{-4}
	3	8	N/a	tanh	114	4.6×10^{-5}
	3	10	N/a	tanh	162	3.7×10^{-5}
	3	20	N/a	tanh	522	3.0×10^{-5}
KAN-ODE	2	4	3	*learned*	64	1.4×10^{-4}
	2	4	4	*learned*	80	5.2×10^{-5}
	2	4	5	*learned*	96	1.2×10^{-4}
	2	6	4	*learned*	120	1.9×10^{-5}
	2	6	5	*learned*	144	1.6×10^{-5}
	2	**10**	**5**	***learned***	**240**	$\mathbf{8.3 \times 10^{-7}}$
	2	20	5	*learned*	480	6.6×10^{-7}
	2	40	5	*learned*	960	6.1×10^{-7}

additionally be generated at arbitrary time steps (e.g. 0.01 s rather than 0.1 s) with no detectable change to the testing results, indicating no overfitting to the specific training timesteps used here.

In terms of computational cost, we found that the KAN-ODEs iterated relatively slowly but actually converged 3–4x faster than the Neural ODEs. On a single CPU core, the 240-parameter KAN-ODE system took around 19 min per 10^4 epochs, compared to the 252-parameter Neural ODE system which took closer to 7 min per 10^4 epochs. However, the KAN-ODEs reached 2.6×10^{-5} training loss in just 10^4 epochs, already beating the best-performing, 10^5 epoch Neural ODEs result of 3.0×10^{-5} training loss. Thus, while the KAN-ODEs iterated 2.5 to 3x slower than the Neural ODEs, they were able to reach the same predictive performance with 10x fewer epochs, resulting in an overall effective 3–4x speedup over the Neural ODEs.

Up to this point, comparisons have been drawn exclusively between a single Neural ODE architecture and a single KAN-ODE architecture. Here, we discuss the effects of varied network architectures on the performance of KAN-ODEs and Neural ODEs, the results of which demonstrate a key scaling improvement of KAN-ODEs. Liu et al. [26] proposed that, thanks to the third-order splines used in the Kolmogorov–Arnold representations, KANs benefit from fourth-order neural scaling, or a quartic decrease in loss with respect to the KAN size (parameterized here as the number of trainable parameters). MLPs, on the other hand, often saturate quickly and can struggle to reach even first-order convergence [26,43]. We investigated this phenomenon here by varying the layer width and grid size of KAN-ODEs, and by varying the layer width and depth of MLP-based Neural ODEs. Fig. 3(C) shows that, even with the modified RBF basis function, we appear to capture the N^{-4} B-spline KAN scaling rate with the KAN-ODEs, while the two depths of MLP appear to be converging at a rate slower than N^{-2}. With just the 240 KAN parameters studied in Fig. 3(A) and (B1) we were able to run the fourth-order convergence of the KAN-ODEs to saturation (larger KANs see no substantial additional performance gain), while MLP-based Neural ODEs would theoretically require upwards of thousands or even tens of thousands of parameters to reach similar performance, if they did not plateau earlier. The specific model sizes used in this convergence test are shown in Table 1, where the models used for Fig. 3(A) and (B1-B2) are in bold. To summarize, we found here that KAN-ODEs beat Neural ODEs in terms of data-fitting performance and computational efficiency in a systematic study of a broad range of network architectures and sizes.

3.1.2. Interpretation and generalization of KAN-ODEs with varying sizes

In this section we implemented sparsification, pruning, and symbolic regression to demonstrate a framework for developing interpretable KAN-ODE solvers. We additionally heuristically explored the generalizability of the various MLP and KAN-based models, where we found that KAN-ODEs have strong generalization potential even with limited training data. In addition to the 252-parameter Neural ODE and 240-parameter KAN-ODE systems (trained in Section 3.1.1) as reference cases, three models were studied: sparse KAN-ODEs trained via parameter regularization and pruning, symbolic KAN-ODEs developed via symbolic regression on each univariate activation of the sparse KAN-ODEs, and a symbolic representation for the entire model via symbolic regression on the global input–output pairs of the sparse KAN-ODEs.

Sparsification, as proposed in [26] and discussed in [44], constitutes an additional term in the loss function to penalize non-zero trainable parameters, driving a given model toward sparser representations with fewer large activations. Here we use the simple L1 norm,

$$\mathcal{L}(\boldsymbol{\theta}) = \mathrm{MSE}\left(\mathbf{u}^{\mathrm{KAN}}(t, \boldsymbol{\theta}), \mathbf{u}^{\mathrm{obs}}(t)\right) + \gamma_{sp}|\boldsymbol{\theta}|_1. \tag{12}$$

This is identical to Eq. (9), with the inclusion of the L1 penalty term with γ_{sp} as a hyperparameter, which drives unnecessary activations toward zero. The logical next step is to reduce the size and complexity of the now-sparse KAN-ODEs by eliminating any trivial nodes via pruning. To do so, we compare the training inputs and outputs of each node against a pruning parameter, γ_{pr}. Any node where the magnitude of at least one input or output exceeds γ_{pr} is kept in the KAN-ODEs, while those with all inputs and outputs smaller than γ_{pr} are pruned. We began with a [2, 10, 5], [10, 2, 5] structure for a KAN (referred to henceforth as the dense

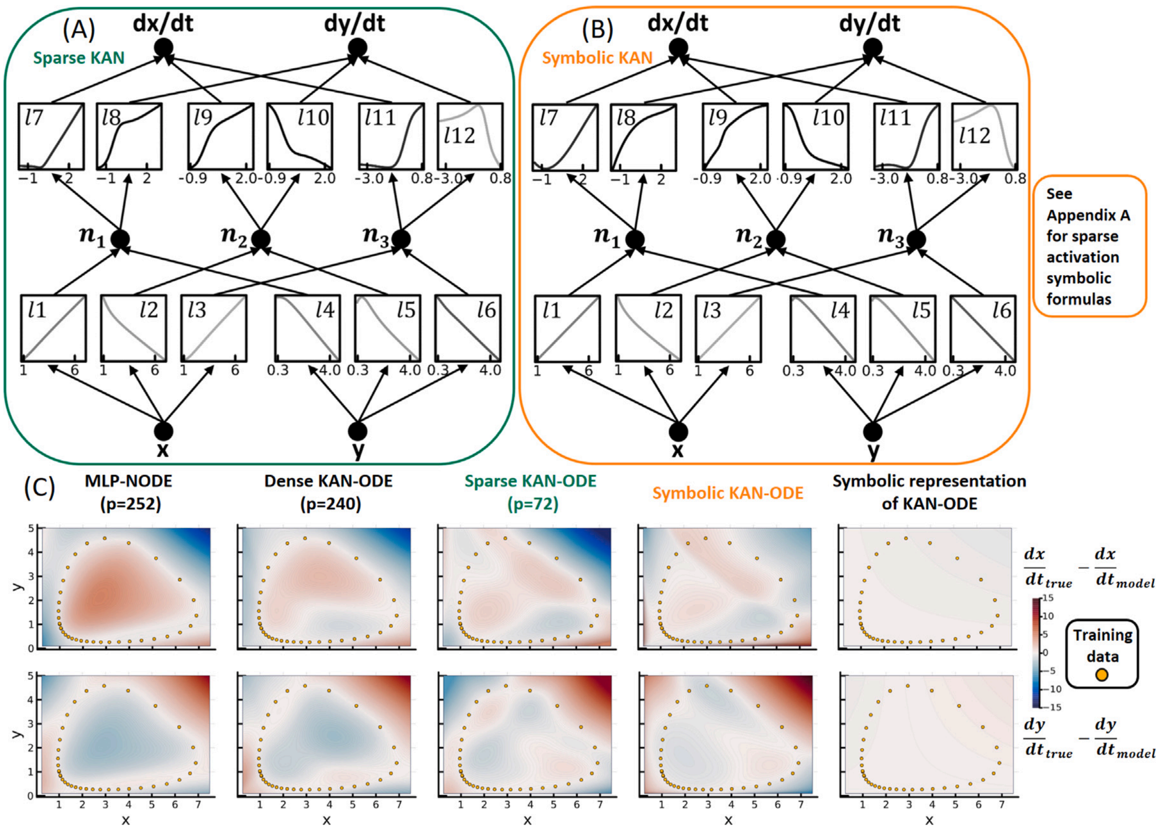

Fig. 4. Sparsification, pruning, symbolic regression, and generalization results for Lotka–Volterra dynamics. (A) Sparse KAN with 72 parameters ([2, 3, 5], [3, 2, 5]), pruned from an initial 240 parameters ([2, 10, 5], [10, 2, 5]). (B) Symbolic KAN derived from (A), where each of the twelve activations is replaced by a univariate symbolic expression. (C) Generalization error when extrapolating outside of the yellow (x, y) points explored in the training data. Each column represents a different gradient getter and has one row each for dx/dt and dy/dt. Generalization is seen to improve when moving from MLP gradients to increasingly sparse KANs, and finally to symbolic representations fitted to KANs. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

