


Past rewinding of fluid dynamics from noisy observation via physics-informed neural computingJaemin Seo ^{*}*Department of Physics, Chung-Ang University, Seoul 06974, South Korea*

(Received 11 July 2023; revised 20 March 2024; accepted 3 June 2024; published 6 August 2024)

Reconstructing the past of observed fluids has been known as an ill-posed problem due to both numerical and physical challenges, especially when observations are distorted by inevitable noise, resolution limits, or unknown factors. When employing traditional differencing schemes to reconstruct the past, the computation often becomes highly unstable or diverges within a few backward time steps from the distorted and noisy observation. Although several techniques have been recently developed for inverse problems, such as adjoint solvers and supervised learning, they are also unrobust against errors in observation when there is time-reversed simulation. Here we present that by using physics-informed neural computing, robust time-reversed fluid simulation is possible. By seeking a solution that closely satisfies the given physics and observations while allowing for errors, it reconstructs the most probable past from noisy observations. Our work showcases time rewinding in extreme fluid scenarios such as shock, instability, blast, and magnetohydrodynamic vortex. Potentially, this can be applied to trace back the interstellar evolution and determining the origin of fusion plasma instabilities.

DOI: [10.1103/PhysRevE.110.025302](https://doi.org/10.1103/PhysRevE.110.025302)**I. INTRODUCTION**

For decades, computational fluid dynamics (CFD) has seen dazzling progress, powered by advancements in computer technology and computational sciences [1]. This development has enabled CFD to successfully simulate the temporal evolution of diverse fluids, ranging from simple incompressible fluid to extreme conditions such as three-dimensional (3D) shock or rarefaction. Furthermore, it has been actively utilized in predictions of magnetohydrodynamics (MHD), such as the evolution of interstellar matters and fusion plasmas. Discretization methods such as the finite difference method (FDM) or finite volume method (FVM) have been actively used for fluid simulation [1]. More recently, machine-learning-based approaches such as adjoint solvers based on differentiable physics have also been developed [2,3], while they are predominantly used for incompressible flow. These methods successfully simulate various fluids by advancing in time from given initial conditions and unfolding the state at each step.

However, backward (or inverse) problems, where a future state is given and the task is to reconstruct the past state that leads to it, are known to be ill-posed problems that are difficult to solve with these traditional methods [4]. Using conventional approaches for backward reconstruction tends to be much more unstable or often leads to numerical divergence. Several approaches have been introduced and developed for solving some kinds of inverse problems, such as estimating unknown coefficients [5], reconstructing internal structures from indirect measurements [6,7], or restoring the state before processes in images [8–10]. However, in the context of time-reversing inverse problems, which is our interest in this work,

relatively few studies have been conducted beyond the limited techniques of supervised learning methods [11]. Things get even worse when the future state includes uncertainties and noise, which are inevitable under real-world physical observations. Even if it is a physical state where the past actually exists, an observed state distorted by noise may not have an exact past state that leads to it.

Figure 1 shows the forward prediction and backward rewinding of the dynamics of the Sod shock tube and Rayleigh-Taylor instability, solved by traditional differencing schemes (Lax-Wendroff FDM and FVM, respectively). In the Sod shock tube example, although the forward prediction exhibits plausible profiles [Fig. 1(a)], the backward rewinding produces unphysical fluctuations near the shock front [Figs. 1(b) and 1(c)]. This is because numerical errors at different discontinuities are summed as their positions converge to a single point, even though the CFL condition is satisfied.

When reconstructing the past from an unreachable state, distorted by noise, it becomes easier for computation to diverge in differencing schemes. Figure 1(e) is a progressed state of the Rayleigh-Taylor instability that occurred in a situation where denser fluid is on top [Fig. 1(d)], which is a physically reachable state under the Euler equation system. Figure 1(f) is a noisy and blurred observation, which is actually a physically distorted state and cannot be reachable under the Euler equation system. When performing backward rewinding through FVM from this unreachable distorted state, values near the interface diverge or imaginary numbers appear after a few steps, as shown with null regions in Fig. 1(g).

Recently, physics-informed neural networks (PINNs) [12–16] have been introduced as an alternative to traditional numerical-solving techniques. Unlike data-driven neural network applications for fluidlike dynamics [17–19], PINN incorporates physics laws, rather than given data, into objectives for training the neural network. It aims

^{*}Contact author: jseo@cau.ac.kr

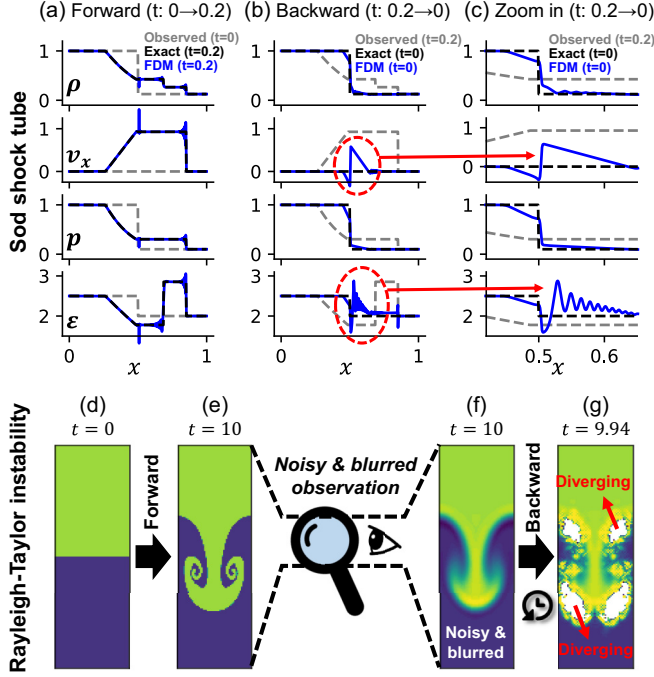


FIG. 1. Forward prediction and backward rewinding using traditional schemes for extreme fluid dynamics scenarios. (a) Forward prediction and (b),(c) backward rewinding with the Lax-Wendroff scheme for the Sod shock tube example, while the backward one shows larger numerical fluctuations (red dashed circles) near the shock front. Here, the dashed light-gray lines are the observed states, the dashed black lines are the exact solutions, and the solid blue lines are the numerical solution obtained by FDM. Rayleigh-Taylor instability at (d) $t = 0$ and (e) $t = 10$. (f) Observed state of the instability at $t = 10$, with added noise and blurs. (g) Backward rewinding from the observed state using a finite volume method, showing numerical oscillation and divergence, as marked with red arrows.

