SIAM J. SCI. COMPUT.
Vol. 46, No. 4, pp. A2528–A2556

ENERGETIC VARIATIONAL NEURAL NETWORK DISCRETIZATIONS OF GRADIENT FLOWS*

ZIQING HU†, CHUN LIU‡, YIWEI WANG§, AND ZHILIANG XU†

Abstract. We present a structure-preserving Eulerian algorithm for solving L^2-gradient flows and a structure-preserving Lagrangian algorithm for solving generalized diffusions. Both algorithms employ neural networks as tools for spatial discretization. Unlike most existing methods that construct numerical discretizations based on the strong or weak form of the underlying PDE, the proposed schemes are constructed based on the energy-dissipation law directly. This guarantees the monotonic decay of the system's free energy, which avoids unphysical states of solutions and is crucial for the long-term stability of numerical computations. To address challenges arising from nonlinear neural network discretization, we perform temporal discretizations on these variational systems before spatial discretizations. This approach is computationally memory-efficient when implementing neural network-based algorithms. The proposed neural network-based schemes are mesh-free, allowing us to solve gradient flows in high dimensions. Various numerical experiments are presented to demonstrate the accuracy and energy stability of the proposed numerical schemes.

Key words. structure-preserving, gradient flows, neural networks, energy stability, Lagrangian schemes

MSC codes. 35K35, 35K55, 49J40, 65M06, 65M12

DOI. 10.1137/22M1529427

1. Introduction.

Evolution equations with variational structures, often termed as gradient flows, have a wide range of applications in physics, material science, biology, and machine learning [14, 60, 78]. These systems not only possess but also are determined by an energy-dissipation law, which consists of an energy of state variables that describes the energetic coupling and competition, and a dissipative mechanism that drives the system to an equilibrium.

More precisely, beginning with a prescribed energy-dissipation law

$$\frac{\mathrm{d}}{\mathrm{d}t}E^{\mathrm{total}} = -\triangle \leq 0, \tag{1.1}$$

where E^{total} is the sum of the Helmholtz free energy $\mathcal{F}$ and the kinetic energy $\mathcal{K}$, and $\triangle$ is the rate of energy dissipation, one could derive the dynamics, the corresponding partial differential equation (PDE), of the system by combining the Least Action Principle (LAP) and the Maximum Dissipation Principle (MDP). The LAP states that the equation of motion of a Hamiltonian system can be derived from the variation of the action functional $\mathcal{A} = \int_0^T (\mathcal{K} - \mathcal{F})\mathrm{d}t$ with respect to state variable $\boldsymbol{x}$. This gives

*Submitted to the journal's Numerical Algorithms for Scientific Computing section October 18, 2022; accepted for publication (in revised form) April 12, 2024; published electronically August 5, 2024.

https://doi.org/10.1137/22M1529427

Funding: The work of the second and third authors was partially supported by the National Science Foundation grants DMS-1950868 and DMS-2153029. The work of the fourth author was partially supported by the National Science Foundation grant CDS&E-MSS 1854779.

†Department of Applied and Computational Mathematics and Statistics, University of Notre Dame, Notre Dame, IN 46556, USA (zhu4@nd.edu, zxu2@nd.edu).

‡Department of Applied Mathematics, Illinois Institute of Technology, Chicago, IL 60616, USA (cliu124@iit.edu).

§Corresponding author. Department of Mathematics, University of California, Riverside, Riverside, CA 92521, USA (yiweiw@ucr.edu).

rise to a unique procedure to derive the "conservative force" in the system. The MDP, on the other hand, derives the "dissipative force" by taking the variation of the dissipation potential $\mathcal{D}$, which equals $\frac{1}{2}\triangle$ in the linear response regime (near equilibrium), with respect to $\boldsymbol{x}_t$. In turn, the force balance condition leads to the PDE of the system

$$\frac{\delta \mathcal{D}}{\delta \boldsymbol{x}_t} = \frac{\delta \mathcal{A}}{\delta \boldsymbol{x}}. \tag{1.2}$$

This procedure is known as the energetic variational approach (EnVarA) [24, 48, 74], developed based on the celebrated work of Onsager [57, 58] and Rayleigh [63] in nonequilibrium thermodynamics. During the past decades, the framework of EnVarA has shown to be a powerful tool for developing thermodynamically consistent models for various complex fluid systems, including two-phase flows [80, 81, 83], liquid crystals [48], ionic solutions [22], and reactive fluids [75, 76]. In a certain sense, the energy-dissipation law (1.1), which comes from the first and second law of thermodynamics [23], provides a more intrinsic description of the underlying physics than the derived PDE (1.2), in particular for systems that possess multiple structures and involve multiscale and multiphysics coupling.

