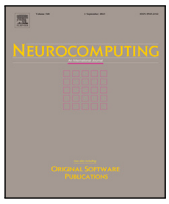
Neurocomputing 637 (2025) 130122

Contents lists available at ScienceDirect

Neurocomputing

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



Survey Paper

Integrating neural networks with numerical schemes for dynamical systems: A review

Jinsong Tang [a,b], Yunjin Tong [c], Lihua Chen [a], Shengze Cai [d], Shiying Xiong [a,*]

[a] Department of Engineering Mechanics, School of Aeronautics and Astronautics, Zhejiang University, Hangzhou, Zhejiang, 310027, China
[b] Huanjiang Laboratory, Zhejiang University, Zhuji, Zhejiang, 311899, China
[c] Stanford Graduate School of Business, Stanford University, Stanford, CA, 94305, USA
[d] College of Control Science and Engineering, Zhejiang University, Hangzhou, Zhejiang, 311899, China

ARTICLE INFO

Communicated by S. Das

Keywords:
Computational dynamics
Numerical schemes
Machine learning
Neural networks

ABSTRACT

As scientific discovery becomes increasingly data-driven, integrating physics-based numerical methods with advanced machine learning (ML) techniques has brought new insight in the analysis of complex physical systems. This paper explores how this integrated approach overcomes the limitations of traditional first-principle methods and brute-force ML techniques to achieve a more precise solution to complex physical problems. Specifically, we review networks that combine classical numerical schemes with neural networks applied to various physical systems. These integrated methods with residual structures effectively adhere to system symmetries and conservation laws. This integration outperforms conventional data-driven techniques in robustness and predictive capability, even with smaller datasets, owing to its improved ability to capture complex physical patterns.

1. Introduction

The integration of physics-based numerical methods with machine learning (ML) represents a promising approach to studying complex physical systems, particularly as data-driven methodologies play an increasingly prominent role in scientific discovery. This review discusses hybrid frameworks that combine numerical schemes with neural networks, aiming to address some limitations of first-principle methods and purely data-driven ML. By preserving symmetries and conservation laws, these methods show potential for delivering accurate and robust solutions, even with limited datasets.

The origins of these paradigms can be traced back to the foundations of scientific inquiry. Since Newton's era, two main paradigms have guided scientific research: the Keplerian (data-driven) approach and the Newtonian (first-principle-based) approach [1]. The first-principle-based approach, while fundamental and elegant, often faces practical limitations due to complex equations and high computational costs. In contrast, the data-driven approach has become highly effective, especially with advancements in statistical techniques and ML. This approach manages high-dimensional functions through statistical analysis of their structures. By integrating learning paradigms with simulation frameworks, these methods significantly enhance the modeling of ordinary differential equations (ODEs) and partial differential equations (PDEs).

Despite their broad applicability, data-driven methods such as black box neural networks face several challenges. These approaches often require large, well-clean datasets and depend on intricate structures that can be highly sensitive to variations in input [2,3]. Furthermore, brute-force ML with tools like deep neural networks struggles with high-dimensional input–output spaces, costly data acquisition, physically implausible results, and poor extrapolation robustness. These factors complicate the accurate prediction of long-term dynamical behaviors.

To address these challenges, integrating physical information with ML provides an effective approach for developing data-driven computational mechanics [4]. The key innovation lies in embedding physics-based priors into learning algorithms, preserving the system's physical structure. Hence, these models outperform state-of-the-art data-driven methods in accuracy, robustness, and capability, even with smaller datasets and shorter training periods. This approach also extends beyond physics. For instance, Udriste et al. [5,6] employed optimization theory in dynamic modeling to explore controllability in nonholonomic macroeconomic systems and multitime optimal growth models. These approaches could be enhanced by physics-informed machine learning techniques to more effectively capture the underlying complex dynamics.

We highlight methods that integrate numerical schemes for ODEs and PDEs into data flows, aligning them with traditional dynamical

* Corresponding author.
E-mail address: shiying.xiong@zju.edu.cn (S. Xiong).

https://doi.org/10.1016/j.neucom.2025.130122
Received 21 August 2024; Received in revised form 13 January 2025; Accepted 23 March 2025
Available online 2 April 2025

system approaches. Significant advancements include the development of residual networks (ResNet) [7] that combine neural networks with discrete Euler integrators for ODEs, and the introduction of Neural ODEs [8] that utilize continuous data flow mechanisms. This progression from ResNet to Neural ODEs represents a shift from discrete to continuous dynamics, enhancing the modeling of ODEs. By employing customized solving schemes, these neural network data flows can be tailored to specific physical and mechanical contexts. The neural networks discussed in this paper are built on these concepts, incorporating targeted modifications to improve expressive power for particular physical applications. We present representative work from solid mechanics, Hamiltonian mechanics, and fluid dynamics.

In solid mechanics, neural networks are used as surrogate models to replace complex physical models. For example, Ma et al. developed neural networks for surface contact collisions [9], and these networks have proven effective in modeling nonlinear materials [10,11]. Zhang et al. developed hierarchical deep-learning neural networks (HiDeNN) using FEM representations, where network weights and biases are dependent on node positions [12–14]. Huang et al. proposed a problem-independent machine learning (PIML) technique to mitigate FEM's computational cost in structural topology optimization, mapping shape functions to material densities within the extended multi-scale FEM (EMsFEM) framework [15,16].

Hamiltonian mechanics, expressed in symplectic spaces, extends beyond classical mechanics to continuous systems like electromagnetic fields and fluids. Greydanus et al. introduced Hamiltonian neural networks (HNNs) to preserve Hamiltonian energy by modifying the loss function [17]. Building on this, methods such as SRNN [18] and SSINN [19] embed symplectic integrators into recurrent neural networks to solve separable Hamiltonian systems and address large-scale N-body problems [20,21]. Jin et al. developed SympNet for handling both separable and nonseparable Hamiltonian systems, though it faces scalability issues with high-dimensional problems [22]. Additionally, Hamiltonian-based neural networks have been adapted for broader applications: Toth et al. created the Hamiltonian generative network (HGN) for inferring dynamics from high-dimensional data [23], and Zhong et al. introduced symplectic ODE-Net (SymODEN) to incorporate external control in Hamiltonian systems [24].

In fluid mechanics, research is increasingly integrating physical priors into ML to ground neural networks in physical laws rather than relying solely on data [25–28]. Raissi et al. introduced physics-informed neural networks (PINNs) to incorporate principles such as symmetry and conservation into the learning process [29,30]. PINNs have been shown to capture irregular PDE solutions without regularization [31] and can model high-speed flows by integrating the Euler equations into the loss function [32]. Embedding PDE structures into neural networks aids in understanding complex fluid dynamics [33–35], though challenges remain due to high dimensionality and limited data. For high Reynolds number flows, traditional turbulence models are still necessary, but data-driven approaches are advancing turbulence prediction [36–39]. Bin et al. found that progressive ML models are compatible with traditional turbulence modeling [40].

To better illustrate the mathematical derivation, architecture, and effectiveness of this approach, we will examine five specific examples: data-driven numerical integration networks (DDNI) [41], symplectic Taylor neural networks (Taylor-nets) [42], nonseparable symplectic neural networks (NSSNNs) [43], Roe neural networks (RoeNet) [44], and the neural vortex method (NVM) [45]. Our analysis highlights two main advantages of integrating numerical schemes with neural networks:

1. **Preserve symmetries:** integrate classical numerical schemes and neural networks to maintain symmetries and conservation laws, ensuring more accurate and efficient solutions.
2. **Enhance expressive power:** use higher-order residual structures to model complex physical phenomena, such as sound waves and vortices, capturing intricate patterns more effectively.

The outline of this paper is as follows. Section 2 covers ResNet and neural ODEs, illustrating their extension to complex physical systems for developing new data-driven numerical methods. Section 3 reviews fundamental principles of various mechanical systems. Section 4 describes the network design. Section 5 summarizes key implementation details and experimental findings. Finally, Section 6 discusses the numerical results, method limitations, and potential future research directions.

2. Residual networks and neural ordinary differential equations

In conventional deep neural networks, the output $\boldsymbol{y}$ of a block is generally represented as:

$$\boldsymbol{y} = \mathcal{F}(\boldsymbol{x}, \boldsymbol{\theta}), \tag{1}$$

where $\boldsymbol{x}$ denotes the input, $\boldsymbol{\theta}$ refers to the parameters of the neural network, and $\mathcal{F}(\boldsymbol{x}, \boldsymbol{\theta})$ represents the function learned by the block. This function typically includes components such as convolutional layers, activation functions, and batch normalization parameters, which encompass weights and biases.

Introduced by He et al. [7], ResNet employs residual learning to effectively train very deep networks and address the challenges associated with deep architectures. The key concept of ResNet is to focus on learning the residual function rather than the entire mapping. Specifically, the network is designed to learn:

$$\boldsymbol{y} = \mathcal{F}(\boldsymbol{x}, \boldsymbol{\theta}) + \boldsymbol{x}. \tag{2}$$

In contrast to the standard approach represented by (1), the formulation in (2) includes a shortcut connection $\boldsymbol{x}$ that directly links the input to the output.

ResNet represents a notable advancement in deep neural network architecture by incorporating residual connections to aid in the learning of complex functions. This design improves training efficiency and performance, particularly for very deep networks, thereby facilitating the effective training of models with hundreds or even thousands of layers and addressing the degradation problem associated with deep architectures. ResNet has achieved state-of-the-art results in various tasks, including image classification and object detection.

Neural ODEs [8] extend the principles of residual learning to continuous dynamics by modeling data transformations as a continuous process governed by an ODE. Specifically, the evolution of the hidden state $\boldsymbol{h}(t)$ over time t is described by:

$$\frac{\mathrm{d}\boldsymbol{h}(t)}{\mathrm{d}t} = \mathcal{F}(\boldsymbol{h}, t, \boldsymbol{\theta}), \tag{3}$$

where $\mathcal{F}$ is a neural network that specifies the rate of change, and $\boldsymbol{\theta}$ denotes its parameters. This approach offers a flexible and memory-efficient alternative to traditional discrete deep networks.

The concepts underlying Neural ODEs offer several opportunities for further development. One potential avenue is improving time integration solvers by utilizing symplectic integrators rather than conventional Euler or RK methods. Additionally, integrating graph neural networks (GNNs) could enhance the modeling of nonlinear interactions between computational units, which would support the Lagrangian description of multi-body systems. Finally, the application of spatial differential operators could provide a framework for describing the spatiotemporal evolution of continuous media governed by PDEs.

Neural ODEs can be extended to the following generalized form:

$$\frac{\partial \boldsymbol{h}(\boldsymbol{x}, t)}{\partial t} = \mathcal{F}(\boldsymbol{h}, \boldsymbol{x}, t, \boldsymbol{\theta}). \tag{4}$$

The network also takes additional inputs $\boldsymbol{x}$, such as spatial coordinates, external responses, and the coupling of multiphysics fields. Additionally, the process of solving Eq. (4) can be customized using specific numerical schemes. By extending the dynamic formulation in (4) and applying specialized solving techniques – such as symplectic schemes,

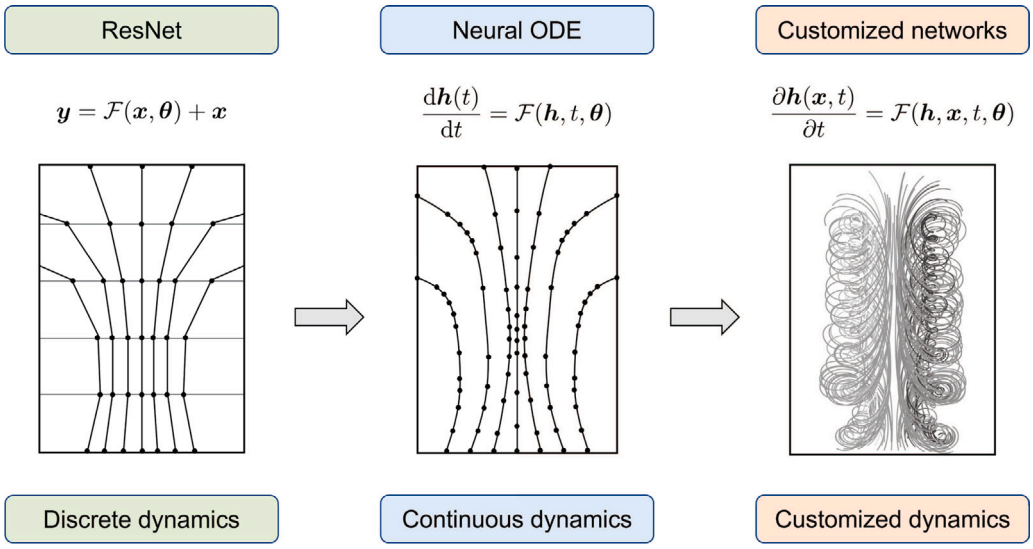


Fig. 1. Methods from ResNet to Neural ODEs to our customized networks for ODE modeling from discrete to continuous to customized dynamics.