to optimize a neural network (θ) that maps continuous spacetime (\mathbf{x}, t) to fluid states (u), in a direction reducing the error with the governing physics equations. Therefore, from the beginning of the computation, the fluid state across the entire time domain is simultaneously updated, while traditional differencing schemes compute the states by advancing over time. Due to these characteristic differences, PINNs demonstrate an advantage in finding solutions to inverse problems [12, 14, 20], which have posed difficulties for traditional schemes. In this study, we introduce a PINN-based approach as a different methodology for a time-reversal problem, past rewinding of fluid dynamics. In particular, from observed states that include unavoidable noise and distortion, PINNs can reconstruct the most probable past.

II. SOLVING FLUID DYNAMICS USING PINN

To predict fluid dynamics, one must find a solution that satisfies the governing equations, along with the given constraints such as initial and boundary conditions. A set of the 2D Euler equations (\mathcal{F}) for fluid dynamics is shown in Eq. (1). Here, ρ and E are the mass density and the energy, and v_x or v_y is each component of the fluid velocity v . The pressure p is determined by the equation of state shown in Eq. (2), where

γ is the adiabatic index. \mathcal{S} represents the source term, such as gravity, external drive, or sink:

$$\mathcal{F} = \frac{\partial}{\partial t} \begin{pmatrix} \rho \\ \rho v_x \\ \rho v_y \\ E \end{pmatrix} + \frac{\partial}{\partial x} \begin{pmatrix} \rho v_x \\ \rho v_x^2 + p \\ \rho v_x v_y \\ (E + p)v_x \end{pmatrix} + \frac{\partial}{\partial y} \begin{pmatrix} \rho v_y \\ \rho v_y v_x \\ \rho v_y^2 + p \\ (E + p)v_y \end{pmatrix} - \mathcal{S} = 0, \quad (1)$$

$$p = (\gamma - 1) \left[E - \frac{1}{2} \rho (v_x^2 + v_y^2) \right]. \quad (2)$$

In the general PINN workflow to solve fluid dynamics, the goal is to optimize a function [$\hat{u} = \theta(x, y, t)$] that maps continuous spacetime to fluid states in a direction that satisfies the governing equations. By constructing this mapping function with a neural network, in which all nodes are differentiable, one can calculate the exact derivative values required for the partial differential equations (PDEs) using the chain rule.

The neural network θ takes spacetime coordinate inputs (x, y, t) and computes the output states ($\hat{u} = \{\rho, v, E\}$). To optimize the internal parameters consisting of θ , an objective function must be set. While the objective function of a data-driven neural network is the difference between the true and predicted values, the objective function in a PINN is designed as the error in the given governing equations. In this study, the objective function consists of a weighted sum ($\mathcal{L}_{\text{total}}$) of three terms in Eqs. (3)–(5): PDE loss \mathcal{L}_{PDE} , boundary condition loss \mathcal{L}_{BC} , and initial condition loss \mathcal{L}_{IC} ,

$$\mathcal{L}_{PDE}(\theta) = \frac{1}{N_{\Omega}} \sum_{j=1}^{N_{\Omega}} [\|\mathcal{F}(x_j, y_j, t_j; \theta)\|_2^2] \quad \text{for } (x_j, y_j, t_j) \in \Omega, \quad (3)$$

$$\mathcal{L}_{BC}(\theta) = \frac{1}{N_{\partial\Omega_{xy}}} \sum_{j=1}^{N_{\partial\Omega_{xy}}} [\|\hat{u}(x_j, y_j, t_j; \theta) - u_{BC}\|_2^2] \quad \text{for } (x_j, y_j, t_j) \in \partial\Omega_{xy}, \quad (4)$$

$$\mathcal{L}_{IC}(\theta) = \frac{1}{N_{\partial\Omega_t}} \sum_{j=1}^{N_{\partial\Omega_t}} [\|\hat{u}(x_j, y_j, t_j; \theta) - u_{IC}\|_2^2] \quad \text{for } (x_j, y_j, t_j) \in \partial\Omega_t, \quad (5)$$

$$\mathcal{L}_{\text{total}}(\theta) = w_{PDE} \mathcal{L}_{PDE}(\theta) + w_{BC} \mathcal{L}_{BC}(\theta) + w_{IC} \mathcal{L}_{IC}(\theta). \quad (6)$$

Here, Ω is the spacetime domain of interest and $\partial\Omega_{xy}$ or $\partial\Omega_t$ is its spatial or temporal boundaries. $N_{\Omega, \partial\Omega_{xy}}$ or $N_{\partial\Omega_t}$ is the sampling counts on each domain. $w_{PDE, BC, \text{ or } IC}$ is the weight for each loss value. Using the calculated fluid states (\hat{u}) and their derivatives ($\frac{\partial \hat{u}}{\partial x}$, $\frac{\partial \hat{u}}{\partial y}$, and $\frac{\partial \hat{u}}{\partial t}$) for the given θ , the loss values from Eqs. (3)–(6) can be obtained. Note that in the loss function, unlike traditional “data-driven” deep learning, the error between the prediction and the true label from the given

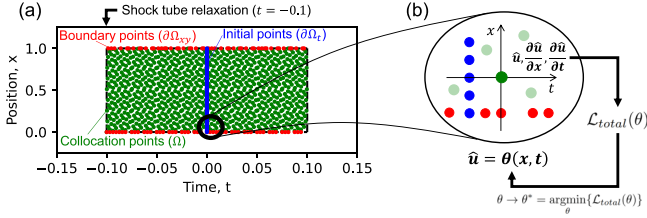


FIG. 2. Description of the domain and the optimization of the neural network for the Sod shock tube example. (a) The spatial and temporal domain including the collocation points (Ω , green at $0 < x < 1$ and $-0.1 < t < 0.1$), boundary points ($\partial\Omega_{xy}$, red at $x = 0$ and $x = 1$), and initial points ($\partial\Omega_t$, blue at $t = 0$). (b) Brief description of computing the loss function and updating the neural network.