From a numerical standpoint, the energy-dissipation law (1.1) also serves as a valuable guideline for developing structure-preserving numerical schemes for these variational systems, as many straightforward PDE-based discretizations may fail to preserve the continuous variational structures, as well as the physical constraints, such as the conservation of mass or momentum, the positivity of mass density, and the dissipation of the energy. As a recent development in this field, in [50], the authors proposed a numerical framework, called a *discrete energetic variational approach,* to address these challenges. Similar methods are also used in [72, 82]. The key idea of this approach is to construct numerical discretizations directly based on the continuous energy-dissipation laws without using the underlying PDE (see section 3.1 for details). The approach has advantages in preserving a discrete counterpart of the continuous energy-dissipation law, which is crucial for avoiding unphysical solutions and the long-term stability of numerical computations. Within the framework of the *discrete energetic variational approach,* Eulerian [55], Lagrangian [50, 51], and particle methods [73] have been developed for various problems.

However, tackling high-dimensional variational problems remains a challenge. Traditional numerical discretizations, such as finite difference, finite element, and finite volume methods, suffer from the well-known curse of dimensionality (CoD) [27]. Namely, the computational complexity of the numerical algorithm increases exponentially with the dimensionality of the problem [4, 27]. Although particle methods hold promise for addressing high-dimensional problems, standard particle methods often lack accuracy and are not suitable for problems involving derivatives.

During recent years, neural networks have demonstrated remarkable success across a wide spectrum of scientific disciplines [12, 31, 41, 43, 47]. Leveraging the potent expressive power of neural network [3, 10, 20], particularly deep neural network (DNN) architectures, there exists a growing interest in developing neural network-based algorithms for PDEs, especially for those in high dimensions [13, 21, 37, 38, 42, 62, 68, 70, 77]. Examples include physics-informed neural network (PINN) [62], deep Ritz method (DRM) [21], deep Galerkin method (DGM) [68], variational PINN [37], and weak adversarial network (WAN) [84] to name a few. A key component of these aforementioned approaches is to represent solutions of PDEs via neural networks. All of these approaches determine the optimal parameters of the neural network by minimizing a

loss function, which is often derived from either the strong or weak form of the PDE (see section 2.1 for more details). By employing neural network approximations, the approximate solution belongs to a space of nonlinear functions, which may lead to a robust estimation by sparser representation and cheaper computation [37].

The goal of this paper is to combine neural network-based spatial discretization with the framework of the discrete energetic variational approach [50], to develop efficient and robust numerical schemes, termed as energetic variational neural network (EVNN) methods. To clarify the idea, we consider the following two types of gradient flows modeled by EnVarA:

- **L^2-gradient flow** that satisfies an energy-dissipation law

$$\frac{\mathrm{d}}{\mathrm{d}t}\mathcal{F}[\varphi] = -\int_\Omega \eta(\varphi)|\varphi_t|^2 \mathrm{d}\boldsymbol{x}. \tag{1.3}$$

- **Generalized diffusion** that satisfies an energy-dissipation law

$$\frac{\mathrm{d}}{\mathrm{d}t}\mathcal{F}[\rho] = -\int_\Omega \eta(\rho)|\mathbf{u}|^2 \mathrm{d}\boldsymbol{x}, \tag{1.4}$$

  where ρ satisfies $\rho_t + \nabla\cdot(\rho\mathbf{u}) = 0$, known as the mass conservation.

Derivation of the underlying PDEs for these types of systems is described in sections 2.1.1 and 2.1.2 of this paper.