Table 1
Overview of mathematical background and numerical solvers.

Dynamics systems	Physical priors	Solvers
Flexible multibody systems	Floating reference frame	Runge–Kutta integrator
Hamiltonian systems	Symplectic structure	Symplectic integrator
Hyperbolic PDEs	Hyperbolic conservation law	Roe solver
Incompressible flows	Helmholtz's theorem	Lagrangian vortex method

vorticity-preserving schemes, and shock-capturing methods – the data flow can be enhanced with specific structural properties.

Fig. 1 illustrates the central theme of our review. The transition from ResNet to Neural ODEs highlights a shift from discrete to continuous dynamics, which enhances the modeling of ODEs. This approach, along with the application of tailored solving schemes, illustrates how neural network data flows can be adapted to physical and mechanical contexts.

3. Dynamics equations

We illustrate the integration of mathematical principles derived from physical problems into numerical schemes through a series of examples. Specifically, the numerical algorithms for the relevant dynamical systems are provided in Appendix. This demonstration highlights how incorporating such mathematical priors can enhance the effectiveness and accuracy of numerical solutions. Table 1 provides a summary of the mathematical foundations and numerical solvers discussed in this section, providing an overview of the key concepts and methodologies employed.

3.1. Rigid–flexible coupling dynamics

We use a floating reference frame to analyze the motion of flexible structures [46–48] and model the rigid–flexible coupling dynamics in flexible multibody systems as follows:

$$\boldsymbol{M}\ddot{\boldsymbol{X}} + \boldsymbol{K}\boldsymbol{X} + \boldsymbol{C}_{\boldsymbol{X}}^{\mathrm{T}}\boldsymbol{\eta} = \boldsymbol{Q}^{l} - \boldsymbol{Q}^{v}, \tag{5}$$

where $\boldsymbol{C}(\boldsymbol{X}, t) = \boldsymbol{0}$ represents the kinematic constraint and $(\boldsymbol{X}_0, \dot{\boldsymbol{X}}_0)$ specifies the initial conditions, $\dot{\boldsymbol{X}}$ and $\ddot{\boldsymbol{X}}$ denote the first and second time derivatives of $\boldsymbol{X}$, respectively. The matrix $\boldsymbol{C}_{\boldsymbol{X}}$ is the Jacobian of the constraint function $\boldsymbol{C}(\boldsymbol{X}, t)$, $\boldsymbol{K}$ is the stiffness matrix, and $\boldsymbol{\eta}$ is the Lagrange multiplier. The mass matrix $\boldsymbol{M}$ and load vectors $\boldsymbol{Q}^{l}$ and $\boldsymbol{Q}^{v}$ are computed as follows:

$$\begin{aligned} \boldsymbol{M} &= \int_{\Omega} \rho \boldsymbol{L}^{\mathrm{T}}(\boldsymbol{X})\boldsymbol{L}(\boldsymbol{X})\,\mathrm{d}\Omega, \quad \boldsymbol{Q}^{l} = \int_{\Omega} \rho \boldsymbol{L}(\boldsymbol{X})^{\mathrm{T}}\boldsymbol{b}\,\mathrm{d}\Omega, \\ \boldsymbol{Q}^{v} &= \int_{\Omega} \rho \boldsymbol{L}^{\mathrm{T}}(\boldsymbol{X})\boldsymbol{a}(\boldsymbol{X}, \dot{\boldsymbol{X}})\,\mathrm{d}\Omega, \end{aligned} \tag{6}$$

where ρ represents the material density, Ω defines the spatial domain of the structure, $\boldsymbol{L}$ characterizes the kinematic properties, $\boldsymbol{b}$ corresponds to the body forces, and $\boldsymbol{a}$ accounts for the Coriolis and centripetal forces.

3.2. Hamiltonian dynamics

A Hamiltonian system is characterized by N pairs of canonical coordinates, which consist of generalized positions $\boldsymbol{q}$ and generalized momenta $\boldsymbol{p}$. The dynamics of these coordinates over time are governed by Hamilton's equations:

$$\frac{\mathrm{d}\boldsymbol{q}}{\mathrm{d}t} = \frac{\partial \mathcal{H}}{\partial \boldsymbol{p}}, \quad \frac{\mathrm{d}\boldsymbol{p}}{\mathrm{d}t} = -\frac{\partial \mathcal{H}}{\partial \boldsymbol{q}}. \tag{7}$$

Here, $\boldsymbol{q} = (q_1, q_2, \ldots, q_N)$ represents the generalized positions, $\boldsymbol{p} = (p_1, p_2, \ldots, p_N)$ denotes the generalized momenta, and $\mathcal{H}(\boldsymbol{q}, \boldsymbol{p})$ is the Hamiltonian function.

3.3. Hyperbolic conservation laws

In fluid mechanics, conservation laws such as those for mass, momentum, and energy are described by first-order quasilinear hyperbolic PDEs:

$$\frac{\partial \boldsymbol{u}}{\partial t} + \frac{\partial \boldsymbol{F}(\boldsymbol{u})}{\partial \boldsymbol{x}} = \boldsymbol{0}, \tag{8}$$

along with appropriate initial and boundary conditions. Here, $\boldsymbol{u}$ is an N_c-component vector representing the conserved quantities, $t \in [t_0, t_1]$ denotes time, $\boldsymbol{x}$ refers to the spatial coordinates within the domain Ω, and $\boldsymbol{F}$ is the N_c-component flux function.

3.4. Vortex dynamics

For an incompressible fluid, the Navier–Stokes (NS) equations can be rewritten in terms of the vorticity field $\boldsymbol{\omega} = \boldsymbol{\nabla} \times \boldsymbol{u}$ as [49]

$$\frac{\mathrm{D}\boldsymbol{\omega}}{\mathrm{D}t} = (\boldsymbol{\omega} \cdot \boldsymbol{\nabla})\boldsymbol{u} + \nu \boldsymbol{\nabla}^2 \boldsymbol{\omega} + \boldsymbol{\nabla} \times \boldsymbol{f}, \quad \nabla^2 \boldsymbol{\Psi} = -\boldsymbol{\omega}, \tag{9}$$

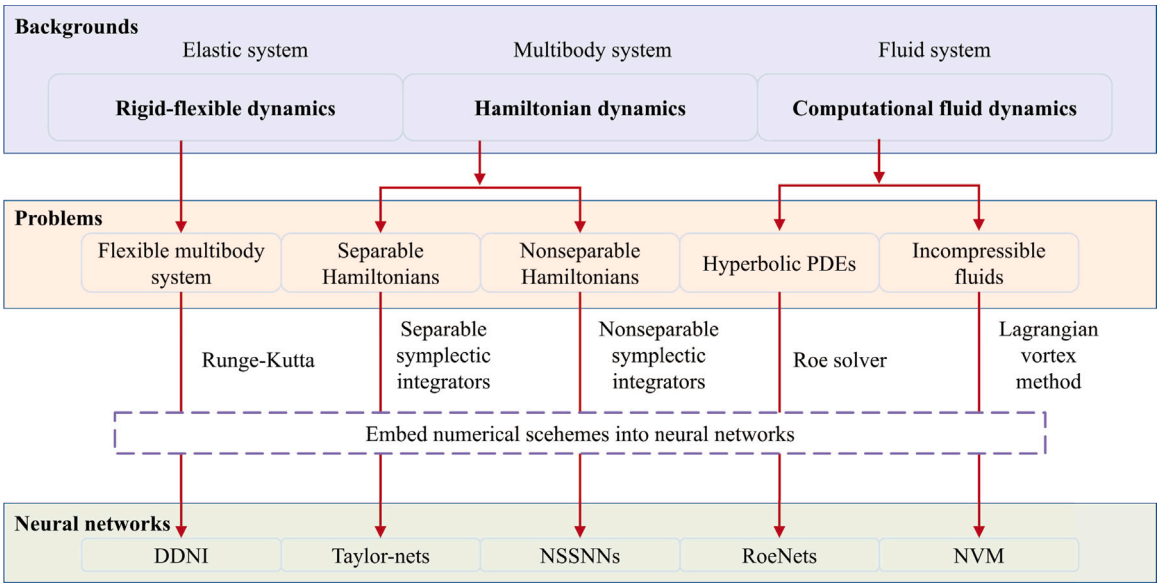


Fig. 2. Schematic diagrams of the integration of numerical schemes with neural networks for solving and predicting dynamic problems.

where $\boldsymbol{f}$ is the body force and $\boldsymbol{\Psi}$ is a vector potential such that $\boldsymbol{u} = \boldsymbol{\nabla} \times \boldsymbol{\Psi}$. This formulation highlights the vorticity dynamics and, in the inviscid limit ($\nu = 0$), describes the motion of vortex lines and surfaces, as outlined by Helmholtz's theorems [50–52].

4. Neural networks based on numerical schemes

We illustrate the development of neural networks that incorporate numerical schemes. Through various examples, we demonstrate how integrating numerical methods with neural networks can enhance their capability to solve and predict complex dynamic problems. Fig. 2 depicts this integration, showcasing how numerical schemes are embedded within neural network architectures to effectively handle and predict dynamic behaviors.

4.1. Data-driven numerical integration networks

For flexible structures, the time integration of (A.2) involves finite element discretization and Gaussian numerical integration (GNI), both of which are computationally intensive. The DDNI method [41] uses deep neural networks with generalized coordinates $\boldsymbol{X}$ and velocities $\dot{\boldsymbol{X}}$ to update dynamic quantities via the relations in (6),

$$\boldsymbol{M}_{\theta_1}(\boldsymbol{X}) \to \boldsymbol{M}, \boldsymbol{Q}^l_{\theta_2}(\boldsymbol{X}) \to \boldsymbol{Q}^l, \boldsymbol{Q}^v_{\theta_3}(\boldsymbol{X}, \dot{\boldsymbol{X}}) \to \boldsymbol{Q}^v, \tag{10}$$

where the parameters $(\theta_1, \theta_2, \theta_3)$ are optimized by minimizing the prediction error between DDNI and GNI. DDNI reduces computational complexity by exploiting mass matrix symmetry and mapping physical degrees of freedom (DOFs) to modal space, thereby simplifying the neural network and facilitating the storage of time-independent quantities, such as the stiffness matrix $\boldsymbol{K}$.

Given the load, initial conditions, and constraints, RK4 is used for time integration. Dynamic quantities $\boldsymbol{M}$, $\boldsymbol{Q}^l$, and $\boldsymbol{Q}^v$ are directly predicted from state variables $\boldsymbol{X}$ and $\dot{\boldsymbol{X}}$ at each time step, advancing the dynamics without traversing numerous elements, thus significantly enhancing efficiency.