training data is not used. The solution function for the fluid dynamics (θ^*) can be determined through the optimization process in Eq. (7),

$$\theta \rightarrow \theta^* = \operatorname{argmin}_{\theta} \{\mathcal{L}_{total}(\theta)\}. \quad (7)$$

In this work, the optimization of PINN is done by the Adam [21] and L-BFGS [22] algorithms implemented by the DeepXDE PYTHON library [15]. More detailed numerical settings with actual scripts can be found in Ref. [23].

A. Sod shock tube example

The Sod shock tube example is one of the common test problems for fluid computation, whose time evolution can be described by solving the Euler equations. In this section, we aim to show that PINNs can perform not only forward prediction, but also backward rewinding in the Sod shock tube problem, where the latter has been challenging via traditional

solvers. To demonstrate this, we have taken an observation not from the typical initial state of the Sod shock tube, but from a state that has progressed by $\Delta t = 0.1$ [which is the state of $t = 0$ in Figs. 2(a) and 3(e)], as shown with the blue dots at $t = 0$ in Fig. 2(a). These observation points were uniformly chosen, and the observed values are input as u_{IC} from Eq. (5). Additionally, the values at the boundary points (red dots) are input as u_{BC} from Eq. (4). Subsequently, Eq. (3) is evaluated for the chosen collocation points (green dots) that include both the past and the future, and the neural network is optimized, as described in Fig. 2(b). Here, the initial points were 1000 uniform points, and the collocation and boundary points were 1000 and 100 points, respectively, from the Hammersley point set, which is a low-discrepancy sequence.

The observed state of the Sod shock tube example is shown in Fig. 3(e), which sets the state at $t = 0$ in Fig. 2(a). Then, we performed temporally bidirectional solving with PINNs. The history of loss values over iterations can be seen in Fig. 3(a), where the loss values are converged after ~ 2500 iterations. Figure 3(b) shows the converged spatiotemporal profile of mass density, with the given observed state represented by the black dashed line. Centered on the observed time point $t = 0$ [Fig. 3(e)], profiles of ρ , v_x , and p from $t = -0.1$ to 0.1 are shown in Figs 3(c)–3(g). For future prediction ($t > 0$), the evolving patterns of the shock front and rarefaction are well captured. Since PINN does not use grids, it does not exhibit numerical oscillations induced by them, as shown in Fig. 1(a). Time-reversed reconstruction ($t < 0$) is also successfully demonstrated. A small error in v_x can be seen in the reconstruction of the initial state at $t = -0.1$, but this is considerably milder compared to the traditional results in Fig. 1(b). If the observed states are noisy or distorted, PINN becomes more favorable for time rewinding than differencing schemes, which will be discussed in the next section. Given

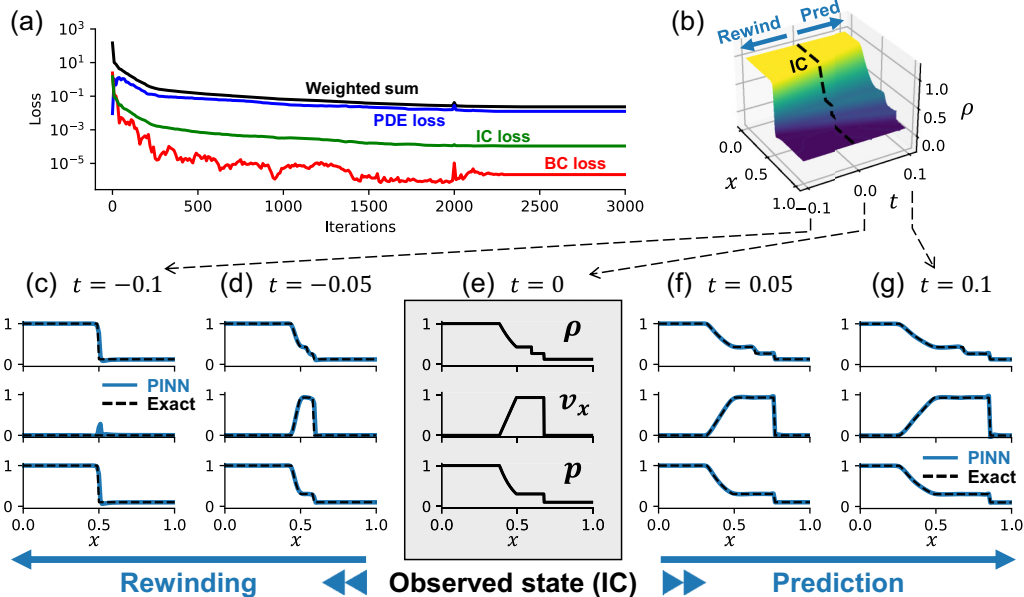


FIG. 3. Overview of the fluid simulation using PINN. (a) The loss history vs optimizing iterations. (b) The spatiotemporal profile of mass density after the convergence, while the observed state at $t = 0$ is indicated with a black dashed line. (c)–(g) The spatial profiles of ρ , v_x , and p at $t = -0.1$ to $t = 0.1$. (e) The given observation, (c),(d) The time-rewound states; (f),(g) the future-predicted states. Solid blue lines are the PINN results, and dashed black lines are the exact solutions.

TABLE I. The mean-squared error of density estimation using FDM and PINN for the Sod shock tube problem. For visibility, the smaller errors between FDM and PINN are italicized. Here, the computational times of FDM and PINN are 1.5 s and 47 s, respectively, for the “Observation w/o noise” case.