Our primary aim is to develop structure-preserving Eulerian algorithms to solve L^2-gradient flows and structure-preserving Lagrangian algorithms to solve generalized diffusions based on their energy-dissipation law by utilizing neural networks as a new tool of spatial discretization. To overcome difficulties arising from neural network discretization, we develop a discretization approach that performs temporal discretizations before spatial discretizations. This approach leads to a computer-memory-efficient implementation of neural network-based algorithms. Since neural networks are advantageous due to their ability to serve as parametric approximations for unknown functions even in high dimensions, the proposed neural-network-based algorithms are capable of solving gradient flows in high-dimension, such as these appear in machine learning [19, 32, 69, 73].

The rest of this paper is organized as follows. Section 2 reviews the EnVarA and some existing neural network-based numerical approaches for solving PDEs. Section 3 of the paper is devoted to the development of the proposed EVNN schemes for L^2-gradient flows and generalized diffusions. Various numerical experiments are presented in section 4 to demonstrate the accuracy and energy stability of the proposed numerical methods. Conclusions are drawn in section 5.

2. Preliminary.

2.1. Energetic variational approach. In this subsection, we provide a detailed derivation of underlying PDEs for L^2-gradient flows and generalized diffusions by the general framework of EnVarA. We refer interested readers to [24, 74] for a more comprehensive review of the EnVarA. In both systems, the kinetic energy $\mathcal{K} = 0$.

2.1.1. L^2-gradient flow. L^2-gradient flows are those systems satisfying the energy-dissipation law:

$$\frac{\mathrm{d}}{\mathrm{d}t}\mathcal{F}[\varphi] = -\int_\Omega \eta|\varphi_t|^2 \mathrm{d}\boldsymbol{x}, \tag{2.1}$$

where φ is the state variable, $\mathcal{F}[\varphi]$ is the Helmholtz free energy, and $\eta > 0$ is the dissipation rate. One can also view φ as the generalized coordinates of the system

[14]. By treating φ as $\boldsymbol{x}$, the variational procedure (1.2) leads to the following L^2-gradient flow equation:

$$\frac{\delta(\frac{1}{2}\int \eta|\varphi_t|^2 \mathrm{d}\boldsymbol{x})}{\delta\varphi_t} = -\frac{\delta\mathcal{F}}{\delta\varphi} \quad \Rightarrow \quad \eta\varphi_t = -\frac{\delta\mathcal{F}}{\delta\varphi}.$$

Many problems in soft matter physics, material science, and machine learning can be modeled as L^2-gradient flows. Examples include Allen–Cahn equation [15], Oseen–Frank and Landau–de Gennes models of liquid crystals [11, 45], phase field crystal models of quasicrystal [34, 44], and generalized Ohta–Kawasaki model of diblock and triblock copolymers [56]. For example, by taking

$$\mathcal{F}[\varphi] = \int_\Omega \frac{1}{2}|\nabla\varphi|^2 + \frac{1}{4\epsilon^2}(\varphi^2-1)^2 \mathrm{d}\boldsymbol{x},$$

which is the classical Ginzburg–Landau free energy, and letting $\eta(\varphi) = 1$, one gets the Allen–Cahn equation

$$\varphi_t = \Delta\varphi - \frac{1}{\epsilon^2}(\varphi^2-1)\varphi.$$

2.1.2. Generalized diffusion. Generalized diffusion describes the space-time evolution of a conserved quantity $\rho(\boldsymbol{x},t)$. Due to the physics law of mass conservation, $\rho(\boldsymbol{x},t)$ satisfies the kinematics

$$\partial_t\rho + \nabla\cdot(\rho\mathbf{u}) = 0, \tag{2.2}$$

where $\mathbf{u}$ is an averaged velocity in the system.