KAN), based on observation from Fig. 3(C) of this architecture's optimal performance. To sparsify the KAN-ODEs, we included the regularization term of Eq. (12) with $\gamma_{sp} = 5 \cdot 10^{-4}$ in the training process. To prune, we trained these KAN-ODEs to convergence, then eliminated nodes that did not satisfy $\gamma_{pr} = 10^{-2}$ before training the remaining KAN-ODEs to convergence once again. Fig. 4(A) shows the sparse KAN-ODEs with a final pruned size of [2, 3, 5], [3, 2, 5] trained using this procedure. Note that the darkness of the activation function curves is determined by the ratio of the outputs to the inputs (larger outputs have thicker lines).

Next, the symbolic KAN-ODEs took the sparse KAN-ODEs and fit symbolic expressions to each of their twelve univariate activation functions using the SymbolicRegression.jl package [45] with the four fundamental binary operators $[+, -, \times, /]$. Detailed results for each of these regressions is available in Appendix A. Here, we plot the twelve symbolic activations for visualization in Fig. 4(B). To summarize, we first sparsified a 240-parameter, 10-node KAN into a 72-parameter, 3-node KAN. Then, we replaced these 72 parameters with just twelve univariate activation functions, a handful of which in the first layer are simple linear functions of the form $y = mx + b$. In terms of interpretability, we moved from a black box MLP-based Neural ODE to a visualizable yet bulky KAN, then to a visualizable and compact KAN, and finally to a straightforward collection of symbolic relationships that capture the Lotka–Volterra dynamics well. None of these steps required knowledge of the system outside of the provided training data.

Finally, we ran the entire Sparse KAN through SymbolicRegression.jl (rather than each activation separately) and found the following relationships which comprise the KAN representation,

$$\begin{aligned} \frac{dx}{dt} &= 1.495x - 0.986xy, \\ \frac{dy}{dt} &= 0.970xy - 2.929y, \end{aligned} \tag{13}$$

which correspond nearly identically to our model parameters of 1.5, 1, 1, and 3. This near-perfect result is effectively the pinnacle of interpretability and accuracy, though we believe is not a realistic option for most systems of larger-scale merit. Complete symbolic regression on an entire learned model or dataset is often not feasible, especially for complex models with higher dimensional inputs and outputs than are studied in this relatively straightforward ODE system [26]. Even when numerically manageable, clean convergence to compact functional forms is not guaranteed. On the other hand, a symbolic KAN like that shown in Fig. 4(B) requires symbolic regression only on a series of straightforward univariate activations and retains entirely interpretable intermediate steps. We present this global symbolic representation result as an interesting aside and a benchmark for comparison, but propose that sparse KAN-ODEs and symbolic KAN-ODEs are the most likely to have broader applicability in more realistic systems depending on the desired degree of interpretability and accuracy.

We move next to a discussion of the generalizability of KAN-ODEs. To conform to prior studies and faciliate fair comparison against MLP-based Neural ODEs, all models used previously in this Lotka–Volterra section were trained on data generated from a single initial condition, as was done in the Neural ODE study of [9]. There exist many other potential (x, y) combinations, however, and here we heuristically tested these trained models on their ability to generalize to completely unseen conditions.

The gradient error landscapes of the five models studied here (MLP-NODEs, Dense KAN-ODEs, Sparse KAN-ODEs, Symbolic KAN-ODEs, and symbolic representation of KAN-ODEs) are shown from left to right, respectively, in Fig. 4(C), where the top row is the dx/dt error and the bottom row is the dy/dt error. The results from the MLP-NODEs in the first column show near-zero error in the immediate vicinity of the training data, as expected from the performance metrics of Fig. 3. However, they also show significant gradient error in almost all (x, y) ranges not directly on or immediately near the training data points, revealing the expected weak generalizability of the standard Neural ODEs due to the fact that training data is limited to the marked points only.

The gradient error landscape of the dense KAN-ODEs beats the MLP in generalizability, with much smaller gradient error values throughout the domain and especially inside of the yellow training points where not only is the overall error substantially reduced, but we begin to additionally observe the formation of thin bands of near-perfect reconstruction away from the data points. This result is rather interesting, as it was originally proposed in [26] that KANs beat MLPs in generalization thanks to their smaller number of parameters, but we observe here a substantial improvement with similarly sized models. In any case, these results suggest the potential of KAN-ODEs in the development of powerful models from limited data. In a brief aside, we also remark on the current result's support of our previous argument (in Section 3.1.1) that the fully-trained dense KAN-ODEs do not overfit to the training data points, given their near-perfect interpolation between these points along the training profile. This result is unsurprising given the adaptive time stepping of the `Tsit5` solver, but is nonetheless a strong result, especially when combined with the additional visible improvement in generalization outside of the training profile. If overfitting becomes substantial in larger-scale KAN-ODE applications, there are a handful of general neural network techniques and Neural-ODE adapted techniques that can be adapted to address this [46–48] including early stopping and network reduction (as proposed here) as well as data expansion, noise injection, and dropout.

The gradient error landscape of the sparse KAN-ODEs shows improvements over the landscape of the dense KAN-ODEs. Inside the yellow data points, we see further formation of near-perfect gradient reconstruction bands, surrounded by relatively low error regions. Results toward the edges of the landscape are mixed: the dx/dt reconstruction captures the top left of the (x, y) domain with excellent accuracy but suffers from a large error in the upper right portion of the domain, while the dy/dt reconstruction performs well toward low y values, but suffers at high y values. On the whole, we remark an overall improvement in generalization with a nearly 4x reduction in KAN size, indicating the utility of sparse KAN-ODEs not just for compact and interpretable models, but also for improved generalization and extrapolation capabilities thanks to their smaller size. This result corroborates similar observations from [26] of the increase in generalizability with decreased KAN parameter counts.

The Symbolic KAN-ODEs see remarkable improvement in the dx/dt generalization, with the low-error area being pushed out significantly toward high x and y values, further growth of the white bands of near-zero error throughout the domain, and a general lightening of the plot indicating better extrapolation and generalization. The dy/dt reconstruction sees modest improvement, especially inside the circle of data points. We propose that this generalization improvement occurs due again to the more compact forms present in the symbolic KAN-ODEs when compared to even the sparse KAN-ODEs, and the extrapolation benefits that come with smaller system representations [26]. In the significantly out-of-training regions of dx/dt where the symbolic KAN-ODEs succeed and the gridded basis functions of the dense KAN-ODEs and Sparse KAN-ODEs fail, we suggest that the input samples may be exiting the trained region of the RBF basis functions yet remaining within the useful range of the expressions used in the symbolic KAN-ODEs.