Time		-0.1	-0.05	0.05	0.1
Observation w/o noise	FDM	0.03155	0.01089	0.007689	0.007623
	PINN	0.06526	0.01326	0.009529	0.009907
Observation with 2% noise	FDM	0.04290	0.02776	0.02403	0.02389
	PINN	0.06191	0.01571	0.01341	0.01525
Observation with 4% noise	FDM	inf	inf	inf	inf
	PINN	0.06908	0.01761	0.01143	0.01954

that PINN has no constraints on the observation time point or temporal direction of solving, both the future prediction and the past reconstruction are performed through a single convergence of the optimization.

B. Influence of noise on the observation

We observed that past rewinding using PINN (Fig. 3) reconstructs a cleaner profile compared to the traditional method (Fig. 1), while well capturing the discontinuity positions. However, it is necessary to numerically assess how superior it is, especially under observation conditions with inevitable noise.

Table I compares the mean-squared errors in density estimation according to the noise levels in the observation state at $t = 0$ for the Sod shock tube problem discussed in Fig. 3. In each case, the method (FDM or PINN) that resulted in less error when predicting the state is highlighted in blue. In the case of observation without noise, FDM showed less prediction error than PINN in both future prediction and past rewinding. This indicates that when the exact state is known, FDM can provide more accurate predictions than PINN. However, when noise is added to the observation, the error in FDM predictions increases rapidly. The errors of the FDM predictions doubled when 2% noise was added to the observation, and the errors diverged to infinity when 4% noise was added. When the noise level is high in FDM, the density goes below zero at some local points, causing numerical divergence. In contrast, PINN maintains a consistent level of error relative to the noise level.

Here, when 2% noise is added, both FDM and PINN show relatively large errors for distant past predictions ($t = -0.1$). This is due to the inevitable uncertainty inherent in ill-posed problems. Although PINN shows larger errors compared to FDM, the more important point is that the size of the errors in PINN does not vary significantly with the magnitude of the noise.

C. Exploration beyond the collocation domain

The PINN methodology, commonly used in scientific research, optimizes a neural network to minimize the loss function on a given collocation domain, Ω , shown in Fig. 2(a). Unlike other machine-learning applications, it does not heavily focus on extrapolation or generalization outside the collocation domain. This is because if the state at a new point outside that domain is of interest, one can simply include that point in the collocation domain without additional

information. However, as the collocation domain expands, issues of nonconvergence or lower optimization efficiency may arise [24,25]. In this case, the extrapolation capability of a PINN model optimized within a limited domain can mitigate these problems. In this section, we aim to examine the extrapolation capability of a PINN model trained in the collocation domain of $-0.1 \leq t \leq 0.1$ and $0 \leq x \leq 1$, shown in Fig. 2(a).

Figure 4(a) shows the collocation domain used for optimization (green filled area) and the areas for extrapolation (blue hatched area). Figures 4(b)–4(d) display the predicted states in the extrapolation area of $0.2 \leq t \leq 0.4$ and $-1.5 \leq x \leq 1.5$. Despite this area being far from the collocation domain more than twice the domain size, it is observed that the positions of the rarefaction, contact, and shock discontinuities are accurately reproduced. While conventional data-driven deep learning is generally vulnerable to extrapolation, interpreting the underlying physics through physics-informed learning provides superior extrapolation capability.

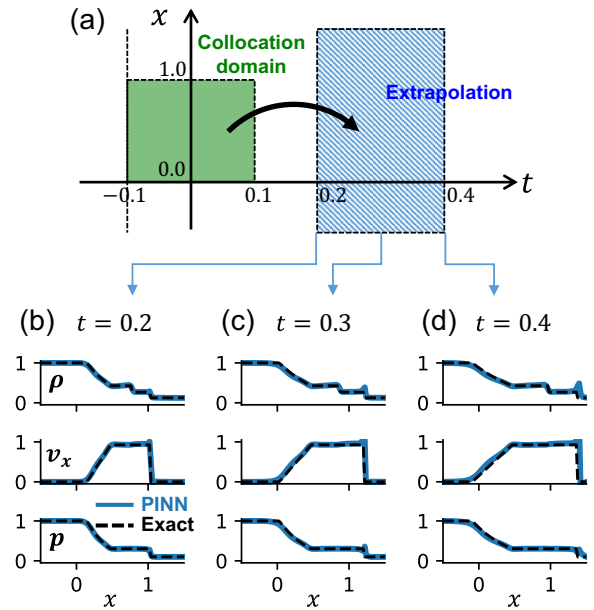


FIG. 4. Extrapolation results using the neural network trained from the limited domain. (a) The domain description for the training (green filled area) and the extrapolation (blue hatched area). (b)–(d) Extrapolation to the unseen area of $t = 0.2, 0.3,$ and 0.4 . Solid blue lines are the PINN results, and dashed black lines are the exact solutions.

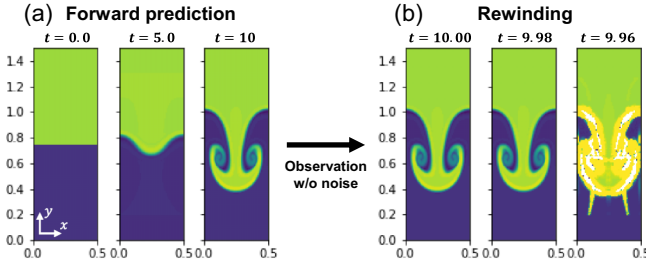


FIG. 5. Forward prediction and backward rewinding of Rayleigh-Taylor instability using an FVM solver. (a) Forward prediction from $t = 0$ to 10. (b) Backward rewinding from $t = 10$ to 9.96, eventually diverging near the interfaces, shown with blanks.