4.2. Symplectic neural networks

Symplectic Taylor neural networks. In the separable Hamiltonian problem governed by (A.6), Taylor-nets [42] predict system evolution using generalized coordinates $\boldsymbol{q}$ and momenta $\boldsymbol{p}$ as state variables. These networks learn the gradients of the Hamiltonian with respect to these coordinates via symmetric networks $\boldsymbol{T}_p$ and $\boldsymbol{V}_q$:

$$\boldsymbol{T}_p(\boldsymbol{p}, \boldsymbol{\theta}_p) \to \frac{\partial T(\boldsymbol{p})}{\partial \boldsymbol{p}}, \quad \boldsymbol{V}_q(\boldsymbol{q}, \boldsymbol{\theta}_q) \to \frac{\partial V(\boldsymbol{q})}{\partial \boldsymbol{q}}, \tag{11}$$

where $(\boldsymbol{\theta}_p, \boldsymbol{\theta}_q)$ are parameters trained to model the right-hand side of (A.6). Taylor-nets utilize symmetric nonlinear terms similar to those in a Taylor polynomial, combined linearly. These networks, $\boldsymbol{T}_p(\boldsymbol{p}, \boldsymbol{\theta}_p)$ and $\boldsymbol{V}_q(\boldsymbol{q}, \boldsymbol{\theta}_q)$, are defined as:

$$\begin{cases} \boldsymbol{T}_p(\boldsymbol{p}, \boldsymbol{\theta}_p) = \sum_{i=1}^{M} (\boldsymbol{A}_i^T \circ f_i \circ \boldsymbol{A}_i - \boldsymbol{B}_i^T \circ f_i \circ \boldsymbol{B}_i) \circ \boldsymbol{p} + \boldsymbol{b}, \\ \boldsymbol{V}_q(\boldsymbol{q}, \boldsymbol{\theta}_q) = \sum_{i=1}^{M} (\boldsymbol{C}_i^T \circ f_i \circ \boldsymbol{C}_i - \boldsymbol{D}_i^T \circ f_i \circ \boldsymbol{D}_i) \circ \boldsymbol{q} + \boldsymbol{d}, \end{cases} \tag{12}$$

where '$\circ$' denotes function composition, $\boldsymbol{A}_i$ and $\boldsymbol{B}_i$ are fully connected layers of size $N_h \times N$, $\boldsymbol{b}$ is a bias vector of dimension N, and M is the number of terms in the Taylor series. $\boldsymbol{T}_p(\boldsymbol{p}, \boldsymbol{\theta}_p)$ takes $\boldsymbol{p}$ as input with parameters $\boldsymbol{\theta}_p = (\boldsymbol{A}_i, \boldsymbol{B}_i, \boldsymbol{b})$. Each negative term $\boldsymbol{B}_i^T \circ f_i \circ \boldsymbol{B}_i$ complements a positive term $\boldsymbol{A}_i^T \circ f_i \circ \boldsymbol{A}_i$, allowing representation of any symmetric matrix. The function f_i represents the ith order term in the Taylor series, defined as $f_i(x) = x^i/i!$. $\boldsymbol{V}_q(\boldsymbol{q}, \boldsymbol{\theta}_q)$ follows a similar structure. During training, these networks minimize the loss between their separable symplectic integration and the ground truth.

Nonseparable symplectic neural networks. NSSNNs [43] are extended to nonseparable Hamiltonian mechanics for the prediction of system evolution. They learn dynamics through an augmented system (A.8), enabling extraction of the energy function $\mathcal{H}(\boldsymbol{q}, \boldsymbol{p})$ via neural network $\mathcal{H}_\theta(\boldsymbol{q}, \boldsymbol{p})$ trained with parameters θ and computation of its gradient $\nabla \mathcal{H}_\theta(\boldsymbol{q}, \boldsymbol{p})$. The input layer of the integrator starts with $(\boldsymbol{q}, \boldsymbol{p}, \boldsymbol{x}, \boldsymbol{y}) = (\boldsymbol{q}_0, \boldsymbol{p}_0, \boldsymbol{q}_0, \boldsymbol{p}_0)$ at $t = t_0$, and the output layer is $(\boldsymbol{q}, \boldsymbol{p}, \boldsymbol{x}, \boldsymbol{y}) = (\boldsymbol{q}_n, \boldsymbol{p}_n, \boldsymbol{x}_n, \boldsymbol{y}_n)$ at $t = t_0 + n\mathrm{d}t$. Furthermore, since $\boldsymbol{x}$ and $\boldsymbol{y}$ theoretically represent $\boldsymbol{q}$ and $\boldsymbol{p}$ in (A.12), the dataset can be constructed with variables $(\boldsymbol{q}, \boldsymbol{p}, \boldsymbol{x}, \boldsymbol{y})$ derived from $(\boldsymbol{q}, \boldsymbol{p})$. The network $\mathcal{H}_\theta$ ensures that $\boldsymbol{\phi}_1^\delta$, $\boldsymbol{\phi}_2^\delta$, and $\boldsymbol{\phi}_3^\delta$ in (A.12) maintain the system's symplectic structure. This property guarantees that constructing the network preserves the Hamiltonian flow's symplecticity.

4.3. Roe neural networks

RoeNet [44] learns the weak solution of (8) without a prescribed $\boldsymbol{F}$, using a neural network based on the Roe solver. It takes $\boldsymbol{u}$ as input

state, predict matrices $\boldsymbol{L}_\theta \to \boldsymbol{L}$ and $\boldsymbol{\Lambda}_\phi \to \boldsymbol{\Lambda}$, and facilitate the Roe solver. Substituting $\boldsymbol{L}_\theta$ and $\boldsymbol{\Lambda}_\phi$ into (A.17) and (A.18) yields

$$\begin{aligned}\boldsymbol{u}_j^{n+1} =& \boldsymbol{u}_j^n - \frac{1}{2}\lambda_r(\boldsymbol{L}^n_{j+\frac{1}{2},\theta})^+(\boldsymbol{\Lambda}^n_{j+\frac{1}{2},\phi} - |\boldsymbol{\Lambda}^n_{j+\frac{1}{2},\phi}|)\boldsymbol{L}^n_{j+\frac{1}{2},\theta}(\boldsymbol{u}^n_{j+1} - \boldsymbol{u}^n_j) \\ & - \frac{1}{2}\lambda_r(\boldsymbol{L}^n_{j-\frac{1}{2},\theta})^+(\boldsymbol{\Lambda}^n_{j-\frac{1}{2},\phi} + |\boldsymbol{\Lambda}^n_{j-\frac{1}{2},\phi}|)\boldsymbol{L}^n_{j-\frac{1}{2},\theta}(\boldsymbol{u}^n_j - \boldsymbol{u}^n_{j-1}),\end{aligned} \tag{13}$$

with

$$\boldsymbol{L}^n_{j+\frac{1}{2},\theta} = \boldsymbol{L}_\theta(\boldsymbol{u}^n_j, \boldsymbol{u}^n_{j+1}), \qquad \boldsymbol{\Lambda}^n_{j+\frac{1}{2},\phi} = \boldsymbol{\Lambda}_\phi(\boldsymbol{u}^n_j, \boldsymbol{u}^n_{j+1}). \tag{14}$$

In RoeNet, Eq. (13) governs the evolution of the system's states from $\boldsymbol{u}_j^n$ to $\boldsymbol{u}_j^{n+1}$. In practice, $\boldsymbol{L}_\theta$ and $\boldsymbol{\Lambda}_\phi$ utilize ResBlocks [7], ending with linear layers sized $N_h \times N_c$ and N_h, respectively. $\boldsymbol{\Lambda}_\phi$'s parameters create a diagonal matrix with N_h entries. RoeNet efficiently predicts solutions for hyperbolic conservation laws, even with discontinuous behavior, using limited discontinuity information from short training windows.

4.4. Neural vortex method

The NVM [45] employs Eulerian representations of the flow field to reconstruct the underlying fluid dynamics using neural networks. Integration of two networks with a vorticity-to-velocity Poisson solver enables high-resolution extraction of Eulerian flow from Lagrangian inductive priors. This approach addresses the difficulty of directly interpreting velocity and pressure fields from high-dimensional observations such as images. The detection network takes a vorticity field as input, which is then split into two branches. One branch predicts the probability of vortex presence using convolution, while the other locates the exact positions of vortices. During training, the network penalizes incorrect position detections only when it fails to detect a vortex in cells as per the ground truth from DNS. This approach mirrors real-time object detection methodologies discussed in Redmon et al. (2016) [53]. Additionally, the focal loss [54] is applied to mitigate issues related to imbalanced classification.

To predict vortex dynamics, two GNNs [21], $\boldsymbol{A}(\theta_1)$ and $\boldsymbol{A}(\theta_2)$, are employed. $\boldsymbol{A}(\theta_1)$ models induced velocities between vortices based on their positions and vorticity detected by the detection network. The output vector from $\boldsymbol{A}(\theta_1)$ characterizes the induced velocity of each vortex element j (where $j \neq i$) on vortex i. This approach sums up all induced velocities on vortex i to compute the total induced velocity from other vortices. On the other hand, $\boldsymbol{A}(\theta_2)$ predicts external force influences based on local vorticity and vortex positions.

5. Numerical results

In this section, we present the applications and impacts of advanced network architectures in computational mechanics, illustrated through several detailed cases. Our focus includes examining the effectiveness and efficiency of these novel approaches in solving complex problems.

For a more comprehensive understanding of the methodologies and outcomes discussed, please refer to the detailed studies and examples provided in the works of Tang et al. [41], Tong et al. [42], Xiong et al. [43], Tong et al. [44], and Xiong et al. [45]. These references offer in-depth analyses and additional context regarding the implementation and results of the discussed network architectures.

5.1. Data-driven numerical integration networks

We present a case study of a two-dimensional rotating beam analyzed with DDNI under harmonic, piecewise linear, and constant loads. Fig. 3 shows the beam's dynamic responses. Validation suggests several advantages of DDNI: it is approximately 15 times faster than GNI and demonstrates computational efficiency comparable to commercial software, even when accounting for training time [41]; it requires only a single training session and performs reliably across a range of operating conditions; and it achieves high accuracy with minimal trade-offs, as illustrated in Fig. 3(f).

5.2. Symplectic Taylor neural networks

We examine three Hamiltonian systems: Pendulum with $H(p,q) = p^2/2 - \cos q$, Lotka–Volterra with $H(p,q) = p - e^p + 2q - e^q$, and Hénon–Heiles with $H(p,q) = (p_1^2 + p_2^2)/2 + (q_1^2 + q_2^2)/2 + (q_1^2 q_2 - q_2^3/3)$. We use the Taylor-net to model and predict system behavior, comparing its performance to that of Neural ODE, as shown in Fig. 4. For each Hamiltonian system, we generate random initial conditions and perform short-term simulations with $\Delta t = 0.01$ using symplectic integration for training. The model then predicts the long-term behavior based on these initial conditions, demonstrating Taylor-net's robust predictive capability over extended timeframes. On the other hand, the Neural ODE relies on a basic Euler method for integration and does not incorporate the symplectic structure or domain-specific priors. As a result, the Neural ODE fails to preserve the system's inherent structure, causing errors to accumulate over time. This highlights the importance of incorporating structural information to prevent error growth and maintain accuracy in long-term predictions.

To evaluate the accuracy of the methods, we define the following error metric:

$$\text{Error} = \sqrt{\sum_{i=1}^{N_t}\left[(p_i^{\mathrm{P}} - p_i^{\mathrm{G}})^2 + (q_i^{\mathrm{P}} - q_i^{\mathrm{G}})^2\right]}, \tag{15}$$

where N_t denotes the total number of time steps for which predictions are made, and the superscripts P and G indicate the predicted values and the ground truth, respectively. This metric quantifies the overall discrepancy between the predicted and actual values, accounting for deviations in both p- and q-components over time.