To derive the generalized diffusion equation by the EnVarA, one should introduce a Lagrangian description of the system. Given a velocity field $\mathbf{u}(\boldsymbol{x},t)$, one can define a flow map $\boldsymbol{x}(\mathbf{X},t)$ through

$$\begin{cases} \dfrac{\mathrm{d}}{\mathrm{d}t}\boldsymbol{x}(\mathbf{X},t) = \mathbf{u}(\boldsymbol{x}(\mathbf{X},t),t), \\ \boldsymbol{x}(\mathbf{X},0) = \mathbf{X}, \end{cases}$$

where $\mathbf{X}\in\Omega_0$ is the Lagrangian coordinates and $\boldsymbol{x}\in\Omega_t$ is the Eulerian coordinates. Here Ω_0 is the initial configuration of the system, and Ω_t is the configuration of the system at time t. For a fixed $\mathbf{X}$, $\boldsymbol{x}(\mathbf{X},t)$ describes the trajectory of a particle (or a material point) labeled by $\mathbf{X}$; while for a fixed t, $\boldsymbol{x}(\mathbf{X},t)$ is a diffeomorphism from Ω_0 to Ω_t (See Figure 2.1 for the illustration). The existence of the flow map $\boldsymbol{x}(\mathbf{X},t)$ requires a certain regularity of $\mathbf{u}(\boldsymbol{x},t)$, for instance, being Lipschitz in $\boldsymbol{x}$.

In a Lagrangian picture, the evolution of the density function $\rho(\boldsymbol{x},t)$ is determined by the evolution of the $\boldsymbol{x}(\mathbf{X},t)$ through the kinematics relation (2.2) (written in the

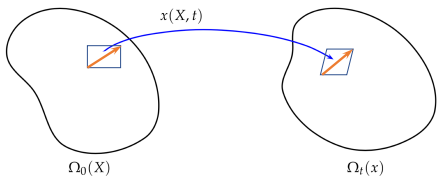


FIG. 2.1. *Schematic illustration of a flow map* $x(\mathbf{X},t)$.

Lagrangian frame of reference). More precisely, one can define the deformation tensor associated with the flow map $\boldsymbol{x}(\mathbf{X},t)$ by

$$\tilde{\mathsf{F}}(\boldsymbol{x}(\mathbf{X},t),t)=\mathsf{F}(\mathbf{X},t)=\nabla_{\mathbf{X}}\boldsymbol{x}(\mathbf{X},t). \tag{2.3}$$

Without ambiguity, we do not distinguish F and $\tilde{\mathsf{F}}$ in the rest of this paper. Let $\rho_0(\mathbf{X})$ be the initial density, then the mass conservation indicates that

$$\rho(\boldsymbol{x}(\mathbf{X},t),t)=\rho_0(\mathbf{X})/\det\mathsf{F}(\mathbf{X},t)\quad\forall\mathbf{X}\in\Omega_0 \tag{2.4}$$

in the Lagrangian frame of reference. This is equivalent to $\rho_t+\nabla\cdot(\rho\mathbf{u})=0$ in the Eulerian frame.

Within (2.4), one can rewrite the energy-dissipation law (1.4) in terms of the flow map $\boldsymbol{x}(\mathbf{X},t)$ and its velocity $\boldsymbol{x}_t(\mathbf{X},t)$ in the reference domain Ω_0. A typical form of the free energy $\mathcal{F}[\rho]$ is given by

$$\mathcal{F}[\rho]=\int_\Omega \omega(\rho)+\rho V(\boldsymbol{x})+\frac{1}{2}\left(\int_\Omega K(\boldsymbol{x},\boldsymbol{y})\rho(\boldsymbol{x})\rho(\boldsymbol{y})\mathrm{d}\boldsymbol{y}\right)\mathrm{d}\boldsymbol{x}, \tag{2.5}$$

where $V(\boldsymbol{x})$ is a potential field, and $K(\boldsymbol{x},\boldsymbol{y})$ is a symmetric nonlocal interaction kernel. In Lagrangian coordinates, the free energy (2.5) and the dissipation in (1.4) become

$$\mathcal{F}[\boldsymbol{x}(\mathbf{X},t)]=\int_{\Omega_0}\omega\left(\frac{\rho_0}{\det\mathsf{F}}\right)\det\mathsf{F}+V(\boldsymbol{x})\rho_0(\mathbf{X})+\frac{\rho_0(\mathbf{X})}{2}\left(\int_{\Omega_0}K(\boldsymbol{x},\boldsymbol{y})\rho_0(\mathbf{X})\mathrm{d}\mathbf{X}\right)\mathrm{d}\mathbf{X}$$

and

$$\mathcal{D}(\boldsymbol{x},\boldsymbol{x}_t)=\frac{1}{2}\int_{\Omega_0}\eta(\rho_0/\det\mathsf{F})|\boldsymbol{x}_t|^2\det\mathsf{F}\mathrm{d}\mathbf{X},$$