III. TIME REWINDING OF 2D FLUID DYNAMICS FROM DISTORTED OBSERVATIONS

Time rewinding via PINN can be usefully employed to estimate the past initial state of observed fluids. For instance, it could potentially reveal the origin of an observed instability in fusion plasma (which will be one of our future works), which is challenging to capture due to spatiotemporal resolution limits, or reconstruct the early distribution of interstellar materials from their limited observations. However, since time rewinding is an ill-posed problem, using conventional discretization method solvers (especially for problems in two dimensions or more) can easily lead to computation divergence, even if the CFL condition is well satisfied. Figure 5(a) shows the results of forward prediction for Rayleigh-Taylor instability using FVM, a representative discretization method, which displays a physically valid development process from $t = 0$ to 10. However, when performing backward rewinding starting from $t = 10$ [Fig. 5(b)], values near the interface rapidly increase (yellow) and diverge (blank area) after just

a few time steps. This is because the drastic difference in the states across adjacent meshes near the boundaries can lead to negative density or imaginary velocity during the backward calculation. In this section, we demonstrate that PINN, the mesh-free method, enables more stable backward rewinding calculations. Particularly, by performing backward rewinding for extreme fluid dynamics problems such as Rayleigh-Taylor instability, blast, and magnetohydrodynamic vortex, we will show that it can be applied to various practical problems.

A. Time rewinding of compressible Euler fluids

Figure 6 shows the results of time rewinding for the Rayleigh-Taylor instability and stellar blast by using PINNs. Here, we intentionally provided a low-resolution blurred observation [Fig. 6(a)] and an artificially distorted state that is difficult to exist [Fig. 6(d)] to demonstrate that PINN operates in situations where it becomes more difficult to use differencing schemes for rewinding, as shown in Figs. 6(c) and 6(f).

First, to solve for the Rayleigh-Taylor instability, gravity effects were added to S in Eq. (1). A small perturbation from equilibrium, with the denser fluid on top, leads to the growing instability shown in Fig. 6(a), after $\Delta t = 10$. Figure 6(b) sequentially presents the results of rewinding its past from this observed state. The past states are reconstructed in a pattern close to the nondistorted original solutions shown as white dashed lines. While a traditional differencing scheme, FVM, yields diverging errors after a few backward time steps from the noisy observation, as shown in Fig. 6(c), PINN could robustly rewind the instability towards the onset phase. This is because PINN allows for errors in the observed state and finds the closest solution, rather than strictly solving equations from an unreachable state such as FVM. Second, Fig. 6(d) presents the pressure of a virtual stellar blast centered on the coordinates $(x, y) = (2.75, 2.75)$. Here, the interface of

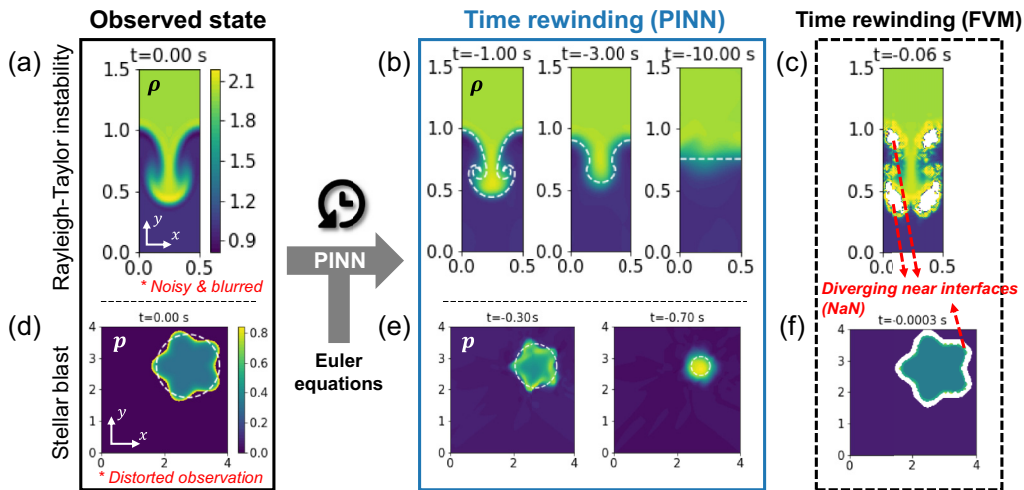


FIG. 6. Time rewinding using PINN for several extreme fluid examples in two dimensions. (a) Observation of a progressed state of Rayleigh-Taylor instability, where noise and blurs are intentionally added. (b) Time rewinding of the instability using PINN, successfully reconstructing a pattern close to the actual state (white dashes). (c) The rewinding of the instability using FVM, causing diverging errors near the interface regions after a few time steps. (d) Distorted observation of a progressed state of the stellar-shape blast, while a nondistorted state is shown with a white dashed line. (e) Time rewinding of the blast using PINN, successfully converging towards the original explosion point (2.75, 2.75). (f) Time rewinding of the blast using FVM, causing diverging errors near the interface regions. The nondistorted exact solutions are shown with white dashed lines for comparison.

the observed state was artificially distorted from a well-known Sedov circular blast [a white dash in Fig. 6(d)] to a stellar shape. This distortion can be due to measurement errors, dark matters, or undiscovered physics laws. Even though the observation was distorted, we can see the physically plausible convergence towards the original explosion point through time rewinding in Fig. 6(e). The converging speed and patterns are close to the nondistorted Sedov blast, shown with white dashes. In this case, FVM produces diverging errors, shown in Fig. 6(f), and shortly, the errors spread to the entire domain. For the sake of brevity, only limited variables are shown in Fig. 6, but evolving data of further variables can be found in Ref. [23].

In the case of 2D time rewinding, a larger blurriness at interfaces is observed in the PINN-reconstructed states in Figs. 6(b) and 6(e), compared to the 1D results in Fig. 3. Part of this blurriness can be interpreted as uncertainty due to the distortion of the observed state. Considering the difficulty in 2D rewinding tasks when using traditional schemes, shown in Figs. 6(c) and 6(f), the PINN results are quite encouraging. Even given the sparse, noisy, or limited observation, PINN finds a solution that approximates the observation as well as the governing equations.