As shown in Fig. 4, the Taylor-net model exhibits significantly higher accuracy, with prediction errors of 0.2298, 0.2652, and 0.5800 across the three systems. In contrast, the Neural ODE model performs poorly, with much larger errors of 13.7000, 39.8828, and 41.4106. These results underscore the limitations of the Neural ODE model in accurately capturing system dynamics.

The lower errors achieved by the Taylor-net model underscore its effectiveness in modeling complex temporal behaviors. Meanwhile, the Neural ODE model's high error rates suggest limitations in its ability to generalize, indicating that further refinement of its architecture or training process may be necessary.

5.3. Nonseparable symplectic neural networks

We showcase the superior performance of NSSNN by predicting the system $\mathcal{H} = 0.5(q^2 + 1)(p^2 + 1)$ [55] over the interval $t = 0$ to $t = 20000$ with initial conditions $(q_0, p_0) = (0, -3)$. The training data spans only $[0, 0.2]$. Fig. 5 compares predictions from various neural networks, all trained under identical conditions, to the ground truth. Neural ODEs exhibit larger errors due to their lack of embedded symplectic structure. While the other two networks, which focus on learning the Hamiltonian, achieve faster learning, HNN's stability is lower compared to NSSNN. NSSNN, with its nonseparable symplectic integrator, excels in long-term predictions. Similarly, the errors associated with the three methods – Neural ODE, HNN, and NSSNN – are computed using (15). The resulting values are 81.6601, 29.3517, and 6.7406, respectively.

In Fig. 6, we use the trained model to predict the dynamics of three aligned 2D vortices, where the evolution of the vortices is driven by the interaction of 6400 particles with vorticity. The results from NSSNN and HNN are compared with the ground truth [43]. The initial vorticity conditions are based on [56]. The key challenge is keeping the vortices separate rather than merging into larger structures. NSSNN preserves the separation of vortices as shown in the ground truth, whereas HNN leads to vortex merging. We remark that NSSNN is distinguished by several notable features. While HNN enforces the conservative properties of a Hamiltonian system through its loss function, it relies on temporal derivatives of momentum and position, which are challenging to obtain, and does not fully preserve the symplectic structure. In

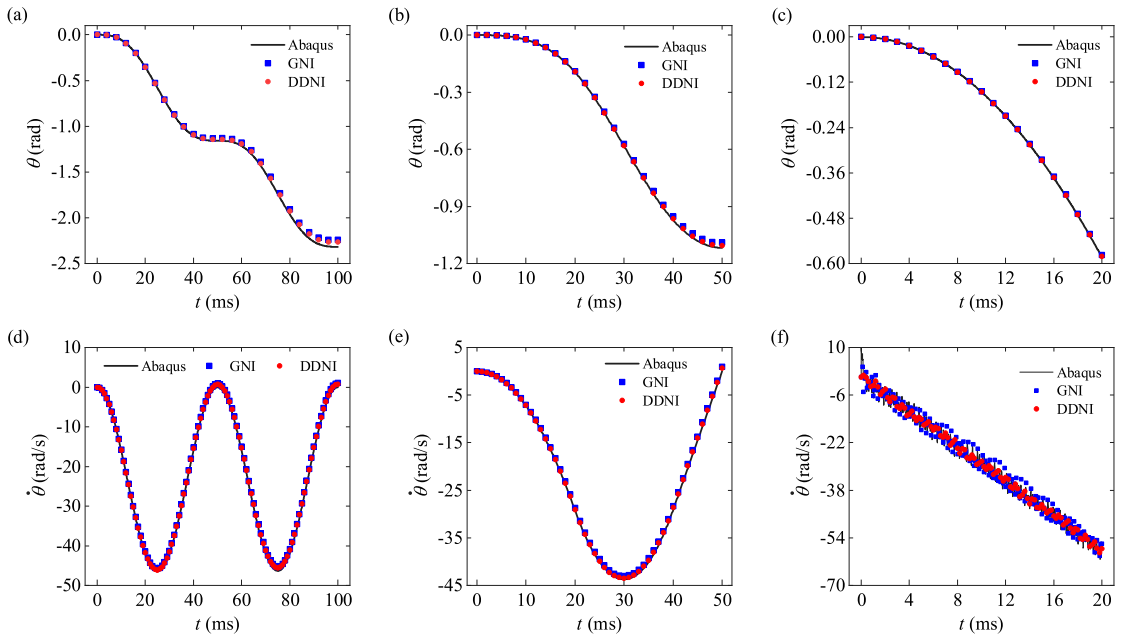


Fig. 3. (a)–(c) rotation angle and (d)–(f) angular velocity correspond to dynamic response results: simple harmonic, piecewise linear, and constant loads, respectively [41].

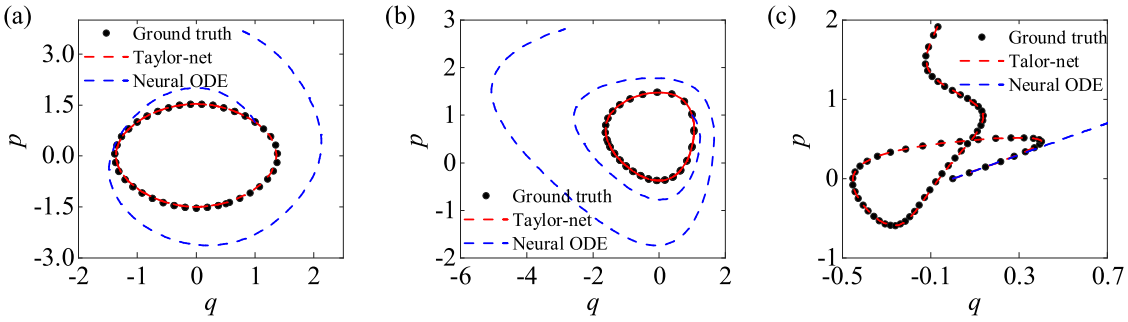


Fig. 4. The prediction for the dynamics of three Hamiltonian systems using Taylor-net: (a) Pendulum for $t = 4\pi$, (b) Lotka–Volterra for $t = 4\pi$, and (c) Hénon–Heiles for $t = 10$. Results are compared with those from the Neural ODE, which uses a simple Euler method without symplectic structure or domain-specific priors, causing error accumulation over time.

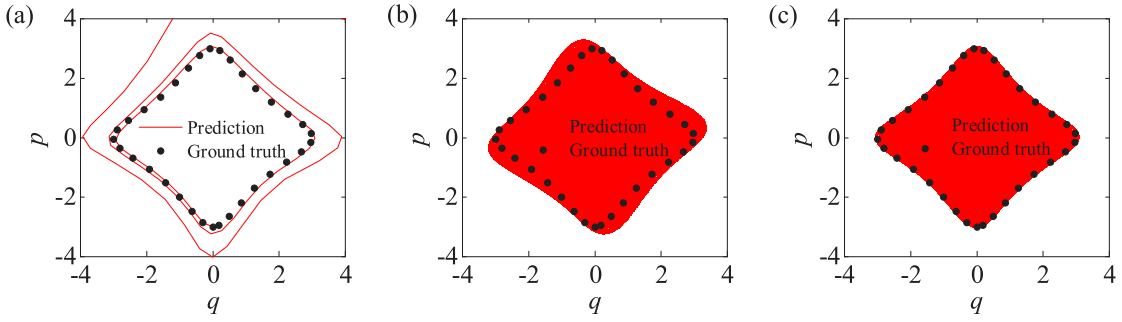


Fig. 5. Comparison of prediction results for the Lotka–Volterra system: (a) Neural ODE, (b) HNN, (c) NSSNN.

contrast, the Neural ODE-based NSSNN overcomes these limitations by incorporating an integrator into the network and embedding the Hamiltonian prior to predict continuous system trajectories, showing strong potential for a wide range of applications.

5.4. Roe neural networks

We first demonstrate RoeNet's ability to predict first-order linear hyperbolic PDEs of the form (8) with $\boldsymbol{F}(\boldsymbol{u}) = \boldsymbol{A}\boldsymbol{u}$, where $\boldsymbol{A}$ is a constant $N_c \times N_c$ matrix. Fig. 7 shows the prediction results for a single-component linear hyperbolic PDE with $\boldsymbol{F} = x$ and initial condition $u(t = 0, x) = e^{-300x^2}$. In Fig. 7(a), we compare the predictions of RoeNet, RoeNet with noisy training data, and the Roe solver against the exact solution at $t = 0.3$. RoeNet outperforms the numerical Roe solver, even with noise $\epsilon \sim \mathcal{N}(0, 0.1)$. At larger t, RoeNet's predictions remain accurate with or without noise, while the Roe solver's performance worsens, as shown in Fig. 7(b). Additionally, we evaluate the temporal evolution of computational errors for the RoeNet in comparison with traditional methods. Fig. 7(c) shows the average deviation $\lambda_u = \langle |u^{\mathrm{P}} - u^{\mathrm{G}}| \rangle$ of

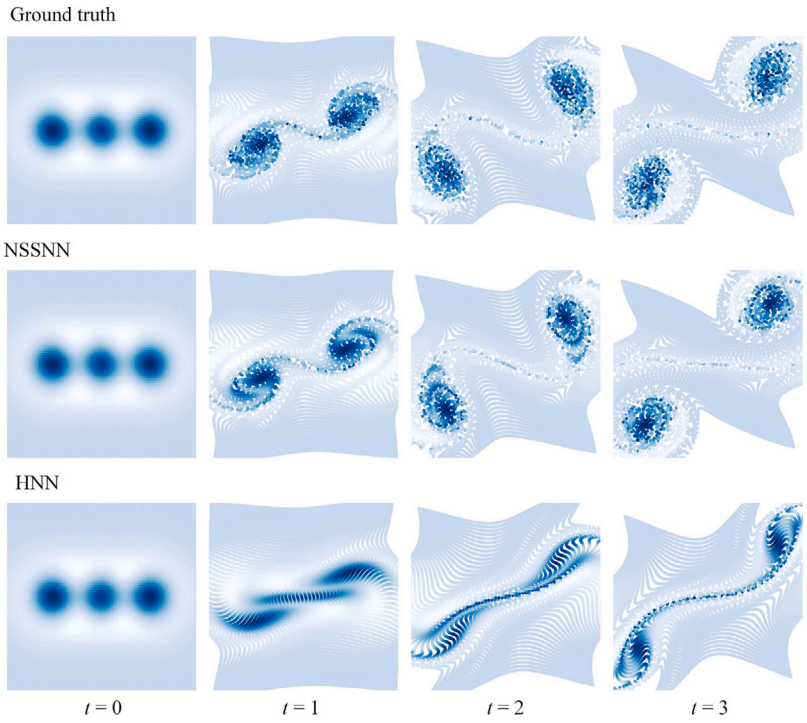


Fig. 6. Visualization of three aligned 2D vortices, where the vortex evolution is driven by the interaction of 6400 particles with vorticity. Results from NSSNN and HNN are compared with the ground truth.

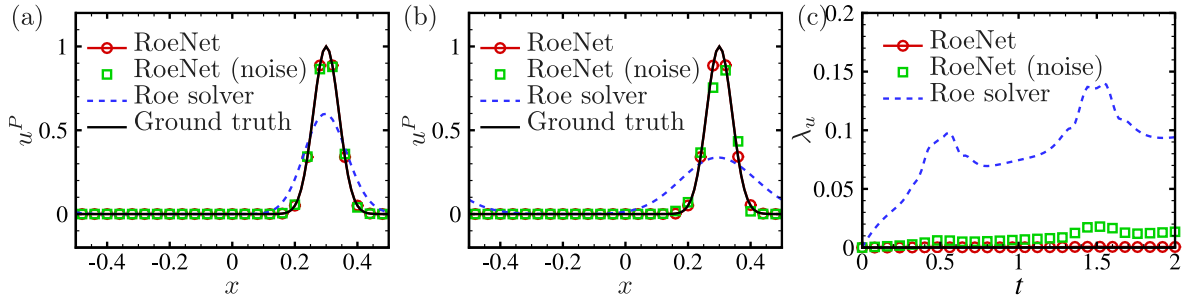


Fig. 7. Comparison of RoeNet and Roe solver for a one-component linear hyperbolic PDE: (a) $t = 0.3$, (b) $t = 1.3$, and (c) average deviation $\lambda_u = \langle |u^{\mathrm{P}} - u^{\mathrm{G}}| \rangle$. "RoeNet (noise)" indicates RoeNet with training noise $\epsilon \sim \mathcal{N}(0, 0.1)$.

the predicted solutions from the exact solution. RoeNet's deviation, indicated by the red circle line, is almost negligible, demonstrating its high accuracy. Even with noise, RoeNet's prediction error is more than ten times smaller than that of the numerical Roe solver, highlighting RoeNet's robustness.