The symbolic representation of the entire KAN obviously generalizes with near-perfect accuracy thanks to its discovery of the exact underlying model form (with near-perfect parameters). This result is plotted in the final column of Fig. 4(C) for reference, but we again emphasize that the other three KAN models are more reasonable choices for realistic applications: Dense KAN-ODEs succeeded with remarkably low training and testing error, sparse KAN-ODEs retained much of this low error while appearing to improve generalizability thanks to their smaller size, and Symbolic KAN-ODEs were substantially more interpretable and human-readable while also appearing to improve generalizability thanks to their smaller size and replacement of limited-scope basis functions with standard symbolic functions.

In this Lotka–Volterra example, we systematically and comprehensively explored the use of KAN-ODEs in a straightforward ODE system and benchmarked their performance against comparable MLP-based Neural ODEs in terms of accuracy, speed, model size, convergence rate, interpretability, and generalization. We found that KAN-ODEs are an extremely efficient method to represent the predator–prey dynamics of this example, and beat Neural ODEs in all relevant quantitative metrics. We additionally explored the interpretability of KAN-ODEs via symbolic regression of each activation function, and then heuristically evaluated a series of MLP and KAN-based ODE approaches in terms of generalizability to unknown data, where we again found KAN-ODEs superior. In the next sections, we present larger-scale examples to showcase the flexibility and inference capabilities of KAN-ODEs.

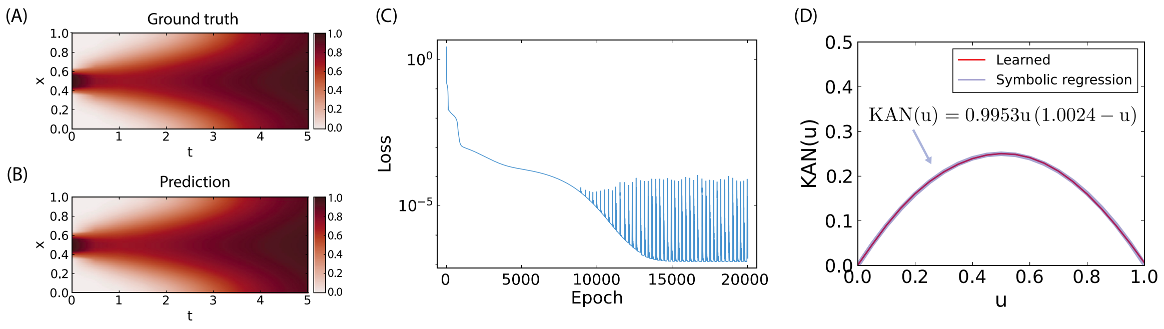

Fig. 5. *Fisher–KPP equation*: (A) Solution field $u(x,t)$ of the ground truth. (B) Solution field $u(x,t)$ of the prediction with a learned KAN-ODE model. (C) Loss function. (D) Learned hidden physics of the reaction source term in the Fisher–KPP equation and its symbolic form.

3.2. Modeling hidden physics in PDEs: Fisher–KPP PDE

In this section, we introduce the capability of KAN-ODEs to learn the hidden physics in PDEs. Key features demonstrated here are the flexibility of KAN-ODEs to be paired with higher-complexity solvers on arbitrary temporal grids, and the capability of KAN-ODEs to directly extract hidden symbolic functional relationships from training data. We demonstrate this using the Fisher–KPP equation representing a reaction–diffusion system,

$$\frac{\partial u}{\partial t} = D\frac{\partial^2 u}{\partial x^2} + ru(1-u), \tag{14}$$

where D is the diffusion coefficient and r is the local growth rate. We assume that the reaction term $ru(1-u)$ is unknown and to be modeled with a KAN.

Training data was synthesized by solving Eq. (14) in the domain of $x \in [0,1]$, $t \in [0,5]$, given the model parameters $D = 0.2$ and $r = 1.0$. The initial condition and boundary conditions were

$$u(x,0) = \frac{1}{2}\left[\tanh\left(\frac{x-0.4}{0.02}\right) - \tanh\left(\frac{x-0.6}{0.02}\right)\right], \tag{15}$$

$$u(0,t) = u(1,t), \tag{16}$$

$$\frac{\partial u}{\partial x}(0,t) = \frac{\partial u}{\partial x}(1,t).$$

Note that this problem setup was adopted from [9]. The spatial grid was discretized with $\Delta x = 0.04$, and the equation was solved with a time step of $\Delta t = 0.5$ using the ODE integrator of `Tsit5` (Tsitouras 5/4 Runge–Kutta method [41]). The obtained solution field $u(x,t)$ is shown in Fig. 5(A) and used as training data.

To model the unknown reaction source term in Eq. (14), we formulated the KAN-ODE such that

$$\frac{\partial u}{\partial t} = D\frac{\partial^2 u}{\partial x^2} + \mathrm{KAN}\,(u,\boldsymbol{\theta}). \tag{17}$$

To utilize backpropagation with the *adjoint* sensitivity method, Eq. (17) was discretized using a central difference scheme and converted to an ODE system using the Method of Lines. The KAN was constructed with a single layer comprising a single node, [1,1,10].

The prediction with a learned model is illustrated in Fig. 5(B), showing accurate reconstruction of the solution field $u(x,t)$ after as few as 5000 updates (Fig. 5(C)) and demonstrating that a single learned activation function comprised of ten gridded basis functions ([1,1,10]) is sufficient to represent the reaction source term (see Appendix B for a detailed discussion of accuracy with varying KAN-ODE architectures). This resultant activation function is shown in Fig. 5(D). We further propose expressing this single activation function symbolically, as was introduced in [26] as a feature of KAN layers. Symbolic regression on this single KAN-ODE activation using SymbolicRegression.jl [45] given the candidate symbolic expressions of $[+,-,\times,/,\sin,\cos,\exp]$ returned the following expression,

$$\mathrm{KAN}\,(u) = 0.995311u(1.002448-u). \tag{18}$$

Given its single-layer, single-node construction, this symbolic expression is the KAN-ODE's approximation of the true reaction source term in Eq. (14). Knowing $r = 1.0$, we see that the KAN-ODE's derived expression is remarkably close to the original formulation. Thus, the KAN-ODE was not only capable of learning the dynamics of physics hidden within a PDE submodel from measurable quantities, but also of returning an accurate and human-interpretable symbolic function that mimics the true governing law with excellent accuracy.