By a direct computation, the force balance equation (1.2) can be written as (see [24] for a detailed computation)

$$\eta(\rho)\mathbf{u}=-\rho\nabla\mu,\quad \mu=\omega'(\rho)+V(\boldsymbol{x})+K*\rho. \tag{2.6}$$

In Eulerian coordinates. Combining (2.6) with the kinematics equation (2.2), one obtains a generalized diffusion equation

$$\rho_t=\nabla\cdot(m(\rho)\nabla\mu),\quad m(\rho)=\rho^2/\eta(\rho), \tag{2.7}$$

where $m(\rho)$ is known as the mobility in physics. Many PDEs in a wide range of applications, including the porous medium equations (PME) [71], nonlinear Fokker–Planck equations [35], Cahn–Hilliard equations [50], Keller–Segel equations [36], and Poisson–Nernst–Planck (PNP) equations [22], can be obtained by choosing $\mathcal{F}[\rho]$ and $\eta(\rho)$ differently.

The velocity equation (2.6) can be viewed as the equation of the flow map $\boldsymbol{x}(\mathbf{X},t)$ in Lagrangian coordinates, i.e.,

$$\eta\left(\frac{\rho_0(\mathbf{X})}{\det\mathsf{F}(\mathbf{X})}\right)\frac{\mathrm{d}}{\mathrm{d}t}\boldsymbol{x}(\mathbf{X},t)=-\frac{\rho_0(\mathbf{X})}{\det\mathsf{F}(\mathbf{X})}\mathsf{F}^{-\mathrm{T}}\nabla_{\mathbf{X}}\mu\left(\frac{\rho_0(\mathbf{X})}{\det\mathsf{F}(\mathbf{X})}\right), \tag{2.8}$$

where $\mathsf{F}(\mathbf{X})=\nabla_{\mathbf{X}}\boldsymbol{x}(\mathbf{X},t)$. The flow map equation (2.8) is a highly nonlinear equation of $\boldsymbol{x}(\mathbf{X},t)$ that involves both F and $\det\mathsf{F}$. It is rather unclear how to introduce a suitable spatial discretization to (2.8) by looking at the equation. However, a

generalized diffusion can be viewed as an L^2-gradient flow of the flow map in the space of diffeomorphism. This perspective gives rise to a natural discretization of generalized diffusions [50].

Remark 2.1. In the case that $\eta(\rho)=\rho$, a generalized diffusion can be viewed as a Wasserstein gradient flow in the space of all probability densities having finite second moments $\mathcal{P}_2(\Omega)$ [35]. Formally, the Wasserstein gradient flow can be defined as a continuous time limit ($\tau\to 0$) from a semidiscrete Jordan-Kinderleherer-Otto (JKO) scheme,

$$\rho^{n+1}=\underset{\rho\in\mathcal{P}_2(\Omega)}{\arg\min}\frac{1}{2\tau}W_2(\rho,\rho^n)^2+\mathcal{F}[\rho],\quad n=0,1,2\ldots, \tag{2.9}$$

where $\mathcal{P}_2(\Omega)=\left\{\rho:\Omega\to[0,\infty)\mid \int_\Omega \rho\,\mathrm{d}\boldsymbol{x}=1,\ \int_\Omega|\boldsymbol{x}|^2\rho(\boldsymbol{x})\mathrm{d}\boldsymbol{x}<\infty\right\}$ and $W_2(\rho,\rho^n)$ is the Wasserstein distance between ρ and ρ^n. The Wasserstein gradient flow is an Eulerian description [6]. Other choices of dissipation can define other metrics in the space of probability measures [1, 46].

2.2. Neural-network-based numerical schemes for PDEs. In this subsection, we briefly review some existing neural network-based algorithms for solving PDEs. We refer interested readers to [18, 52] for detailed reviews.