We also apply RoeNet to solve a three-component linear hyperbolic PDE (8) with

$$\begin{cases} \boldsymbol{F} = \begin{bmatrix} 0.3237 & 2.705 & 5.4101 \\ 0.3597 & -0.4388 & -2.8777 \\ -0.0144 & 0.0576 & 1.1151 \end{bmatrix} \boldsymbol{x}, \\ \boldsymbol{u}(t=0, x \le 0) = (0.4, 0.4, 0.4), \quad \boldsymbol{u}(t=0, x > 0) = (-0.4, -0.4, -0.4). \end{cases} \tag{16}$$

Fig. 8 shows the exact solutions and the predicted results for the three components $u^{(1)}$, $u^{(2)}$, and $u^{(3)}$ of a Riemann problem with a linear flux function. The predictions by RoeNet match the exact solutions perfectly, while the Roe solver shows obvious errors around the discontinuities at $x \approx \pm 0.3$.

We show that RoeNet can predict long-term discontinuities of the inviscid Burgers' equation using only a short window of continuous training data. This equation, given by (8) with $F = \frac{1}{2}u^2$ and $u(t = 0, x) = \frac{1}{2} + \sin(2\pi x)$, is a fundamental PDE in multiple fields and can develop shock waves. In the absence of an analytical solution, Fig. 9 compares RoeNet's predictions with those of the Roe solver at $t = 0$, $t = 0.15$, and $t = 0.3$. The perfect match between RoeNet and the Roe solver demonstrates RoeNet's ability to predict future discontinuities from limited data, marking a significant breakthrough in predictive capabilities.

We remark that current neural network-based methods typically solve PDEs using datasets or predict continuous solutions. PINNs require knowledge of the governing PDE and periodic feedback, often relying on Hessian-based optimizers, which can increase training time [4]. In contrast, RoeNet only requires training data and employs gradient-based optimizers, leading to reduced computational costs and enhanced efficiency [44]. While conventional networks may struggle with predicting discontinuous solutions without a governing equation,

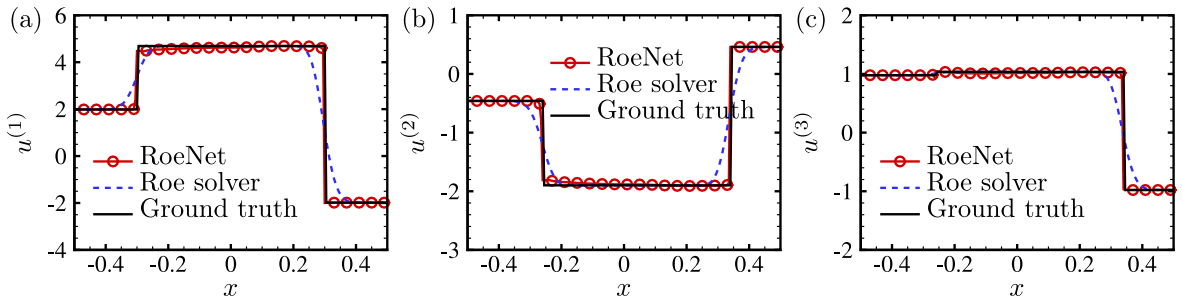


Fig. 8. Riemann problem with three components and a linear flux function. (a), (b), and (c) show the predictions by RoeNet and Roe solver, along with the exact solutions for the components $u^{(1)}$, $u^{(2)}$, and $u^{(3)}$, respectively.

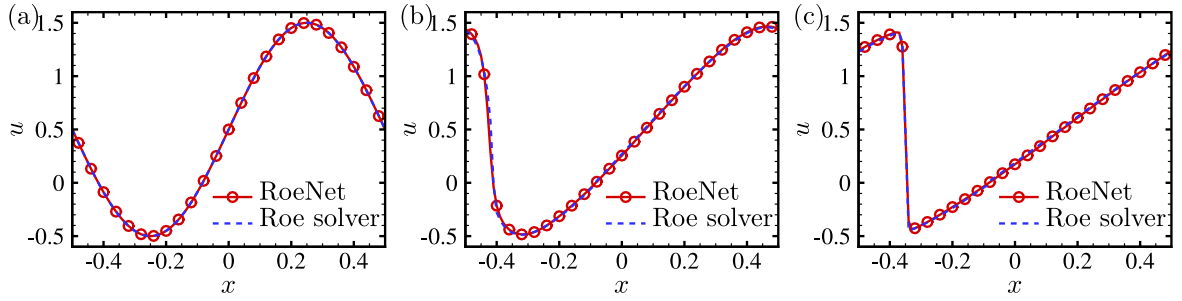


Fig. 9. Inviscid Burgers' equation with $u(t=0,x)=0.5+\sin(2\pi x)$ at (a) $t=0$, (b) $t=0.15$, and (c) $t=0.3$.

RoeNet shows improved performance in tasks where traditional methods encounter difficulties, especially for larger time values not included in the training data.

5.5. Neural vortex method

Figure 5 of [45] assesses the predictive capabilities of the NVM for the NS equations within a periodic domain, highlighting its effectiveness in capturing fluid dynamics in comparison to traditional LVM. The analysis focuses on two vortex particles, characterized by their predetermined initial positions and strengths. The results indicate that NVM's predictions align closely with those from DNS, whereas LVM exhibits significant errors in predicting vortex positions. Furthermore, NVM maintains a lower relative error over longer prediction intervals, highlighting its superior accuracy and reliability compared to traditional methods.

NVM also effectively predicts complex turbulence systems, as illustrated in Fig. 10, which shows two-dimensional Lagrangian scalar fields at $t=1$ with initial condition $\phi=x$ and resolution 2000^2. The fields evolve with $O(10)$ and $O(100)$ NVM vortex particles, each with random positions $\sim U(0,4)$ and strengths $\sim U(0,2)$ at the initial time, using the same trained model. By applying backward-particle-tracking to the NVM particle velocity fields, we solve the scalar field evolution equation and extract material structures. Fig. 10(a) shows clear spiral structures with fewer particles, while Fig. 10(b) displays turbulence with many particles. NVM demonstrates its potential to model complex turbulence with notable detail and efficiency, running on relatively modest hardware. For example, it was tested on a laptop with an Intel Core i7-9750H processor at 2.60 GHz, 16 GB of RAM, and an NVIDIA GeForce RTX 2060 GPU with 16 GB memory. This suggests that NVM can achieve reasonable accuracy without the need for advanced or specialized computing resources, making it a practical and accessible tool for turbulence modeling [45].

6. Conclusion

This manuscript offers a review of recent research where data-driven methods and computational mechanics intersect. It explores how combining physics-informed numerical schemes with advanced neural network architectures has led to notable improvements in predicting complex dynamical systems. By integrating these approaches, researchers have managed to maintain the physical structures, mathematical symmetries, and conservation principles of the systems. These approaches effectively reduce the reliance on the intensive numerical computations often required by traditional methods during prediction tasks. For instance, under the same computational conditions, DDNI achieves a computation time that is only 1/15th of what traditional methods require and surpasses some commercial software in efficiency [41]. Similarly, other network models demand five times as many samples as Taylor-net to achieve a comparable validation loss under identical conditions [42]. Additionally, while the Roe solver with 2000 grids provides lower accuracy than Roenet with only 100 grids, it incurs significantly higher computational costs [44]. These advancements demonstrate significant potential for enhancing accuracy, robustness, and efficiency, even when datasets are limited or training periods are constrained.

We review several mechanical paradigms that employ advanced algorithms with physics-informed priors to tackle nonlinear dynamical challenges. These include approaches like GNI for rigid–flexible coupling dynamics, symplectic structures for Hamiltonian systems, a Roe solver for hyperbolic PDEs, and LVM for incompressible fluid dynamics. By incorporating physics-based priors, these methods aim to narrow the solution space and reduce computational demands, while also improving prediction reliability. Integrating these structures into neural networks enhance physical interpretability and can better capture complex physical patterns, potentially leading to improvements in both accuracy and applicability.

These models have some limitations. Neural networks with embedded integrators often need longer training periods compared to those trained with datasets that include explicit time derivatives. Besides, explicit time-stepping schemes usually require small time steps to ensure accuracy. While this enhances discretization, it can also raise training costs and risk gradient explosion. Moreover, these methods are tailored for particular physical problems, which can limit their broader applicability.

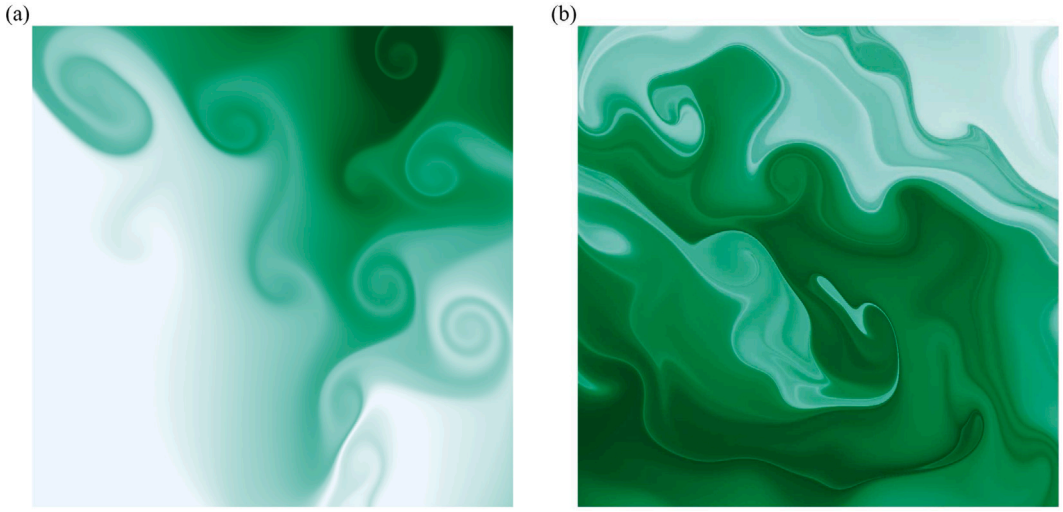


Fig. 10. Two-dimensional Lagrangian scalar fields at $t = 1$ with initial condition $\phi = x$ and resolution 2000^2. The evolution is driven by (a) $O(10)$ and (b) $O(100)$ random NVM vortex particles [45].

Future research in this field is expected to address several key areas. First, exploring implicit schemes, such as recurrent neural network (RNN) structures [18,57], may offer potential benefits in terms of stability and efficiency. Furthermore, many current models are end-to-end systems that do not account for environmental variability; hence, integrating online learning techniques to enhance adaptability in varying conditions is an area of growing interest. Another important focus will be the development of scalable methods that can generalize across a range of PDEs, with the aim of establishing a more versatile and broadly applicable framework. Additionally, applying these models to practical engineering challenges, particularly in learning essential quantities of dynamical systems such as the Hamiltonian function, could prove valuable, given their significance in real-world engineering contexts despite often being unknown.

CRediT authorship contribution statement

Jinsong Tang: Writing – original draft, Visualization, Methodology, Formal analysis. **Yunjin Tong:** Writing – review & editing, Formal analysis. **Lihua Chen:** Writing – review & editing, Formal analysis. **Shengze Cai:** Writing – review & editing, Validation. **Shiying Xiong:** Writing – review & editing, Supervision, Project administration, Methodology, Funding acquisition, Conceptualization.