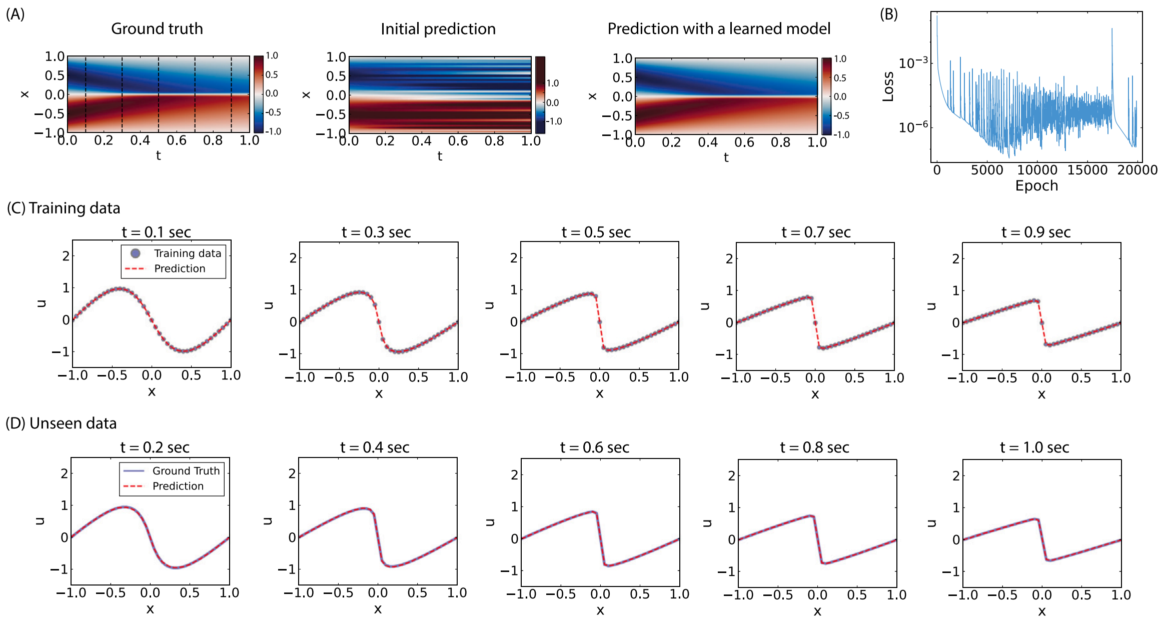

Fig. 6. *Burgers' equation*: (A) Solution fields $u(x,t)$ for the ground truth, prediction with the initialized model, and prediction with the trained model. The black dashed lines in the solution field of the ground truth subplot indicate the training data. (B) Loss function. (C) Training data and inferred solution profiles at times t = 0.1, 0.3, 0.5, 0.7, and 0.9 s. (D) Unseen ground truth data and inferred solution profiles at selected times t = 0.2, 0.4, 0.6, 0.8, and 1.0 s.

3.3. Data-driven solutions of PDEs

3.3.1. Burgers' equation

Our third example explores the use of KAN-ODEs to infer the unknown hidden states $u(x,t)$ of a PDE system, effectively representing all of its spatiotemporal dynamics. Here we considered the Burgers' convection–diffusion equation as an example.

$$\frac{du}{dt} + u\frac{du}{dx} = \frac{0.01}{\pi}\frac{d^2u}{dx^2}. \tag{19}$$

The computational domain was defined as $x \in [-1.0, 1.0]$ and $t \in [0.0, 1.0]$.

In order to prepare training data, Eq. (19) was discretized using a central difference scheme with $\Delta x = 0.05$ such that $\mathbf{u}(t) = \left[u\left(x_0,t\right), u\left(x_1,t\right), \dots, u\left(x_N,t\right)\right]$. The initial and boundary conditions were defined as

$$u(x,0) = -\sin(\pi x), \tag{20}$$

$$u(-1,t) = 0, \tag{21}$$

$$u(1,t) = 0.$$

Then, the discretized governing equation was solved using these initial and boundary conditions with a time step of $\Delta t = 0.01$ using the ODE integrator of `Tsit5` (Tsitouras 5/4 Runge–Kutta method [41]). The resultant solution field is illustrated in Fig. 6(A). The solution profiles at the selected times ($t \in \{0.1,\ 0.3,\ 0.5,\ 0.7,\ 0.9\}$) were collected as training data, with an emphasis here on a sparse training dataset that tests the extrapolation capability of the KAN-ODEs. To learn the latent solution, we modeled the KAN-ODEs such that

$$\frac{\partial \mathbf{u}(t)}{\partial t} = \mathrm{KAN}(\mathbf{u}(t), \theta). \tag{22}$$

In this formulation, the KAN works as a non-linear operator for the partial differential equation (PDE) solution. The KAN-ODEs were constructed with 2 layers: [51, 10, 5], [10, 51, 5]. Ten nodes in the intermediate layer were used to preserve the high-dimensional information. Note that the number of input and output dimensions of the KAN was determined by the number of discretized state variables in $\mathbf{u}$. The KAN parameters were updated with a learning rate of 0.01 using the ADAM optimizer.

After 20,000 training epochs (Fig. 6(B)), the KAN-ODEs can be seen in Fig. 6(A) to predict the temporal evolution of the PDE well despite only receiving five training snapshots. These snapshots and their accurate KAN-ODE dynamical representations are shown in Fig. 6(C), where strong agreement is seen at all spatial locations, even those near the shock wave where the training data spatial sampling is very sparse (in addition to sparse temporal sampling). More importantly, the trained KAN-ODEs are able to extrapolate effectively to infer the solution fields at the unseen testing times in Fig. 6(D), once again with strong performance

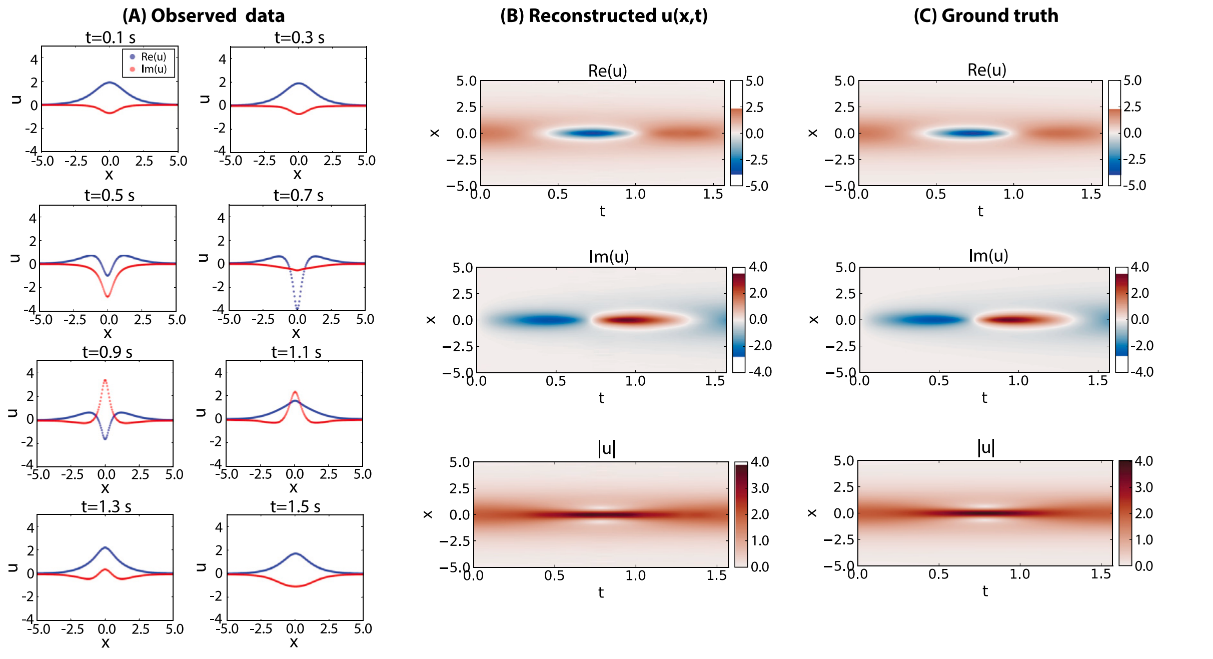

Fig. 7. *Schrödinger equation*: (A) Observed data $u(x,t) = \mathrm{Re}(u) + i\mathrm{Im}(u)$ at the selected times of $t = 0.1$, 0.3, 0.5, 0.7, 0.9, 1.1, 1.3, and 1.5 s as training data. Blue circles denote $\mathrm{Re}(u)$ and red circles denote $\mathrm{Im}(u)$. (B) Reconstructed $u(x,t)$. (C) Ground truth $u(x,t)$. Note that $\mathrm{Re}(u)$ and $\mathrm{Im}(u)$ denote the real and imaginary parts of u, respectively. Also, $|u| = \sqrt{\mathrm{Re}(u)^2 + \mathrm{Im}(u)^2}$. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

at the shock. As a whole, this case demonstrates the ability of KAN-ODEs to leverage their temporal grid agnosticism to accurately predict a system's behavior at unseen times, and to extrapolate significant meaning from spatially and temporally sparse datasets. An in-depth discussion of the model accuracy and KAN architecture can be found in Appendix B.