Considering a PDE subject to a certain boundary condition

$$\mathcal{L}\varphi(\boldsymbol{x})=f(\boldsymbol{x}),\quad \boldsymbol{x}\in\Omega\subset\mathbb{R}^d,\quad \mathcal{B}\varphi(\boldsymbol{x})=g(\boldsymbol{x}),\quad \boldsymbol{x}\in\partial\Omega, \tag{2.10}$$

and assuming the solution can be approximated by a neural network, denoted by $\varphi_{\mathrm{NN}}(\boldsymbol{x};\boldsymbol{\Theta})$, where $\boldsymbol{\Theta}$ is the set of parameters in the neural network, the PINN [62] and similar methods [13, 68, 77] find the optimal parameter $\boldsymbol{\Theta}^*$ by minimizing a loss function defined as

$$L(\boldsymbol{\Theta})=\frac{1}{N_{in}}\sum_{i=1}^{N_{in}}\left(\mathcal{L}\varphi_{\mathrm{NN}}(\boldsymbol{x}_i,\boldsymbol{\Theta})-f(\boldsymbol{x}_i)\right)^2+\frac{\lambda}{N_b}\sum_{j=1}^{N_b}(\mathcal{B}\varphi_{\mathrm{NN}}(\boldsymbol{s}_j;\boldsymbol{\Theta})-g(\boldsymbol{s}_j))^2. \tag{2.11}$$

Here $\{\boldsymbol{x}_i\}_{i=1}^{N_{in}}$ and $\{\boldsymbol{s}_j\}_{j=1}^{N_b}$ are sets of samples in Ω and $\partial\Omega$, respectively, which can be drawn uniformly or by following some prescribed distributions. The parameter λ is used to weigh the sum in the loss function. Minimizers of the loss function (2.11) can be obtained by using some optimization methods, such as AdaGrad and Adam. Of course, the objective function of this minimization problem, in general, is not convex even when the initial problem is. Obtaining the global minimum of (2.11) is highly nontrivial.

In contrast to the PINN, which is based on the strong form of a PDE, the DRM [21] is designed for solving PDEs using their variational formulations. If (2.10) is an Euler–Lagrangian equation of some energy functional

$$I[\varphi,\nabla\varphi]=\int_\Omega W(\varphi(\boldsymbol{x}),\nabla_{\boldsymbol{x}}\varphi(\boldsymbol{x}))\mathrm{d}\boldsymbol{x}+\lambda\int_{\partial\Omega}|\mathcal{B}\varphi-g(\boldsymbol{x})|^2\mathrm{d}S, \tag{2.12}$$

then the loss function can be defined directly by

$$L(\theta)=\frac{1}{N_{in}}\sum_{i=1}^{N_{in}}W(\varphi_{\mathrm{NN}}(\boldsymbol{x}_i),\nabla_{\boldsymbol{x}}\varphi_{\mathrm{NN}}(\boldsymbol{x}_i))+\frac{\lambda}{N_b}\sum_{j=1}^{N_b}(\mathcal{B}\varphi_{\mathrm{NN}}(\boldsymbol{s}_j)-g(\boldsymbol{s}_j))^2. \tag{2.13}$$

The last term in (2.12) is a penalty term for the boundary condition. The original Dirichlet boundary condition can be recovered with $\lambda\to\infty$. Again, the samples

$\{\boldsymbol{x}_i\}_{i=1}^{N_{in}}$ and $\{\boldsymbol{s}_j\}_{j=1}^{N_b}$ can be uniformly sampled in Ω and $\partial\Omega$ or sampled following some other prescribed distributions, respectively. Additionally, both the variational PINN [37] and the WAN [84] utilize the Galerkin formulation to solve PDEs. The variational PINN [37] stems from the Petrov–Galerkin method, represents the solution via a DNN, and keeps test functions belonging to linear function spaces. The WAN employs the primal and adversarial networks to parameterize the weak solution and test functions, respectively [84].