B. Comparison to the adjoint solver using differentiable physics

Another precursor in physics-based deep learning is the adjoint solver [2] based on differentiable physics (DP) [3]. A DP solver is a computational tool that leverages the principles of differential equations and optimization techniques (utilizing machine learning) to efficiently solve inverse problems in physics. Notably, unlike FDM and FVM, the DP solver can prevent values from diverging to infinity or becoming imaginary. However, many DP solvers focus on demonstrating incompressible fluids (such as liquid simulation or simple smoke plumes) with incompressible Navier-Stokes equations, and applying them to Euler equations in this study (such as compressible Rayleigh-Taylor instability) requires additional techniques, beyond the scope of our work. Instead, in this section, we provide an indirect comparison with PINN results through the outcomes obtained using a DP solver for Rayleigh-Taylor instability in the incompressible limit. Figure 7 presents another baseline result for the incompressible Rayleigh-Taylor instability, showing the forward prediction [$t = 0$ to 10 in Fig. 7(a)] and backward rewinding [$t = 10$ to 0 in Fig. 7(b)] results calculated using PHIFLOW [3], a PYTHON package implementing the DP solver.

Note that in Fig. 7, due to the application of the incompressible constraint, the state at $t = 10$ differs from the compressible Rayleigh-Taylor instability shown in Fig. 5. Despite dealing with incompressible fluid, which has fewer governing equations and less complexity than compressible fluid, Fig. 7 shows that the DP solver relatively fails to reconstruct the past state. It does not diverge, unlike the results of FVM in Fig. 5, but during the rewinding process, small errors accumulate and amplify due to the butterfly effect, shown as red dashed circles in Fig. 7(b). The rewind result at $t = 0$ in Fig. 7(b) shows significant differences compared to the ground truth in Fig. 7(a).

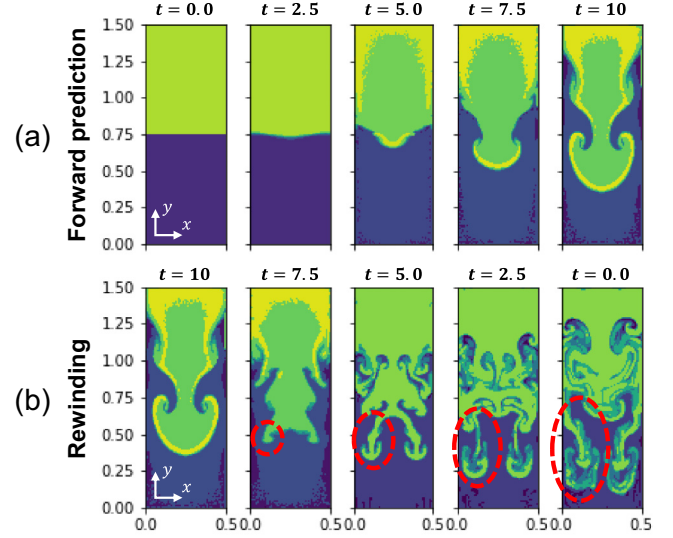


FIG. 7. Forward prediction and backward rewinding of Rayleigh-Taylor instability using a differentiable physics solver, PHIFLOW. (a) Forward prediction from $t = 0$ to 10. (b) Backward rewinding from $t = 10$ to 0. This solver does not diverge, but fails to reconstruct the initial state, as small errors amplify.

C. Time rewinding of magnetohydrodynamic vortex

To time rewind the dynamics of actual interstellar material or fusion plasma instability, it is necessary to solve magnetohydrodynamics (MHD), which considers the effects of magnetic fields (\mathbf{B}) on the fluids. 2D MHD consists of a set of equations, shown in Eq. (8), where more physical variables are tightly coupled than in Eq. (1). The equation of state is given by Eq. (9). Here, $B_{x \text{ or } y}$ is each component of the magnetic field, and $p_{\text{tot}} = p + \frac{1}{2}(B_x^2 + B_y^2)$ is the total pressure,

$$\mathcal{F} = \frac{\partial}{\partial t} \begin{pmatrix} \rho \\ \rho v_x \\ \rho v_y \\ B_x \\ B_y \\ E \end{pmatrix} + \frac{\partial}{\partial x} \begin{pmatrix} \rho v_x \\ \rho v_x^2 + p_{\text{tot}} - B_x^2 \\ \rho v_x v_y - B_x B_y \\ 0 \\ B_y v_x - B_x v_y \\ (E + p_{\text{tot}})v_x - B_x(\mathbf{v} \cdot \mathbf{B}) \end{pmatrix} + \frac{\partial}{\partial y} \begin{pmatrix} \rho v_y \\ \rho v_y v_x - B_y B_x \\ \rho v_y^2 + p_{\text{tot}} - B_y^2 \\ B_x v_y - B_y v_x \\ 0 \\ (E + p_{\text{tot}})v_y - B_y(\mathbf{v} \cdot \mathbf{B}) \end{pmatrix} = 0, \quad (8)$$

$$p = (\gamma - 1) \left[E - \frac{1}{2} \rho (v_x^2 + v_y^2) - \frac{1}{2} (B_x^2 + B_y^2) \right]. \quad (9)$$

To demonstrate time rewinding for MHD, we have brought the Orszag-Tang MHD vortex example. Figure 8(a) is the progressed MHD vortex observed at $t = 1.5$, with a limited resolution of (128×128) . Here, the boundary conditions are set as the periodic condition for both the x and y components, instead of a Dirichlet condition shown in Eq. (4). The results

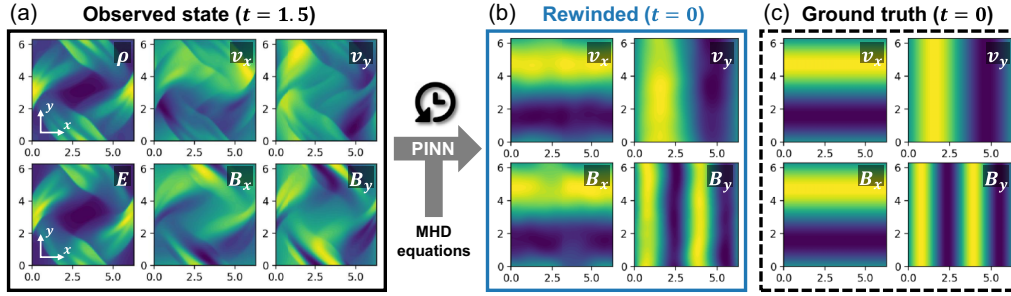


FIG. 8. Time rewinding using PINN for Orszag-Tang MHD vortex. (a) Observation of a progressed state of the Orszag-Tang MHD vortex at $t = 1.5$. (b) Time rewinding of the MHD vortex using PINN, successfully reconstructing the initial sinusoidal patterns. (c) The ground truth of the initial profiles at $t = 0$.