3.3.2. *Schrödinger equation*

The last example demonstrates surrogate modeling with KAN-ODEs in a more involved dynamical system with complex-valued states. The Schrödinger equation is a classical field equation used in quantum mechanics.

$$i\frac{\partial u}{\partial t} + \frac{1}{2}\frac{\partial^2 u}{\partial x^2} + |u|^2 u = 0, \tag{23}$$

where $u \in \mathbb{C}^1$. The time and spatial domains were set to $t \in [0, \pi/2]$ and $x \in [-5, 5]$. The initial condition and boundary conditions were defined as following in Eqs. (24)–(25). Then, the governing equations were solved with a time step of $\Delta t = 0.01$ and grid size of $\Delta x = 0.05$ using the stiff ODE integrator of `Rodas5` (A 5th order A-stable stiffly stable Rosenbrock method with a stiff-aware 4th order interpolant [39,49]). Note that the problem settings used in this example were identical to those in [6].

$$u(x,0) = \mathrm{sech}(x), \tag{24}$$

$$u(-5,t) = u(5,t), \qquad \frac{\partial u}{\partial x}(-5,t) = \frac{\partial u}{\partial x}(5,t). \tag{25}$$

The KAN-ODEs as a surrogate model (Eq. (22)) were constructed with 2 layers: [402, 10, 10], [10, 402, 10]. The input dimension was doubled from the dimension of the discretized u (201 to 402) to account for both the real and imaginary values of u.

We once again gave the KAN-ODEs training data at just eight selected times in Fig. 7(A), and yet the trained KAN-ODEs successfully reconstruct the complete spatiotemporal profile of the wave field $u(x,t)$ (Fig. 7(B)), which agrees with the ground truth $u(x,t)$ (Fig. 7(C)) at all times. We find this result remarkable, especially as the KAN-ODEs had zero knowledge of the underlying physics and instead were forced to infer the complete solution to the nonlinear and complex dynamics present in this system from eight samples only.

In this section, we demonstrated the robustness of KAN-ODEs by learning shock formation behavior and complex-valued wave propagation in data-lean frameworks and without prior knowledge of the system dynamics. In both cases the KAN-ODEs were able to match the training data and generalize well in the significant windows between training data slices, showing promise as a novel data-driven dynamical system modeling approach. With these PDE examples, we showed that KAN-ODEs can retrieve not only temporal dynamics but also spatial information without special treatment in the KAN architecture.

4. Conclusions

This work proposed the use of Kolmogorov–Arnold networks as gradient-getters in the Neural ODE framework. This novel combination of data-driven techniques advances our capabilities in the field of scientific machine learning to efficiently infer interpretable and modular dynamical system models from various sources of data. The replacement of MLPs with KANs in this framework preserves the black-box style approach with zero prior knowledge of the dynamical system physics needed for training. However, it opens the door to significant interpretability of the learned model ("opening" the black box) via activation function visualization and symbolic regression. Quantitatively, the KAN-ODE framework was also shown to outperform similar MLP-based Neural ODEs in a wide array of metrics including training accuracy, testing accuracy, convergence speed, neural scaling rate, and generalizability. The strong neural scaling offered by the Kolmogorov–Arnold representation held when applied as a KAN-ODE with RBF basis functions, allowing for significantly better training and testing performance in a fraction of the computational time and with a significantly smaller network size. We additionally demonstrated the capability of KAN-ODEs to extrapolate information from sparse or limited temporal data to the rest of a solution domain even in stiff and complex-valued nonlinear PDEs, and to learn symbolic expressions for PDE source terms with high accuracy via optional postprocessing. In addition to improvements over MLP-based Neural ODEs, KAN-ODEs offer an alternative to SINDy, PINN, and similar interpretable machine learning techniques in their flexibility and generality - no prior knowledge or assumptions are needed about the governing laws or candidate function space to train the KAN-ODE system effectively, although such information can be added if desired as shown here in the source term regression of the Fisher–KPP PDE and the symbolic KAN leveraged for Lotka–Volterra dynamics. KAN-ODEs are a compact, efficient, flexible, and interpretable approach for data-driven modeling of dynamical systems that show significant promise for use in the field of scientific machine learning.

CRediT authorship contribution statement

Benjamin C. Koenig: Writing – original draft, Software, Methodology, Investigation. **Suyong Kim:** Writing – original draft, Software, Methodology, Investigation, Conceptualization. **Sili Deng:** Writing – review & editing, Resources, Funding acquisition.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

The KAN-ODE codes used to generate the results presented in this work are publicly available at https://github.com/DENG-MIT/KAN-ODEs.

Acknowledgments

This work is supported by the National Science Foundation (NSF) under Grant No. CBET-2143625. BCK is partially supported by the NSF Graduate Research Fellowship under Grant No. 1745302.

Appendix A. Sparse regression on Lotka–Volterra KAN-ODEs

Symbolic functions corresponding to the Symbolic KAN-ODEs plotted in Fig. 4(B) are provided in Table A.1. See Fig. 4(B) for a visual representation of the twelve activations and of the hidden layer nodes n_1 through n_3. The selection of candidate functions was carried out to balance accuracy with sparsity. For example, a linear term suffices to capture the linear behavior of $l1$ connecting x and n_1, while $l2$ connecting x and n_2 requires an additional $1/x$ term due to its strong nonlinear behavior. Similarly, the relatively complex $l11$ connecting n_3 to dx/dt uses a linear term in addition to a polynomial division term. All activations were derived independently of each other, based on the immediately relevant univariate input/output pairs.

Appendix B. Varying KAN-ODE Architectures for Fisher–KPP and Burgers' equations

Additional tests were performed to provide KAN-ODE performance metrics for varying KAN architectures in the *Fisher–KPP equation* and *Burgers' equation* examples. Table B.1 shows the test matrix with different sizes of KANs. For the *Fisher–KPP equation*, a relatively small KAN with very few parameters was used to model the hidden physics in the interpretable form (i.e. replacing only the source term). Larger KANs with more parameters (thanks to the larger input/output dimension) were constructed for the *Burgers' equation* to build a surrogate model that predicted the entire solution field. Fig. B.1 illustrates the training losses with respect to the number of parameters in these two examples. We surprisingly found in both cases that small architectures of KANs, i.e. those with zero to one hidden node and three to five grid points, can successfully learn these dynamical systems with a training loss smaller than 10^{-6}. Furthermore, in most cases, KAN-ODEs have a scaling rate of loss $\mathcal{L} \propto N^{-4}$, where N is the total number of parameters in the KAN. This result empirically supports the validity of the scaling law in the original KAN [26], as well as results seen in the Lotka–Volterra example shown in Fig. 3(C).

Table A.1
Symbolic KAN activations for Lotka–Volterra dynamics. The first layer takes the inputs x and y while the second layer takes the inputs n_1, n_2, and n_3. Note that $n_i(x, y) = n_i(x) + n_i(y)$ and $dx/dt(n_1, n_2, n_3) = dx/dt(n_1) + dx/dt(n_2) + dx/dt(n_3)$ by definition in the KAN.

Layer	Activation	Symbolic expression
1	$n_1(x)$	$0.545x - 0.204$
	$n_2(x)$	$-0.277x + 0.425 + \frac{0.794}{x}$
	$n_3(x)$	$0.334x - 0.635$
	$n_1(y)$	$-0.605y + 0.220 - \frac{0.120}{y}$
	$n_2(y)$	$-0.506y + 1.864 - \frac{0.108}{y}$
	$n_3(y)$	$-0.780y - 0.102$
2	$\frac{dx}{dt}(n_1)$	$0.090n_1^3 + 0.520n_1^2 + 0.890n_1 - 0.411$
	$\frac{dy}{dt}(n_1)$	$0.168n_1^3 - 0.887n_1^2 + 2.242n_1 + 0.748$
	$\frac{dx}{dt}(n_2)$	$-n_2^2 + 4.990n_2 + \frac{0.439n_2}{n_2^2 + 0.0695} - 1.652$
	$\frac{dy}{dt}(n_2)$	$-n_2 + \frac{-3.201n_2}{n_2^2 + 0.719n_2 + 0.675}$
	$\frac{dx}{dt}(n_3)$	$0.684n_3 + \frac{2.929n_3}{n_3^2 + 1.307n_3 + 1.41} - 0.490$
	$\frac{dy}{dt}(n_3)$	$-\frac{1.281n_3}{n_3^2 + 0.800} + 0.353$

Table B.1
KAN-ODE architectures tested for performance evaluation with the Fisher–KPP equation and the Burgers' equation.