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.

Acknowledgments

The authors acknowledge the support from the National Natural Science Foundation of China (Grants No. 12302294 and 12432010).

Appendix. Numerical schemes

We present several numerical schemes for solving the dynamics equations outlined in Section 3. These schemes not only provide solutions to the equations but also serve as templates for designing neural networks. By incorporating these established methods, the neural networks are able to learn and replicate the dynamics more effectively. This approach ensures that the networks maintain the stability and accuracy of the original schemes, while also offering greater flexibility and adaptability in addressing complex problems.

RK method. To solve Eq. (5) numerically, it is often rewritten in an augmented form:

$$\begin{bmatrix} \boldsymbol{M} & \boldsymbol{C}_{\boldsymbol{X}} \\ \boldsymbol{C}_{\boldsymbol{X}}^{\mathrm{T}} & \boldsymbol{0} \end{bmatrix} \begin{bmatrix} \ddot{\boldsymbol{X}} \\ \boldsymbol{\eta} \end{bmatrix} = \begin{bmatrix} \boldsymbol{Q}^{l} - \boldsymbol{Q}^{v} - \boldsymbol{K}\boldsymbol{X} \\ \boldsymbol{Q}^{c} \end{bmatrix}. \tag{A.1}$$

In this formulation, $\boldsymbol{Q}^{c}$ represents the constraint force [48]. By defining the vector $\boldsymbol{y} = [\boldsymbol{y}_1^{\mathrm{T}}, \boldsymbol{y}_2^{\mathrm{T}}, \boldsymbol{y}_3^{\mathrm{T}}]^{\mathrm{T}}$ with $\boldsymbol{y}_1 = \dot{\boldsymbol{X}}$, $\boldsymbol{y}_2 = \boldsymbol{\eta}$, and $\boldsymbol{y}_3 = \boldsymbol{X}$, Eq. (A.1) can be expressed in the general form of an ODE:

$$\frac{\mathrm{d}\boldsymbol{y}}{\mathrm{d}t} = \underbrace{\left\{ \begin{bmatrix} \boldsymbol{M} & \boldsymbol{C}_{\boldsymbol{X}} \\ \boldsymbol{C}_{\boldsymbol{X}}^{\mathrm{T}} & \boldsymbol{0} \end{bmatrix}^{-1} \begin{bmatrix} \boldsymbol{Q}^{l} - \boldsymbol{Q}^{v} - \boldsymbol{K}\boldsymbol{X} \\ \boldsymbol{Q}^{c} \end{bmatrix} \right\}}_{\boldsymbol{y}_1} \triangleq \boldsymbol{f}(t, \boldsymbol{y}), \tag{A.2}$$

with the initial condition $\boldsymbol{y}(t_0) = [\dot{\boldsymbol{X}}_0^{\mathrm{T}}, \boldsymbol{0}^{\mathrm{T}}, \boldsymbol{X}_0^{\mathrm{T}}]^{\mathrm{T}}$.

The first-order Runge–Kutta (RK) method, commonly known as the Euler method, updates the state vector $\boldsymbol{y}$ by advancing in the direction of $\boldsymbol{f}(t, \boldsymbol{y})$ over a timestep $\mathrm{d}t$. The updated state at $t + \mathrm{d}t$ is given by:

$$\boldsymbol{y}(t + \mathrm{d}t) = \boldsymbol{y}(t) + \boldsymbol{f}(t, \boldsymbol{y}) \cdot \mathrm{d}t. \tag{A.3}$$

While it is straightforward and effective for simple problems, it may not provide sufficient accuracy for problems requiring high precision over extended periods.

The fourth-order RK method, or RK4, calculates the next state by averaging the derivatives evaluated at multiple points within the timestep $\mathrm{d}t$. The updated state at $t + \mathrm{d}t$ is given by:

$$\boldsymbol{y}(t + \mathrm{d}t) = \boldsymbol{y}(t) + \frac{\mathrm{d}t}{6}(\boldsymbol{k}_1 + 2\boldsymbol{k}_2 + 2\boldsymbol{k}_3 + \boldsymbol{k}_4), \tag{A.4}$$

where the increments $[\boldsymbol{k}_1, \boldsymbol{k}_2, \boldsymbol{k}_3, \boldsymbol{k}_4]$ are defined as:

$$\left[\boldsymbol{f}(t, \boldsymbol{y}), \boldsymbol{f}\left(t + \frac{\mathrm{d}t}{2}, \boldsymbol{y} + \frac{\boldsymbol{k}_1 \mathrm{d}t}{2}\right), \boldsymbol{f}\left(t + \frac{\mathrm{d}t}{2}, \boldsymbol{y} + \frac{\boldsymbol{k}_2 \mathrm{d}t}{2}\right), \boldsymbol{f}(t + \mathrm{d}t, \boldsymbol{y} + \boldsymbol{k}_3 \mathrm{d}t) \right]. \tag{A.5}$$

RK4 achieves a global error of $O(\mathrm{d}t^4)$, providing high accuracy that is suitable for a wide range of applications.

Symplectic integrator. For Hamiltonian systems described by (7), the traditional RK method can be refined to better preserve the geometric symmetries inherent to Hamiltonian ODEs. If the Hamiltonian system is separable, meaning that $\mathcal{H} = T(\boldsymbol{p}) + V(\boldsymbol{q})$, it can be expressed as:

$$\frac{\mathrm{d}\boldsymbol{q}}{\mathrm{d}t} = \frac{\partial T(\boldsymbol{p})}{\partial \boldsymbol{p}}, \quad \frac{\mathrm{d}\boldsymbol{p}}{\mathrm{d}t} = -\frac{\partial V(\boldsymbol{q})}{\partial \boldsymbol{q}}. \tag{A.6}$$

A fourth-order symplectic integrator updates the momentum $\boldsymbol{p}$ and position $\boldsymbol{q}$ over a time step $\mathrm{d}t$ as follows: the momentum is updated

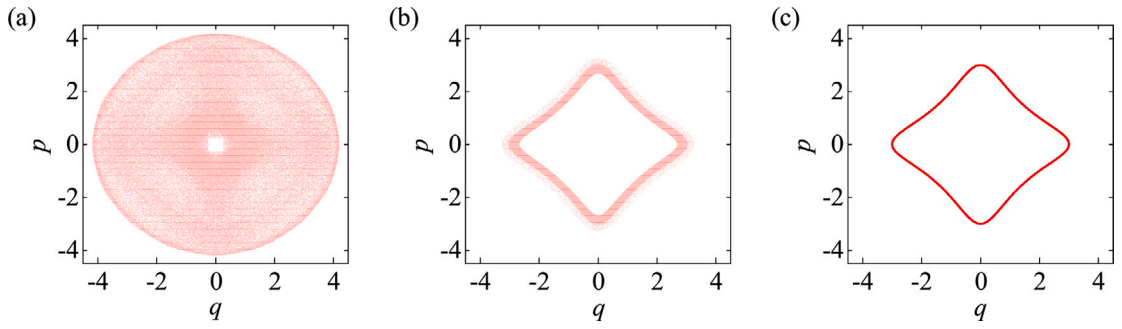


Fig. A.11. Numerical integration results using a symplectic integrator (A.11) for the system $\mathcal{H} = (q^2+1)(p^2+1)/2$ over $t = 0$ to $t = 20000$ for different ω values: (a) $\omega = 0$, (b) $\omega = 0.95$, and (c) $\omega = 100$.

by $\boldsymbol{p} - d_j \nabla V(\boldsymbol{q}) \cdot \mathrm{d}t$, and the position is updated by $\boldsymbol{q} + c_j \nabla T(\boldsymbol{p}) \cdot \mathrm{d}t$ for $j = 1, 2, 3, 4$. The coefficients c_j and d_j are selected to minimize lower-order error terms, ensuring fourth-order accuracy. These coefficients are given by [58]:

$$\begin{aligned} c_1 = c_4 &= \frac{1}{2(2-2^{1/3})}, \quad c_2 = c_3 = \frac{1-2^{1/3}}{2(2-2^{1/3})}, \\ d_1 = d_3 &= \frac{1}{2-2^{1/3}}, \quad d_2 = -\frac{2^{1/3}}{2-2^{1/3}}, \quad d_4 = 0. \end{aligned} \tag{A.7}$$

By applying these updates at each time step $\mathrm{d}t$, the system can be iteratively advanced from the initial state $(\boldsymbol{q}_0, \boldsymbol{p}_0)$ at time t_0 to the state $(\boldsymbol{q}_n, \boldsymbol{p}_n)$ at time $t_0 + n\mathrm{d}t$, where n denotes the number of time steps.

For a more general Hamiltonian system that is not separable, Tao [55] proposed a high-order, explicit, and symplectic time integrator. This method involves an augmented Hamiltonian defined as:

$$\overline{\mathcal{H}}(\boldsymbol{q}, \boldsymbol{p}, \boldsymbol{x}, \boldsymbol{y}) := \mathcal{H}_A + \mathcal{H}_B + \omega \mathcal{H}_C, \tag{A.8}$$

where the terms $\mathcal{H}_\Delta (\Delta = A, B, C)$ are defined as:

$$\mathcal{H}_A = \mathcal{H}(\boldsymbol{q}, \boldsymbol{y}), \quad \mathcal{H}_B = \mathcal{H}(\boldsymbol{x}, \boldsymbol{p}), \quad \mathcal{H}_C = \frac{1}{2}\left(\|\boldsymbol{q}-\boldsymbol{x}\|_2^2 + \|\boldsymbol{p}-\boldsymbol{y}\|_2^2\right), \tag{A.9}$$

and ω is a constant that controls the interaction between the original Hamiltonian system and an artificial constraint. The Hamiltonian equations for this augmented Hamiltonian are:

$$\begin{cases} \frac{\mathrm{d}\boldsymbol{q}}{\mathrm{d}t} = \frac{\partial \overline{\mathcal{H}}}{\partial \boldsymbol{p}} = \frac{\partial \mathcal{H}(\boldsymbol{x},\boldsymbol{p})}{\partial \boldsymbol{p}} + \omega(\boldsymbol{p}-\boldsymbol{y}), & \frac{\mathrm{d}\boldsymbol{p}}{\mathrm{d}t} = -\frac{\partial \overline{\mathcal{H}}}{\partial \boldsymbol{q}} = -\frac{\partial \mathcal{H}(\boldsymbol{q},\boldsymbol{y})}{\partial \boldsymbol{q}} - \omega(\boldsymbol{q}-\boldsymbol{x}), \\ \frac{\mathrm{d}\boldsymbol{x}}{\mathrm{d}t} = \frac{\partial \overline{\mathcal{H}}}{\partial \boldsymbol{y}} = \frac{\partial \mathcal{H}(\boldsymbol{q},\boldsymbol{y})}{\partial \boldsymbol{y}} - \omega(\boldsymbol{p}-\boldsymbol{y}), & \frac{\mathrm{d}\boldsymbol{y}}{\mathrm{d}t} = -\frac{\partial \overline{\mathcal{H}}}{\partial \boldsymbol{x}} = -\frac{\partial \mathcal{H}(\boldsymbol{x},\boldsymbol{p})}{\partial \boldsymbol{x}} + \omega(\boldsymbol{q}-\boldsymbol{x}), \end{cases} \tag{A.10}$$

with the initial condition $(\boldsymbol{q}, \boldsymbol{p}, \boldsymbol{x}, \boldsymbol{y})|_{t=t_0} = (\boldsymbol{q}_0, \boldsymbol{p}_0, \boldsymbol{q}_0, \boldsymbol{p}_0)$. These equations have the same solution as (7) in the sense that $(\boldsymbol{q}, \boldsymbol{p}, \boldsymbol{x}, \boldsymbol{y}) = (\boldsymbol{q}, \boldsymbol{p}, \boldsymbol{q}, \boldsymbol{p})$. The coefficient ω acts as a regularizer, thereby enhancing the stability of the numerical results.