of reconstructing its initial state ($t = 0$) by using PINN are shown in Fig. 8(b). Note that since the periodic boundary condition is applied, not a Dirichlet, there are virtually no hints other than the MHD equations when reconstructing the initial state. Nonetheless, rewinding with PINN was able to reconstruct the patterns almost identical to the actual ground truth of sinusoidal profiles [Fig. 8(c)]. While visible errors do exist, PINN could clearly capture the physically meaningful mode structures, which are important to understand the source, stability, or impact of the MHD phenomenon. Furthermore, as PINN directly utilizes the deep-learning framework, which is rapidly advancing, its performance will continue to improve with advancements in neural network theory and optimization algorithms.

IV. CONCLUSION

In this study, we developed a PINN-based fluid simulation technique that can perform time rewinding of fluid states from noisy observation, which had limitations before due to not only numerical but also inherent physical issues with traditional differencing schemes such as FDM or FVM. Because observed states distorted by noise or unknown factors are states that cannot be reached by given physical laws, there has been difficulty in reconstructing their past with the traditional method of strictly calculating differential equations [see Fig. 6(c)]. On the other hand, when using PINN, we can reconstruct the probable past of the observed fluid, as shown in Fig. 6(b), as it can provide the most approximate solution while allowing errors in the given observations and governing equations. The incorporation of physical laws and observation data in PINN effectively serves as a “regularizer” as commonly employed in other inverse problems [12]. We demonstrated time rewinding not only for extreme dynamics such as instability and blast, but also for MHD vortex. Time rewinding of MHD will be applicable to reconstructing the evolution process of interstellar matter from observations, or understanding the origin of instabilities occurring in fusion plasmas. In particular, nuclear fusion devices such as KSTAR [26,27] are equipped with multidimensional imaging diagnostics to observe MHD instabilities [28,29]. However, it was nearly impossible to reconstruct the evolution process of instability backwards due to its limited observation area and resolution. Using PINN, reconstructing the early stage of MHD instabilities, such as disruptive tearing instability [30]

and MHD eigenmodes [31], will help us to understand their mechanisms.

Other areas where time rewinding can be applied include the following: (i) Estimating the epicenter of earthquakes or nuclear tests from shock waves observed on the ground and underwater. (ii) Estimating the origin of bacteria and virus spreads that follow a reaction-diffusion system similar to fluid. (iii) Understanding the early area of tokamak breakdown that was difficult to measure due to the validity regime limit of existing diagnostic systems.

However, this work serves as a proof of concept for past rewinding from noisy observations, and there still exist challenges when applying it to real physical phenomena. One of the current limitations of PINN is the decreased accuracy when trying to estimate points that are far from the observation time. Recently, techniques such as ensemble methods and time adaptation have been introduced to improve the long-term accuracy of PINN [25]. Since PINN directly utilizes versatile deep-learning frameworks, as AI technologies advance, particularly in neural network theory and optimization algorithms, we expect that the performance of numerical simulations applying PINN will also improve.

ACKNOWLEDGMENT

This work was supported by a National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIT) (Grant No. RS-2024-00346024).

APPENDIX: NUMERICAL SETTINGS AND PERFORMANCE

In implementing PINNs for the fluid dynamics problems in our paper, we utilized the DeepXDE library [15] with a TensorFlow backend [32]. The neural network weights were optimized using the Adam [21] and L-BFGS [22] algorithms, sequentially. The number of hidden layers and neurons was set to three layers with 32 neurons each for the 1D Sod problem and four layers with 64 neurons each for the 2D problems. Hyperbolic tangent (tanh) activation functions were applied to add nonlinearity in the hidden layers. Here, we observe that using piecewise functions for nonlinear activation such as ReLU causes lower performance, probably due to the edge effect near the boundary.

TABLE II. The numerical settings for training the neural network shown in this paper.

	Sod shock tube	Rayleigh-Taylor instability	Blast	MHD vortex
Governing equation	1D Euler equations	2D Euler equations	2D Euler equations	2D MHD equations
Inputs	(x, t)	(x, y, t)	(x, y, t)	(x, y, t)
Outputs	(ρ, v_x, E)	(ρ, v_x, v_y, E)	(ρ, v_x, v_y, E)	$(\rho, v_x, v_y, B_x, B_y, E)$
Layers \times neurons	3×32	4×64	4×64	4×64
Learning rate for Adam	N/A	0.001	0.01	0.001
Epochs for Adam	0	2×10^5	5×10^2	2×10^4
Epochs for L-BFGS	3×10^3	1×10^3	1×10^3	1×10^3
$(w_{PDE}, w_{BC}, w_{IC})$	(1, 1, 100)	(1, 1, 10)	(1, 1, 1)	(1, 1, 10)

For the 1D Euler equations, the input to the neural network is (x, t) and the output is (ρ, v_x, E) . In the 2D Euler equations, the input is (x, y, t) and the output is (ρ, v_x, v_y, E) . Lastly, for the 2D MHD equations, the input is (x, y, t) and the output is $(\rho, v_x, v_y, B_x, B_y, E)$. Other detailed settings for training are shown in Table II and the actual PYTHON scripts and observation data can be found in Ref. [23].

For the forward schemes, for comparison, the Lax-Wendroff finite difference method was used for the 1D problem, and the finite volume method was used for 2D problems.

The computational experiments in this study were performed on an Apple M1 processor. The elapsed wall time required for each problem to reach saturation was 47 seconds for the Sod problem, 21 minutes for the stellar blast, 17 hours for the Rayleigh-Taylor instability, and 13 hours for the MHD vortex. The time taken to reach saturation seems to vary depending on the characteristics of the problem, but more analysis will be needed regarding it. The computation time appears to be a drawback compared to differencing schemes that maintain relatively consistent wall times when the resolution is fixed.