	Depth	Layer width	Grid size	No. Params
Fisher–KPP equation	1	1	[2, 3, 5, 10, 20]	[3, 4, 6, 11, 21]
	2	3	[2, 3, 5, 10]	[18, 24, 36, 66]
Burgers' equation	2	1	[2, 3, 5, 10]	[246, 328, 492, 902]
	2	3	[2, 3, 5, 10]	[738, 984, 1476, 2706]
	2	5	[2, 3, 5, 10]	[1230, 1640, 2460, 4510]
	2	10	[2, 3, 5, 10]	[2460, 3280, 4920, 9020]
	2	20	[2, 3, 5, 10]	[4920, 6560, 9840, 18040]

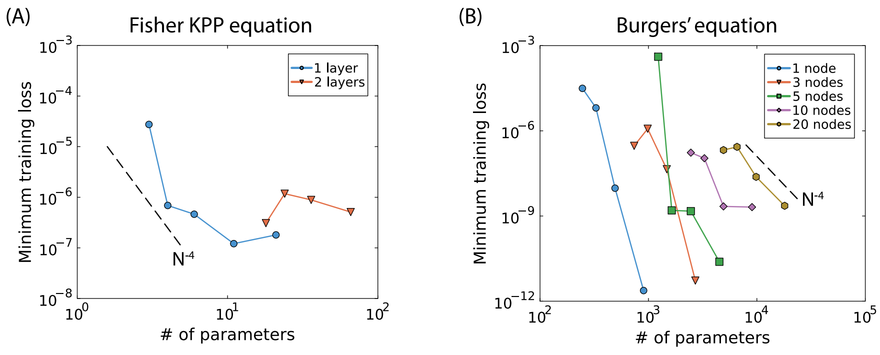

Fig. B.1. *Performance test with different KAN-ODE architectures*: Training loss as a function of the total number of parameters. (A) Fisher–KPP equation. (B) Burgers' equation. Loss scaling of $\mathcal{L} \propto N^{-4}$, where N is the total number of parameters in the KAN, is also illustrated as a reference line.

Appendix C. Additional example with the Allen–Cahn equation

In this appendix, we investigated the Allen–Cahn equation which describes phase separation in multi-component alloy systems. The Allen–Cahn equation was used as another canonical example to demonstrate two KAN-ODE tasks: modeling hidden physics and reconstructing complete solution fields. The Allen–Cahn equation is expressed by Eq. (C.1).

$$\frac{\partial u}{\partial t} = 0.0001\frac{\partial^2 u}{\partial x^2} + 5u - 5u^3. \tag{C.1}$$

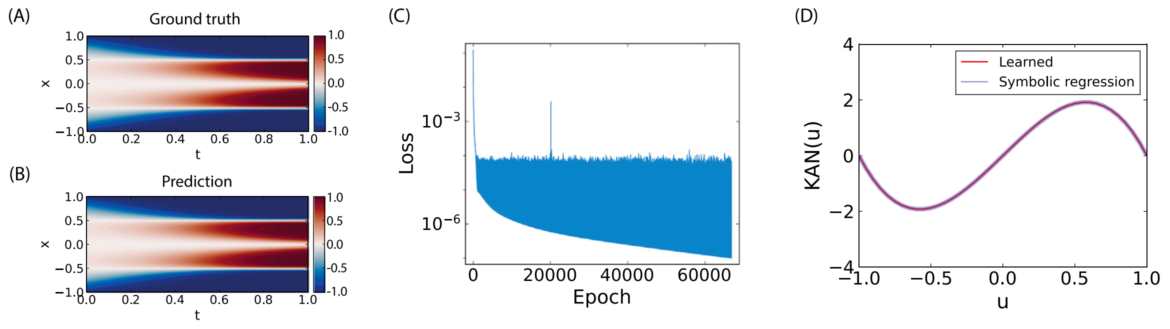

Fig. C.1. *Allen–Cahn equation*: (A) Solution field $u(x,t)$ of the ground truth. (B) Solution field $u(x,t)$ of the prediction with a trained KAN-ODE model. (C) Loss function evolution. (D) Learned hidden physics of the reaction source term in the Allen–Cahn equation, and its symbolic expression.

Table C.1
Seven candidate symbolic expressions of the reaction source term in the Allen–Cahn equation (Eq. (C.1)). The complexities of the symbolic equations and the corresponding losses are also tabulated.

Complexity	Loss	Equation
1	9.1219×10^{-1}	u
3	6.5775×10^{-1}	$1.8526u$
5	6.5775×10^{-1}	$u/0.5398 + 8.3105 \times 10^{-4}$
7	4.2106×10^{-1}	$(2.4821 - u^2)u$
9	1.1523×10^{-5}	$(5.0015 - 5.0021u^2)u$
11	1.0832×10^{-5}	$5.0021(0.9999 - u^2)u$
13	1.0493×10^{-5}	$5.0021u(0.9999 - u(u + 3.7283 \times 10^{-4}))$

The computational domain was defined as $x \in [-1,1]$ and $t \in [0,1]$. The initial and boundary conditions were set as Eqs. (C.2)–(C.3).

$$u(x,0) = x^2 \cos(\pi x), \tag{C.2}$$

$$u(-1,t) = u(1,t), \qquad \frac{\partial u}{\partial x}(-1,t) = \frac{\partial u}{\partial x}(1,t). \tag{C.3}$$

Eq. (C.1) was discretized with $\Delta x = 0.05$ using a central difference scheme. Then, the discretized PDE was solved by the `Tsit5` ODE integrator.

C.1. Modeling hidden physics in PDEs

To model the unknown source term in Eq. (C.1), we formulated the KAN-ODE such that

$$\frac{\partial u}{\partial t} = 0.0001\frac{\partial^2 u}{\partial x^2} + \mathrm{KAN}(u,\theta). \tag{C.4}$$

Similarly to the *Fisher–KPP* example, Eq. (C.4) was discretized using a central difference scheme and converted to an ODE system using the Method of Lines. We constructed the KAN model with a single layer comprising a single node, [1,1,10], thereby encoding just one activation function.

The KAN-ODE model was trained on the data shown in Fig. C.1(A). In this example, we applied an early stop when the error reached 10^{-7}. The prediction with the learned model is illustrated in Fig. C.1(B), showing accurate reconstruction of the solution field $u(x,t)$ after ~65,000 updates (Fig. C.1(C)). This resultant single activation function is shown in Fig. C.1(D). For the modeled activation function, a symbolic expression was derived, using the SymbolicRegression.jl package [45]. Given the candidate symbolic expressions of $[+,-,\times,/]$, it returned the expressions tabulated in Table C.1. Among them, the following was the simplest expression of the low-loss group (i.e. the group of expressions that jumped from 10^{-1} to 10^{-5} loss),

$$\mathrm{KAN}(u) = (5.0015 - 5.0021u^2)u. \tag{C.5}$$

Eq. (C.5) is remarkably close to the exact source term of $5u - 5u^3$. Note that all the symbolic expressions with errors on the order of 10^{-5} in Table C.1 have an identical form with slightly different coefficients. This example provides another demonstration of accurate and human-interpretable modeling of hidden physics from measurable quantities via KAN-ODEs.