High-order symplectic integrators can be constructed for $\overline{\mathcal{H}}$ with explicit updates. Specifically, $(\boldsymbol{q}, \boldsymbol{p}, \boldsymbol{x}, \boldsymbol{y})$ can be updated as follows:

$$(\boldsymbol{q}_i, \boldsymbol{p}_i, \boldsymbol{x}_i, \boldsymbol{y}_i) = \boldsymbol{\phi}_1^{\mathrm{d}t/2} \circ \boldsymbol{\phi}_2^{\mathrm{d}t/2} \circ \boldsymbol{\phi}_3^{\mathrm{d}t} \circ \boldsymbol{\phi}_2^{\mathrm{d}t/2} \circ \boldsymbol{\phi}_1^{\mathrm{d}t/2} \circ (\boldsymbol{q}_{i-1}, \boldsymbol{p}_{i-1}, \boldsymbol{x}_{i-1}, \boldsymbol{y}_{i-1}), \tag{A.11}$$

where $\boldsymbol{\phi}_1^\delta(\boldsymbol{q}, \boldsymbol{p}, \boldsymbol{x}, \boldsymbol{y})$, $\boldsymbol{\phi}_2^\delta(\boldsymbol{q}, \boldsymbol{p}, \boldsymbol{x}, \boldsymbol{y})$, and $\boldsymbol{\phi}_3^\delta(\boldsymbol{q}, \boldsymbol{p}, \boldsymbol{x}, \boldsymbol{y})$ are defined as:

$$\begin{bmatrix} \boldsymbol{q} \\ \boldsymbol{p} - \delta[\partial \mathcal{H}(\boldsymbol{q},\boldsymbol{y})/\partial \boldsymbol{q}] \\ \boldsymbol{x} + \delta[\partial \mathcal{H}(\boldsymbol{q},\boldsymbol{y})/\partial \boldsymbol{y}] \\ \boldsymbol{y} \end{bmatrix}, \begin{bmatrix} \boldsymbol{q} + \delta[\partial \mathcal{H}(\boldsymbol{x},\boldsymbol{p})/\partial \boldsymbol{p}] \\ \boldsymbol{p} \\ \boldsymbol{x} \\ \boldsymbol{y} - \delta[\partial \mathcal{H}(\boldsymbol{x},\boldsymbol{p})/\partial \boldsymbol{x}] \end{bmatrix}, \text{ and } \frac{1}{2}\begin{bmatrix} \begin{pmatrix} \boldsymbol{q}+\boldsymbol{x} \\ \boldsymbol{p}+\boldsymbol{y} \end{pmatrix} + \boldsymbol{R}^\delta \begin{pmatrix} \boldsymbol{q}-\boldsymbol{x} \\ \boldsymbol{p}-\boldsymbol{y} \end{pmatrix} \\ \begin{pmatrix} \boldsymbol{q}+\boldsymbol{x} \\ \boldsymbol{p}+\boldsymbol{y} \end{pmatrix} - \boldsymbol{R}^\delta \begin{pmatrix} \boldsymbol{q}-\boldsymbol{x} \\ \boldsymbol{p}-\boldsymbol{y} \end{pmatrix} \end{bmatrix}, \tag{A.12}$$

respectively. Here

$$\boldsymbol{R}^\delta := \begin{bmatrix} \cos(2\omega\delta)\boldsymbol{I} & \sin(2\omega\delta)\boldsymbol{I} \\ -\sin(2\omega\delta)\boldsymbol{I} & \cos(2\omega\delta)\boldsymbol{I} \end{bmatrix}, \tag{A.13}$$

where $\boldsymbol{I}$ is the identity matrix. We remark that $\boldsymbol{x}$ and $\boldsymbol{y}$ are auxiliary variables theoretically equal to $\boldsymbol{q}$ and $\boldsymbol{p}$.

Even if $\mathcal{H}(\boldsymbol{q}, \boldsymbol{p})$ is integrable, $\mathcal{H}_A + \mathcal{H}_B$ in the extended phase space $(\boldsymbol{q}, \boldsymbol{p}, \boldsymbol{x}, \boldsymbol{y})$ may not be integrable without binding ($\omega = 0$). As ω increases, the phase space for $\overline{\mathcal{H}}$ becomes more regular [59]. Fig. A.11 shows trajectories from $[-3, 0, -3, 0]$ using a second-order symplectic integrator for the Hamiltonian $\mathcal{H}(q, p) = \frac{1}{2}(q^2+1)(p^2+1)$. Increasing ω reduces the chaotic region, leading to a stable limit cycle.

Roe solver. Roe [60] introduced an approximate Riemann solver for hyperbolic conservation laws using the Godunov scheme. This solver estimates the numerical flux $\boldsymbol{F}$ at interfaces between neighboring cells in a discretized space–time domain. In one dimension, Roe discretizes (8) as

$$\boldsymbol{u}_j^{n+1} = \boldsymbol{u}_j^n - \lambda_r \left(\hat{\boldsymbol{F}}_{j+\frac{1}{2}}^n - \hat{\boldsymbol{F}}_{j-\frac{1}{2}}^n \right), \tag{A.14}$$

where $\lambda_r = \Delta t / \Delta x$ is the ratio of the temporal step size Δt to the spatial step size Δx, $j = 1, \ldots, N_g$ is the grid node index, and

$$\hat{\boldsymbol{F}}_{j+\frac{1}{2}}^n = \hat{\boldsymbol{F}}(\boldsymbol{u}_j^n, \boldsymbol{u}_{j+1}^n) \tag{A.15}$$

with

$$\hat{\boldsymbol{F}}(\boldsymbol{u}, \boldsymbol{v}) = \frac{1}{2}\left[\boldsymbol{F}(\boldsymbol{u}) + \boldsymbol{F}(\boldsymbol{v}) - |\tilde{\boldsymbol{A}}(\boldsymbol{u}, \boldsymbol{v})|(\boldsymbol{v} - \boldsymbol{u})\right]. \tag{A.16}$$

Designing an effective Roe solver depends on ensuring that the Roe matrix $\tilde{\boldsymbol{A}}$ meets three conditions: diagonalizability with real eigenvalues $\tilde{\boldsymbol{A}} = \boldsymbol{L}^{-1}\boldsymbol{\Lambda}\boldsymbol{L}$, where $\boldsymbol{L}$ is invertible and $\boldsymbol{\Lambda} = \mathrm{diag}(\Lambda_1, \ldots, \Lambda_{N_c})$ is a diagonal matrix; consistency with the Jacobian, $\lim_{\boldsymbol{u}_j, \boldsymbol{u}_{j+1} \to \boldsymbol{u}} \tilde{\boldsymbol{A}}(\boldsymbol{u}_j, \boldsymbol{u}_{j+1}) = \partial \boldsymbol{F}(\boldsymbol{u}) / \partial \boldsymbol{u}$; and preservation of the conservation law for the physical quantity $\boldsymbol{u}$ across cell interfaces: $\boldsymbol{F}_{j+1} - \boldsymbol{F}_j = \tilde{\boldsymbol{A}}(\boldsymbol{u}_{j+1} - \boldsymbol{u}_j)$. Substituting Eqs. (A.15), (A.16), and $|\tilde{\boldsymbol{A}}| = \boldsymbol{L}^{-1}|\boldsymbol{\Lambda}|\boldsymbol{L}$, where $|\boldsymbol{\Lambda}| = \mathrm{diag}(|\Lambda_1|, \ldots, |\Lambda_{N_c}|)$, into Eq. (A.14) along with the third Roe condition yields

$$\begin{aligned} \boldsymbol{u}_j^{n+1} =& \boldsymbol{u}_j^n - \frac{1}{2}\lambda_r [(\boldsymbol{L}^n_{j+\frac{1}{2}})^{-1}(\boldsymbol{\Lambda}^n_{j+\frac{1}{2}} - |\boldsymbol{\Lambda}^n_{j+\frac{1}{2}}|)\boldsymbol{L}^n_{j+\frac{1}{2}}(\boldsymbol{u}^n_{j+1} - \boldsymbol{u}^n_j) \\ &+ (\boldsymbol{L}^n_{j-\frac{1}{2}})^{-1}(\boldsymbol{\Lambda}^n_{j-\frac{1}{2}} + |\boldsymbol{\Lambda}^n_{j-\frac{1}{2}}|)\boldsymbol{L}^n_{j-\frac{1}{2}}(\boldsymbol{u}^n_j - \boldsymbol{u}^n_{j-1})], \end{aligned} \tag{A.17}$$

with

$$\boldsymbol{L}^n_{j+\frac{1}{2}} = \boldsymbol{L}(\boldsymbol{u}^n_j, \boldsymbol{u}^n_{j+1}), \quad \boldsymbol{\Lambda}^n_{j+\frac{1}{2}} = \boldsymbol{\Lambda}(\boldsymbol{u}^n_j, \boldsymbol{u}^n_{j+1}). \tag{A.18}$$

Eq. (A.17) describes the evolution from $\boldsymbol{u}_j^n$ to $\boldsymbol{u}_j^{n+1}$ in the Roe solver framework.

Lagrangian vortex method (LVM). The LVM discretizes (9) using N particles, transforming it into a system of ODEs governing both the strengths $\boldsymbol{\Gamma} = \{\boldsymbol{\Gamma}_i \mid i = 1, \ldots, N\}$ and positions $\boldsymbol{X} = \{\boldsymbol{X}_i \mid i = 1, \ldots, N\}$:

$$\frac{\mathrm{d}\boldsymbol{\Gamma}_i}{\mathrm{d}t} = \boldsymbol{\gamma}_i, \quad \frac{\mathrm{d}\boldsymbol{X}_i}{\mathrm{d}t} = \boldsymbol{u}_i + \boldsymbol{v}_i. \tag{A.19}$$

Here, $\boldsymbol{\Gamma}_i$ represents the particle strength, derived as the integral of $\boldsymbol{\omega}$ over the ith computational element, and $\boldsymbol{u}_i$ is the induced velocity

calculated by the Biot–Savart (BS) law. Additionally, $\boldsymbol{\gamma}_i$ and $\boldsymbol{v}_i$ denote the change rate of particle strength and drift velocity [61].

Data availability

Data will be made available on request.