- [1] J. Tu, G. H. Yeoh, C. Liu, and Y. Tao, *Computational Fluid Dynamics: A Practical Approach* (Elsevier, Amsterdam, 2024).
- [2] A. McNamara, A. Treuille, Z. Popović, and J. Stam, Fluid control using the adjoint method, *ACM Trans. Graph.* **23**, 449 (2004).
- [3] P. Holl, V. Koltun, and N. Thuerey, Learning to control PDEs with differentiable physics, [arXiv:2001.07457](https://arxiv.org/abs/2001.07457).
- [4] A. Tarantola, *Inverse Problem Theory and Methods for Model Parameter Estimation* (SIAM, Philadelphia, 2005).
- [5] S. Wang and R. Ni, Solving of two-dimensional unsteady-state heat-transfer inverse problem using finite difference method and model prediction control method, *Complexity* **2019**, 7432138 (2019).
- [6] C.-M. Fan, P.-W. Li, and W. Yeh, Generalized finite difference method for solving two-dimensional inverse cauchy problems, *Inverse Prob. Sci. Engineer.* **23**, 737 (2015).
- [7] Y. Gu, L. Wang, W. Chen, C. Zhang, and X. He, Application of the meshless generalized finite difference method to inverse heat source problems, *Intl. J. Heat Mass Transf.* **108**, 721 (2017).
- [8] A. Buades, B. Coll, and J. M. Morel, A review of image denoising algorithms, with a new one, *Multiscale Model. Simulat.* **4**, 490 (2005).
- [9] G. Yu, G. Sapiro, and S. Mallat, Solving inverse problems with piecewise linear estimators: From Gaussian mixture models to structured sparsity, *IEEE Trans. Image Proc.* **21**, 2481 (2012).
- [10] M. T. McCann, K. H. Jin, and M. Unser, Convolutional neural networks for inverse problems in imaging: A review, *IEEE Signal Proc. Mag.* **34**, 85 (2017).
- [11] M. Shustak and E. Landa, Time reversal for wave refocusing and scatterer detection using machine learning, *Geophysics* **83**, T257 (2018).
- [12] J. Seo, Solving real-world optimization tasks using physics-informed neural computing, *Sci. Rep.* **14**, 202 (2024).
- [13] B. P. van Milligen, V. Tribaldos, and J. A. Jiménez, Neural network differential equation and plasma equilibrium solver, *Phys. Rev. Lett.* **75**, 3594 (1995).
- [14] M. Raissi, P. Perdikaris, and 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**, 686 (2019).
- [15] L. Lu, X. Meng, Z. Mao, and G. E. Karniadakis, Deepxde: A deep learning library for solving differential equations, *SIAM Rev.* **63**, 208 (2021).
- [16] J. Seo, I. H. Kim, and H. Nam, Leveraging physics-informed neural computing for transport simulations of nuclear fusion plasmas, *Nucl. Eng. Technol.* (2024), doi:10.1016/j.net.2024.07.048.
- [17] J. Seo *et al.*, Feedforward beta control in the KSTAR tokamak by deep reinforcement learning, *Nucl. Fusion* **61**, 106010 (2021).
- [18] J. Seo *et al.*, Development of an operation trajectory design algorithm for control of multiple 0D parameters using deep

- reinforcement learning in KSTAR, *Nucl. Fusion* **62**, 086049 (2022).
- [19] R. Shousha *et al.*, Machine learning-based real-time kinetic profile reconstruction in DIII-D, *Nucl. Fusion* **64**, 026006 (2024).
- [20] S. Joung *et al.*, Gs-deepnet: Mastering tokamak plasma equilibria with deep neural networks and the Grad-Shafranov equation, *Sci. Rep.* **13**, 15799 (2023).
- [21] D. P. Kingma and J. Ba, Adam: A method for stochastic optimization, [arXiv:1412.6980](https://arxiv.org/abs/1412.6980).
- [22] D. C. Liu and J. Nocedal, On the limited memory BFGS method for large scale optimization, *Math. Program.* **45**, 503 (1989).
- [23] See <https://github.com/jaem-seo/time-rewinding-pinn>.
- [24] A. S. Krishnapriyan, A. Gholami, S. Zhe, R. M. Kirby, and M. W. Mahoney, Characterizing possible failure modes in physics-informed neural networks, *Advanc. Neural Inf. Proc. Syst.* **34**, 26548 (2021).
- [25] K. Haitsiukevich and A. Ilin, Improved training of physics-informed neural networks with model ensembles, *2023 International Joint Conference on Neural Networks (IJCNN)*, Gold Coast, Australia (IEEE, 2023), pp. 1–8.
- [26] G. Lee *et al.*, Design and construction of the KSTAR tokamak, *Nucl. Fusion* **41**, 1515 (2001).
- [27] Y.-S. Na *et al.*, Observation of a new type of self-generated current in magnetized plasmas, *Nat. Commun.* **13**, 6477 (2022).
- [28] G. S. Yun *et al.*, Development of KSTAR ECE imaging system for measurement of temperature fluctuations and edge density fluctuations, *Rev. Sci. Instrum.* **81**, 10D930 (2010).
- [29] M. J. Choi *et al.*, Effects of plasma turbulence on the nonlinear evolution of magnetic island in tokamak, *Nat. Commun.* **12**, 375 (2021).
- [30] J. Seo *et al.*, Avoiding fusion plasma tearing instability with deep reinforcement learning, *Nature (London)* **626**, 746 (2024).
- [31] J. Seo, Y.-S. Na and T. S. Hahm, Ion heating by nonlinear landau damping of high- n toroidal Alfvén eigenmodes in ITER, *Nucl. Fusion* **61**, 096022 (2021).
- [32] M. Abadi *et al.*, TensorFlow: Large-scale machine learning on heterogeneous systems (unpublished), <http://tensorflow.org/>, Software available from tensorflow.org.