C.2. Data-driven solutions of PDEs

We further demonstrate KAN-ODEs as surrogate models using the Allen–Cahn equation by replacing the entire right-hand side of Eq. (C.1) with a KAN. To mimic sparse data measurements, only five profiles at the selected times $t \in \{0.1, 0.3, 0.5, 0.7, 0.9\}$

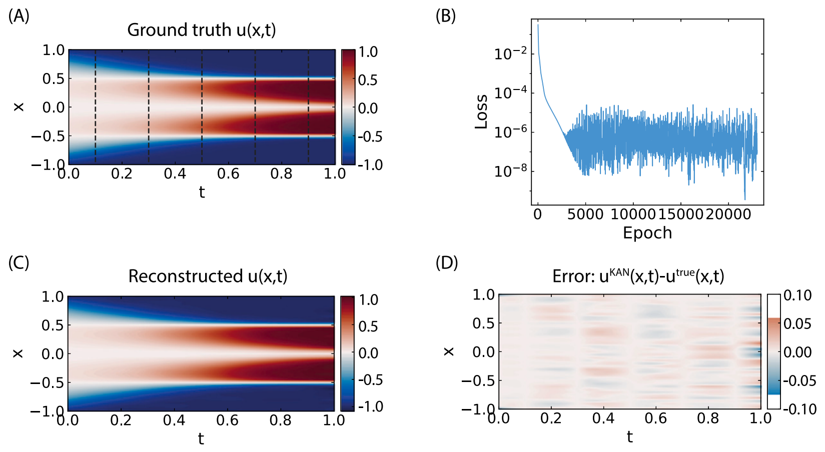

Fig. C.2. *Allen–Cahn equation*: Demonstration of KAN-ODEs as a surrogate model. (A) Ground truth $u(x,t)$. The dashed lines indicate the training data at t = 0.1, 0.3, 0.5, 0.7, 0.9. (B) MSE loss during training. (C) Reconstructed $u(x,t)$ by the trained KAN-ODEs. (D) The error between the reconstructed $u^{\text{KAN}}(x,t)$ and the ground truth $u^{\text{true}}(x,t)$.

were collected from the ground truth simulation and used as training data (Fig. C.2(A)), while testing involved reconstruction of the entire time history. The KAN-ODEs as a surrogate model were constructed using 2 layers with 10 hidden nodes ([41,10,10], [10,41,10]). Eq. (22) was trained to learn the dynamics of the phase transition as shown in the loss plot (Fig. C.2(B)). The resulting field reconstruction of $u(x,t)$ is illustrated in Fig. C.2(C), which qualitatively matches well with the ground truth. The spatio-temporal error between the prediction via KAN-ODEs and the ground truth ($u^{\text{KAN}}(x,t) - u^{\text{true}}(x,t)$) in Fig. C.2(D) depicts near-zero error at and surrounding each training time slice, and small but noticeable error in the large windows between time steps, especially near the end of the simulation time. For example, the error is not appreciable at time $t = 0.1$, but begins to grow before shrinking back to near-zero by $t = 0.3$ when the second observation takes place. While the errors introduced here are fairly minor, this result implies that the KAN-ODEs need more training data in order to extrapolate quantitatively well to times significantly outside of their training data window, like $t = 1.5$ s or $t = 2.0$ s. Within the 1 s range considered here, however, these results highlight the strong capability of KAN-ODEs to accurately model and reconstruct physical phenomena between training samples with very sparse measurements and *without any prior physical knowledge* included.

References

[1] G.E. Karniadakis, I.G. Kevrekidis, L. Lu, P. Perdikaris, S. Wang, L. Yang, Physics-informed machine learning, Nat. Rev. Phys. 3 (6) (2021) 422–440, http://dx.doi.org/10.1038/s42254-021-00314-5.
[2] S.L. Brunton, J.L. Proctor, J.N. Kutz, Discovering governing equations from data by sparse identification of nonlinear dynamical systems, Proc. Natl. Acad. Sci. 113 (15) (2016) 3932–3937, http://dx.doi.org/10.1073/pnas.1517384113.
[3] S.L. Brunton, J.N. Kutz, Promising directions of machine learning for partial differential equations, Nat. Comput. Sci. (2024) 1–12, http://dx.doi.org/10.1038/s43588-024-00643-2.
[4] S.H. Rudy, S.L. Brunton, J.L. Proctor, J.N. Kutz, Data-driven discovery of partial differential equations, Sci. Adv. 3 (4) (2017) e1602614, http://dx.doi.org/10.1126/sciadv.1602614.
[5] S.L. Brunton, J.L. Proctor, J.N. Kutz, Discovering governing equations from data by sparse identification of nonlinear dynamical systems, Proc. Natl. Acad. Sci. 113 (15) (2016) 3932–3937, http://dx.doi.org/10.1073/pnas.1517384113.
[6] M. Raissi, P. Perdikaris, G.E. Karniadakis, Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations, J. Comput. Phys. 378 (2019) 686–707, http://dx.doi.org/10.1016/j.jcp.2018.10.045.
[7] R.T.Q. Chen, Y. Rubanova, J. Bettencourt, D. Duvenaud, Neural ordinary differential equations, 2019, http://dx.doi.org/10.48550/arXiv.1806.07366, arXiv:1806.07366.
[8] S. Kim, W. Ji, S. Deng, Y. Ma, C. Rackauckas, Stiff neural ordinary differential equations, Chaos 31 (9) (2021) 093122, http://dx.doi.org/10.1063/5.0060697.
[9] R. Dandekar, K. Chung, V. Dixit, M. Tarek, A. Garcia-Valadez, K.V. Vemula, C. Rackauckas, Bayesian neural ordinary differential equations, 2022, http://dx.doi.org/10.48550/arXiv.2012.07244, arXiv:2012.07244.
[10] Y. Lu, A. Zhong, Q. Li, B. Dong, Beyond finite layer neural networks: Bridging deep architectures and numerical differential equations, in: International Conference on Machine Learning, PMLR, 2018, pp. 3276–3285.
[11] E. Haber, L. Ruthotto, Stable architectures for deep neural networks, Inverse Probl. 34 (1) (2017) 014004, http://dx.doi.org/10.1088/1361-6420/aa9a90.
[12] W. Ji, W. Qiu, Z. Shi, S. Pan, S. Deng, Stiff-PINN: Physics-informed neural network for stiff chemical kinetics, J. Phys. Chem. A 125 (36) (2021) 8098–8106, http://dx.doi.org/10.1021/acs.jpca.1c05102.
[13] S. Cuomo, V.S. Di Cola, F. Giampaolo, G. Rozza, M. Raissi, F. Piccialli, Scientific machine learning through physics–informed neural networks: Where we are and what's next, J. Sci. Comput. 92 (3) (2022) 88, http://dx.doi.org/10.1007/s10915-022-01939-z.
[14] N. Geneva, N. Zabaras, Modeling the dynamics of PDE systems with physics-constrained deep auto-regressive networks, J. Comput. Phys. 403 (2020) 109056, http://dx.doi.org/10.1016/j.jcp.2019.109056.
[15] P. Ren, C. Rao, Y. Liu, J.-X. Wang, H. Sun, PhyCRNet: Physics-informed convolutional-recurrent network for solving spatiotemporal PDEs, Comput. Methods Appl. Mech. Engrg. 389 (2022) 114399, http://dx.doi.org/10.1016/j.cma.2021.114399.

[16] L. Lu, R. Pestourie, W. Yao, Z. Wang, F. Verdugo, S.G. Johnson, Physics-informed neural networks with hard constraints for inverse design, SIAM J. Sci. Comput. 43 (6) (2021) B1105–B1132, http://dx.doi.org/10.1137/21M1397908.