References

[1] E. Weinan, The dawning of a new era in applied mathematics, Notices Amer. Math. Soc. 68 (2021) 565–571.
[2] S. Brunton, J. Kutz, Data-Driven Science and Engineering: Machine Learning, Dynamical Systems, and Control, Cambridge University Press, 2022.
[3] J. Schmidhuber, Deep learning in neural networks: An overview, Neural Netw. 61 (2015) 85–117.
[4] G. Karniadakis, I. Kevrekidis, L. Lu, P. Perdikaris, S. Wang, L. Yang, Physics-informed machine learning, Nat. Rev. Phys. 3 (2021) 422–440.
[5] C. Udriste, M. Ferrara, D. Zugrăvescu, F. Munteanu, Controllability of a nonholonomic macroeconomic system, J. Optim. Theory Appl. 154 (2012) 1036–1054.
[6] C. Udriste, M. Ferrara, Multitime models of optimal growth, WSEAS Trans. Math. 7 (2008) 51–55.
[7] K. He, X. Zhang, S. Ren, J. Sun, Deep residual learning for image recognition, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognitionn, 2016, pp. 770–778.
[8] R. Chen, Y. Rubanova, J. Bettencourt, D. Duvenaud, Neural ordinary differential equations, in: Conference on Neural Information Processing Systems, 2018.
[9] J. Ma, S. Dong, G. Chen, P. Peng, L. Qian, A data–driven normal contact force model based on artificial neural network for complex contacting surfaces, Mech. Syst. Signal Proc. 156 (2021) 107612.
[10] A. Frankel, C. Hamel, D. Bolintineanu, K. Long, S. Kramer, Machine learning constitutive models of elastomeric foams, Comput. Method Appl. Mech. Eng. 391 (2022) 114492.
[11] F. Masi, I. Stefanou, P. Vannucci, V. Maffi-Berthier, Thermodynamics-based artificial neural networks for constitutive modeling, J. Mech. Phys. Solids 147 (2021) 104277.
[12] L. Zhang, L. Cheng, H. Li, J. Gao, C. Yu, R. Domel, Y. Yang, S. Tang, W. Liu, Hierarchical deep–learning neural networks: finite elements and beyond, Comput. Mech. 67 (2021) 207–230.
[13] S. Saha, Z. Gan, L. Cheng, J. Gao, O. Kafka, X. Xie, H. Li, M. Tajdari, H. Kim, W. Liu, Hierarchical deep learning neural network (HiDeNN): An artificial intelligence (AI) framework for computational science and engineering, Comput. Method Appl. Mech. Eng. 373 (2021) 113452.
[14] Y. Liu, C. Park, Y. Lu, S. Mojumder, W. Liu, D. Qian, HiDeNN-FEM: a seamless machine learning approach to nonlinear finite element analysis, Comput. Mech. 72 (2023) 173–194.
[15] M. Huang, Z. Du, C. Liu, Y. Zheng, T. Cui, Y. Mei, X. Li, X. Zhang, X. Guo, Problem-independent machine learning (PIML)-based topology optimization-A universal approach, Extrem. Mech. Lett. 56 (2022) 101887.
[16] M. Huang, T. Cui, C. Liu, Z. Du, J. Zhang, C. He, X. Guo, A problem-independent machine learning (PIML) enhanced substructure-based approach for large-scale structural analysis and topology optimization of linear elastic structures, Extrem. Mech. Lett. 63 (2023) 102041.
[17] S. Greydanus, M. Dzamba, J. Yosinski, Hamiltonian neural networks, in: Proceedings of the 33rd International Conference on Neural Information Processing Systems, 2019.
[18] Z. Chen, J. Zhang, M. Arjovsky, L. Bottou, Symplectic recurrent neural networks, in: International Conference on Learning Representations, 2020.
[19] D. DiPietro, S. Xiong, B. Zhu, Sparse symplectically integrated neural networks, in: Advances in Neural Information Processing Systems, 2020.
[20] A. Sanchez-Gonzalez, V. Bapst, K. Cranmer, P. Battaglia, Hamiltonian graph networks with ODE integrators, 2019, arXiv:1909.12790.
[21] P. Battaglia, R. Pascanu, M. Lai, D.J. Rezende, Interaction networks for learning about objects, relations and physics, in: Advances in Neural Information Processing Systems, 29, 2016.
[22] P. Jin, Z. Zhang, A. Zhu, Y. Tang, G. Em Karniadakis, SympNets: Intrinsic structure-preserving symplectic networks for identifying Hamiltonian systems, Neural Netw. 132 (2020) 166–179.
[23] P. Toth, D. Rezende, A. Jaegle, S. Racaniére, A. Botev, I. Higgins, Hamiltonian generative networks, in: International Conference on Learning Representations, 2020.
[24] Y. Zhong, B. Dey, A. Chakraborty, Symplectic ODE-Net: learning Hamiltonian dynamics with control, in: International Conference on Learning Representations, 2020.
[25] B. Daniels, I. Nemenman, Automated adaptive inference of phenomenological dynamical models, Nat. Commun. 6 (2015) 8133.
[26] J. Wang, J. Wu, H. Xiao, Physics-informed machine learning approach for reconstructing Reynolds stress modeling discrepancies based on DNS data, Phys. Rev. Fluids 2 (2017) 034603.
[27] J. Hammond, F. Montomoli, M. Pietropaoli, R. Sandberg, V. Michelassi, Machine learning for the development of data-driven turbulence closures in coolant systems, J. Turbomach. 144 (2022) 081003.
[28] X. Xu, F. Waschkowski, A. Ooi, R. Sandberg, Towards robust and accurate Reynolds-averaged closures for natural convection via multi-objective CFD-driven machine learning, Int. J. Heat Mass Transfer 187 (2022) 122557.
[29] M. Raissi, P. Perdikaris, G. 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.
[30] D. Zhang, L. Guo, G. Karniadakis, Learning in modal space: solving time-dependent stochastic PDEs using physics-informed neural networks, SIAM J. Sci. Comput. 42 (2019) A639–A665.
[31] C. Michoski, M. Milosavljevic, T. Oliver, D. Hatch, Solving differential equations using deep neural networks, Neurocomputing 399 (2019) 193–212.
[32] Z. Mao, A. Jagtap, G. Karniadakis, Physics-informed neural networks for high-speed flows, Comput. Method Appl. Mech. Eng. 360 (2020) 112789.
[33] M. Raissi, A. Yazdani, G. Karniadakis, Hidden fluid mechanics: Learning velocity and pressure fields from flow visualizations, Science 367 (2020) 1026–1030.
[34] K. Lye, S. Mishra, D. Ray, Deep learning observables in computational fluid dynamics, J. Comput. Phys. 410 (2020) 109339.
[35] A. Mohan, N. Lubbers, M. Chertkov, D. Livescu, Embedding hard physical constraints in neural network coarse-graining of three-dimensional turbulence, Phys. Rev. Fluids 8 (2023) 014604.
[36] M. Milano, P. Koumoutsakos, Neural network modeling for near wall turbulent flow, J. Comput. Phys. 182 (2002) 1–26.
[37] K. Duraisamy, G. Iaccarino, H. Xiao, Turbulence modeling in the age of data, Annu. Rev. Fluid Mech. 51 (2019) 357–377.
[38] X. Yang, S. Zafar, J. Wang, H. Xiao, Predictive large-eddy-simulation wall modeling via physics-informed neural networks, Phys. Rev. Fluids 4 (2019) 034602.
[39] X. Huang, X. Yang, R. Kunz, Wall-modeled large-eddy simulations of spanwise rotating turbulent channels—Comparing a physics-based approach and a data-based approach, Phys. Fluids 31 (2019) 125105.
[40] Y. Bin, L. Chen, G. Huang, X. Yang, Progressive, extrapolative machine learning for near-wall turbulence modeling, Phys. Rev. Fluids 7 (2022) 084610.
[41] J. Tang, L. Qian, J. Ma, L. Chen, G. Chen, Z. Chen, W. Huang, Knowledge-dominated and data-driven rigid-flexible coupling dynamics for rotating flexible structure, Know.- Based Systs. 296 (2024) 111853.
[42] Y. Tong, S. Xiong, X. He, G. Pan, B. Zhu, Symplectic neural networks in taylor series form for Hamiltonian systems, J. Comput. Phys. 437 (2021) 110325.
[43] S. Xiong, Y. Tong, X. He, S. Yang, C. Yang, B. Zhu, Nonseparable symplectic neural networks, in: International Conference on Learning Representations, 2021.
[44] Y. Tong, S. Xiong, X. He, S. Yang, Z. Wang, R. Tao, R. Liu, B. Zhu, RoeNet: Predicting discontinuity of hyperbolic systems from continuous data, Internat. J. Numer. Methods Engrg. 125 (2024) e7406.
[45] S. Xiong, X. He, Y. Tong, Y. Deng, B. Zhu, Neural vortex method: From finite Lagrangian particles to infinite dimensional Eulerian dynamics, Comput. & Fluids 258 (2023) 105811.
[46] A. Shabana, Dynamics of Multibody Systems, Cambridge University Press, New York, 2020.
[47] Q. Peng, M. Li, Comparison of finite element methods for dynamic analysis about rotating flexible beam, Nonlinear Dynam. 111 (2023) 13753–13779.
[48] J. Tang, L. Qian, L. Chen, G. Chen, Y. Li, Flexible multibody dynamic analysis of shells with an edge center-based strain smoothing MITC method, Nonlinear Dynam. 111 (2023) 3253–3277.
[49] A.J. Majda, A.L. Bertozzi, Vorticity and Incompressible Flow, Cambridge University Press, 2001.
[50] H. Helmholtz, Uber integrale der hydrodynamischen Gleichungen welche den Wirbel-bewegungen ensprechen, J. Reine Angew. Math. 55 (1858) 25–55.
[51] Y. Yang, D. Pullin, On Lagrangian and vortex-surface fields for flows with Taylor–Green and Kida–Pelz initial conditions, J. Fluid Mech. 661 (2010) 446–481.
[52] S. Xiong, Y. Yang, The boundary-constraint method for constructing vortex-surface fields, J. Comput. Phys. 339 (2017) 31–45.
[53] J. Redmon, S. Divvala, R. Girshick, A. Farhadi, You only look once: unified, real-time object detection, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognitionn, 2016.
[54] T. Lin, P. Goyal, R. Girshick, K. He, P. Dollár, Focal loss for dense object detection, IEEE Trans. Vis. Comput. Graphics (2017) 2980–2988.
[55] M. Tao, Explicit symplectic approximation of nonseparable Hamiltonians: Algorithm and long time performance, Phys. Rev. E 94 (2016) 043303.
[56] Z. Qu, X. Zhang, M. Gao, C. Jiang, B. Chen, Efficient and conservative fluids using bidirectional mapping, ACM Trans. Graph. 38 (2019) 4.
[57] T. Hughes, I. Williamson, M. Minkov, S. Fan, Wave physics as an analog recurrent neural network, Sci. Adv. 5 (2019) 6946.
[58] E. Forest, R. Ruth, Fourth-order symplectic integration, Phys. D 43 (1990) 105–117.

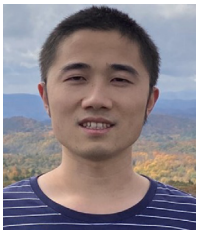
[59] A. Kolmogorov, On the conservation of conditionally periodic motions under small perturbation of the Hamiltonian, Dokl. Akad. Nauk SSSR 98 (1954) 527–530.
[60] P. Roe, Approximate riemann solvers, parameter vectors and difference schemes, J. Comput. Phys. 43 (1981) 357–372.
[61] J. Hao, S. Xiong, Y. Yang, Tracking vortex surfaces frozen in the virtual velocity in non-ideal flows, J. Fluid Mech. 863 (2019) 513–544.

Jinsong Tang received his Ph.D. in engineering from Nanjing University of Science and Technology, Nanjing, China, in 2024. He is now a postdoctoral fellow at Zhejiang University majoring in Mechanics, and his research interests include flexible manipulator dynamics, data-driven computational mechanics.

Yunjin Tong received her B.A. degree in Mathematics and Computer Science from Dartmouth College in 2024. She is now a Ph.D. student at Stanford Graduate School of Business in Operations Information and Technology. Her research focuses on machine learning, environmental impact of AI and data centers, and computer simulation.

Chen Lihua received her B.Eng degree from the Xi'an Jiaotong University in 1994 and Ph.D. degree from Zhejiang University, China, in 1999. She is now an associate professor at Zhejiang University majoring in fluid mechanics. Her research interests include heat transfer, energy storage and conversion,computational flow dynamics.

Shengze Cai received the B.Sc. and the Ph.D. degrees from Zhejiang University, Hangzhou, China, in 2014 and 2019, respectively. He is currently a tenure-track assistant professor with the College of Control Science and Engineering, Zhejiang University (ZJU). Prior to joining ZJU, he was a Post-Doctoral Research Associate with the Division of Applied Mathematics, Brown University, Providence, RI, USA. His research interests include scientific machine learning, control and optimization as well as flow visualization techniques.



Shiying Xiong received his B.S. in Physics from Jilin University (2014) and Ph.D. in Fluid Mechanics from Peking University (2019). He was a postdoctoral researcher at Dartmouth College and briefly at HKUST (2019–2022). Currently, he is an Assistant Professor at Zhejiang University, focusing on vortex dynamics, interfacial dynamics, and scientific machine learning.