[17] S. Alkhadhr, M. Almekkawy, Wave equation modeling via physics-informed neural networks: Models of soft and hard constraints for initial and boundary conditions, Sensors 23 (5) (2023) 2792, http://dx.doi.org/10.3390/s23052792.

[18] Z. Lai, C. Mylonas, S. Nagarajaiah, E. Chatzi, Structural identification with physics-informed neural ordinary differential equations, J. Sound Vib. 508 (2021) 116196, http://dx.doi.org/10.1016/j.jsv.2021.116196.

[19] W. Ji, S. Deng, Autonomous discovery of unknown reaction pathways from data by chemical reaction neural network, J. Phys. Chem. A 125 (4) (2021) 1082–1092, http://dx.doi.org/10.1021/acs.jpca.0c09316.

[20] W. Ji, F. Richter, M.J. Gollner, S. Deng, Autonomous kinetic modeling of biomass pyrolysis using chemical reaction neural networks, Combust. Flame 240 (2022) 111992, http://dx.doi.org/10.1016/j.combustflame.2022.111992.

[21] B.C. Koenig, P. Zhao, S. Deng, Accommodating physical reaction schemes in DSC cathode thermal stability analysis using chemical reaction neural networks, J. Power Sources 581 (2023) 233443, http://dx.doi.org/10.1016/j.jpowsour.2023.233443.

[22] Q. Li, H. Chen, B.C. Koenig, S. Deng, Bayesian chemical reaction neural network for autonomous kinetic uncertainty quantification, Phys. Chem. Chem. Phys. 25 (5) (2023) 3707–3717, http://dx.doi.org/10.1039/D2CP05083H.

[23] B.C. Koenig, H. Chen, Q. Li, P. Zhao, S. Deng, Uncertain lithium-ion cathode kinetic decomposition modeling via Bayesian chemical reaction neural networks, Proc. Combust. Inst. 40 (1) (2024) 105243, http://dx.doi.org/10.1016/j.proci.2024.105243.

[24] E. Kaiser, J.N. Kutz, S.L. Brunton, Sparse identification of nonlinear dynamics for model predictive control in the low-data limit, Proc. R. Soc. A 474 (2219) (2018) 20180335, http://dx.doi.org/10.1098/rspa.2018.0335.

[25] U. Fasel, J.N. Kutz, B.W. Brunton, S.L. Brunton, Ensemble-SINDy: Robust sparse model discovery in the low-data, high-noise limit, with active learning and control, Proc. R. Soc. A 478 (2260) (2022) 20210904, http://dx.doi.org/10.1098/rspa.2021.0904, arXiv:2111.10992.

[26] Z. Liu, Y. Wang, S. Vaidya, F. Ruehle, J. Halverson, M. Soljačić, T.Y. Hou, M. Tegmark, KAN: Kolmogorov-arnold networks, 2024, http://dx.doi.org/10.48550/arXiv.2404.19756, arXiv:2404.19756.

[27] C.J. Vaca-Rubio, L. Blanco, R. Pereira, M. Caus, Kolmogorov-arnold networks (KANs) for time series analysis, 2024, http://dx.doi.org/10.48550/arXiv.2405.08790, arXiv:2405.08790.

[28] K. Xu, L. Chen, S. Wang, Kolmogorov-arnold networks for time series: Bridging predictive power and interpretability, 2024, http://dx.doi.org/10.48550/arXiv.2406.02496, arXiv:2406.02496.

[29] K. Hornik, M. Stinchcombe, H. White, Multilayer feedforward networks are universal approximators, Neural Netw. 2 (5) (1989) 359–366, http://dx.doi.org/10.1016/0893-6080(89)90020-8.

[30] A.N. Kolmogorov, On the representation of continuous functions of several variables as superpositions of continuous functions of a smaller number of variables., Dokl. Akad. Nauk (108(2)) (1956).

[31] Z. Li, Kolmogorov-arnold networks are radial basis function networks, 2024, http://dx.doi.org/10.48550/arXiv.2405.06721, arXiv:2405.06721.

[32] V. Puri, KolmogorovArnold.jl, 2024, GitHub repository. Retrieved from https://github.com/vpuri3/KolmogorovArnold.jl.

[33] P. Ramachandran, B. Zoph, Q.V. Le, Searching for activation functions, 2017, http://dx.doi.org/10.48550/arXiv.1710.05941, arXiv:1710.05941.

[34] S. Kim, S. Deng, Inference of chemical kinetics and thermodynamic properties from constant-volume combustion of energetic materials, Chem. Eng. J. 469 (2023) 143779, http://dx.doi.org/10.1016/j.cej.2023.143779.

[35] C. Rackauckas, Y. Ma, J. Martensen, C. Warner, K. Zubov, R. Supekar, D. Skinner, A. Ramadhan, A. Edelman, Universal differential equations for scientific machine learning, 2021, http://dx.doi.org/10.48550/arXiv.2001.04385, arXiv:2001.04385.

[36] T. Maly, L.R. Petzold, Numerical methods and software for sensitivity analysis of differential-algebraic systems, Appl. Numer. Math. 20 (1–2) (1996) 57–79, http://dx.doi.org/10.1016/0168-9274(95)00117-4.

[37] Y. Cao, S. Li, L. Petzold, Adjoint sensitivity analysis for differential-algebraic equations: algorithms and software, J. Comput. Appl. Math. 149 (1) (2002) 171–191, http://dx.doi.org/10.1016/S0377-0427(02)00528-9.

[38] Y. Cao, S. Li, L. Petzold, R. Serban, Adjoint sensitivity analysis for differential-algebraic equations: The adjoint DAE system and its numerical solution, SIAM J. Sci. Comput. 24 (3) (2003) 1076–1089, http://dx.doi.org/10.1137/S1064827501380630.

[39] C. Rackauckas, Q. Nie, DifferentialEquations.jl – A performant and feature-rich ecosystem for solving differential equations in Julia, J. Open Res. Softw. 5 (1) (2017) http://dx.doi.org/10.5334/jors.151.

[40] A. Pal, Lux: Explicit parameterization of deep neural networks in Julia, 2023, http://dx.doi.org/10.5281/zenodo.7808904.

[41] C. Tsitouras, Runge–Kutta pairs of order 5(4) satisfying only the first column simplifying assumption, Comput. Math. Appl. 62 (2) (2011) 770–775, http://dx.doi.org/10.1016/j.camwa.2011.06.002.

[42] D.P. Kingma, J. Ba, Adam: A method for stochastic optimization, 2017, http://dx.doi.org/10.48550/arXiv.1412.6980, arXiv:1412.6980.

[43] E.J. Michaud, Z. Liu, M. Tegmark, Precision machine learning, Entropy 25 (1) (2023) 175, http://dx.doi.org/10.3390/e25010175.

[44] Blealtan, Efficient-kan, 2024, URL https://github.com/Blealtan/efficient-kan.

[45] M. Cranmer, Interpretable machine learning for science with PySR and SymbolicRegression.jl, 2023, http://dx.doi.org/10.48550/arXiv.2305.01582, arXiv:2305.01582.

[46] X. Liu, T. Xiao, S. Si, Q. Cao, S. Kumar, C.-J. Hsieh, Neural SDE: Stabilizing Neural ODE Networks with Stochastic Noise, 2019, http://dx.doi.org/10.48550/arXiv.1906.02355, arXiv:1906.02355.

[47] X. Ying, An overview of overfitting and its solutions, J. Phys. Conf. Ser. 1168 (2) (2019) 022022, URL http://dx.doi.org/10.1088/1742-6596/1168/2/022022.

[48] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, R. Salakhutdinov, Dropout: A simple way to prevent neural networks from overfitting, J. Mach. Learn. Res. 15 (56) (2014) 1929–1958.

[49] G. Di Marzo, RODAS5 (4)-Méthodes De Rosenbrock D'ordre 5 (4) Adaptées Aux Problemes Différentiels-Algébriques MSc Mathematics Thesis, 1993.