The neural network-based algorithms mentioned above focus on elliptic equations. For an evolution equation of the form

$$\begin{cases} \partial_t \varphi(\boldsymbol{x},t) = F(t,\boldsymbol{x},\varphi), & (\boldsymbol{x},t) \in \Omega \times (0,\infty), \\ \varphi(\boldsymbol{x},0) = \varphi_0(\boldsymbol{x}), & \boldsymbol{x} \in \Omega \end{cases} \tag{2.14}$$

with a suitable boundary condition, the predominant approach is to treat the time variable as an additional dimension, and the PINN type techniques can be applied. However, these approaches are often expensive and may fail to capture the dynamics of these systems. Unlike spatial variables, there exists an inherent order in time, where the solution at time T is determined by the solutions in $t < T$. It is difficult to generate samples in time to maintain this causality. Very recently, new efforts have been made in using neural networks to solve evolution PDEs [8, 16]. The idea of these approaches is to use neural networks with time-dependent parameters to represent the solutions. For instance, Neural Galerkin method, proposed in [8], parameterizes the solution as $\varphi_h(\boldsymbol{x};\boldsymbol{\Theta}(t))$ with $\boldsymbol{\Theta}(t)$ being the neural network parameters, and defines a loss function in terms of $\boldsymbol{\Theta}$ and $\dot{\boldsymbol{\Theta}}$ through a residual function, i.e.,

$$J(\boldsymbol{\Theta},\boldsymbol{\eta}) = \frac{1}{2}\int |\nabla_{\boldsymbol{\Theta}}\varphi_h \cdot \boldsymbol{\eta} - F(\boldsymbol{x},t,\varphi_h(\boldsymbol{x};\boldsymbol{\Theta}))|^2 \mathrm{d}\nu_\Theta(\boldsymbol{x}), \tag{2.15}$$

where $\nu_\Theta(\boldsymbol{x})$ is a suitable measure which might depend on $\boldsymbol{\Theta}$. By taking variation of $J(\boldsymbol{\Theta},\boldsymbol{\eta})$ with respect to $\boldsymbol{\eta}$, the neural Galerkin method arrives at an ODE of $\boldsymbol{\Theta}(t)$:

$$\mathsf{M}(\boldsymbol{\Theta})\dot{\boldsymbol{\Theta}} = F(t,\boldsymbol{\Theta}), \quad \boldsymbol{\Theta}(0) = \boldsymbol{\Theta}_0, \tag{2.16}$$

where $\mathsf{M}(\boldsymbol{\Theta}) = \int \nabla_{\boldsymbol{\Theta}}\varphi_h \otimes \nabla_{\boldsymbol{\Theta}}\varphi_h \mathrm{d}\nu_{\boldsymbol{\Theta}}(\boldsymbol{x})$, and $F(t,\boldsymbol{\Theta}) = \int \nabla_{\boldsymbol{\Theta}}\varphi_h F(\boldsymbol{x},t,\varphi_h)\mathrm{d}\nu_{\boldsymbol{\Theta}}(\boldsymbol{x})$. The ODE (2.16) can be solved by standard explicit or implicit time-marching schemes.

It's important to note that, with the exception of DRM, all existing neural-network-based methods are developed based on either the strong or weak forms of PDEs. While DRM utilizes the free energy functional, it is only suitable for solving static problems, i.e., finding the equilibrium of the system. Additionally, most of these existing approaches are Eulerian methods. For certain types of PDEs, like generalized diffusions, these methods may fail to preserve physical constraints, such as the positivity and conservation of mass of a probability function.

3. Energetic variational neural network.

In this section, we present the structure-preserving EVNN discretization for solving both L^2-gradient flows and generalized diffusions. As mentioned earlier, the goal is to construct a neural-network discretization based on the energy-dissipation law, without working on the underlying PDE. One can view our method as a generalization of the DRM to evolution equations.

3.1. EVNN scheme for L^2-gradient flows.

Before we discuss the neural network discretization, we first briefly review the discrete energetic variational approach proposed in [50]. Given a continuous energy-dissipation law (1.1), the discrete

energetic variational approach first constructs a finite-dimensional approximation to this law by introducing a spatial discretization of the state variable φ, denoted by $\varphi_h(\boldsymbol{x};\boldsymbol{\Theta}(t))$, where $\boldsymbol{\Theta}(t)\in\mathbb{R}^K$ is the parameter to be determined. By replacing φ by φ_h, one can obtain a semidiscrete energy-dissipation law (here we assume $\mathcal{K}=0$ in the continuous model for simplicity) in terms of $\boldsymbol{\Theta}$:

$$\frac{\mathrm{d}}{\mathrm{d}t}\mathcal{F}_h\left[\boldsymbol{\Theta}(t)\right]=-\triangle_h[\boldsymbol{\Theta}(t),\boldsymbol{\Theta}'(t)], \tag{3.1}$$

where $\mathcal{F}_h\left[\boldsymbol{\Theta}(t)\right]=\mathcal{F}[\varphi_h(\boldsymbol{x};\boldsymbol{\Theta})]$, and $\triangle_h[\boldsymbol{\Theta}(t),\boldsymbol{\Theta}'(t)]=\int\eta(\varphi_h)|\nabla_{\boldsymbol{\Theta}}\varphi_h\cdot\boldsymbol{\Theta}'(t)|^2\mathrm{d}\boldsymbol{x}$. For example, in Galerkin methods, one can let $\varphi_h(\boldsymbol{x},t)$ be $\varphi_h(\boldsymbol{x},t)=\sum_{i=1}^K\gamma_i(t)\psi_i(\boldsymbol{x})$ with $\{\psi_i\}_{i=1}^K$ being the set of basis functions. Then $\boldsymbol{\Theta}(t)$ is a vector given by $\boldsymbol{\Theta}(t)=(\gamma_1(t),\gamma_2(t),\ldots,\gamma_K(t))^{\mathrm{T}}\in\mathbb{R}^K$.

By employing the energetic variational approach in the semidiscrete level (3.1), one can obtain an ODE system of $\boldsymbol{\Theta}(t)$. Particularly, in the linear response regime, $\mathcal{D}_h(\boldsymbol{\Theta}(t),\boldsymbol{\Theta}'(t))=\frac{1}{2}\int\eta(\varphi_h)|\nabla_{\boldsymbol{\Theta}}\varphi_h\cdot\boldsymbol{\Theta}'(t)|^2\mathrm{d}\boldsymbol{x}$ is a quadratic function of $\boldsymbol{\Theta}'$. The ODE system of $\boldsymbol{\Theta}(t)$ can then be written as

$$\mathsf{D}\left(\boldsymbol{\Theta}\right)\boldsymbol{\Theta}'(t)=-\frac{\delta\mathcal{F}_h}{\delta\boldsymbol{\Theta}}, \tag{3.2}$$

where

$$\frac{\delta\mathcal{F}_h}{\delta\boldsymbol{\Theta}}=\frac{\delta\mathcal{F}}{\delta\varphi}\nabla_{\boldsymbol{\Theta}}\varphi_h,\quad \mathsf{D}\left(\boldsymbol{\Theta}\right)=\int\eta(\varphi_h)(\nabla_{\boldsymbol{\Theta}}\varphi_h\otimes\nabla_{\boldsymbol{\Theta}}\varphi_h)\mathrm{d}\boldsymbol{x}. \tag{3.3}$$

The ODE (3.2) is the same as the ODE (2.16) in the Neural Galerkin method [8] for L^2 gradient flows, although the derivation is different.

Since (3.2) is a finite-dimensional gradient flow, one can then construct a minimizing movement scheme for $\boldsymbol{\Theta}$ [25]: finding $\boldsymbol{\Theta}^{n+1}$ such that

$$\boldsymbol{\Theta}^{n+1}=\underset{\boldsymbol{\Theta}\in\mathcal{S}_{ad}^h}{\arg\min}\,J_n(\boldsymbol{\Theta}),\quad J_n(\boldsymbol{\Theta})=\frac{(\mathsf{D}_*^n(\boldsymbol{\Theta}-\boldsymbol{\Theta}^n))\cdot(\boldsymbol{\Theta}-\boldsymbol{\Theta}^n)}{2\tau}+\mathcal{F}_h(\boldsymbol{\Theta}). \tag{3.4}$$

Here $\mathcal{S}_{ad}^h$ is the admissible set of $\boldsymbol{\Theta}$ inherited from the admissible set of φ, denoted by $\mathcal{S}$, and D_*^n is a constant matrix. A typical choice is $\mathsf{D}_*^n=\mathsf{D}(\boldsymbol{\Theta}^n)$. An advantage of this scheme is that

$$\mathcal{F}_h(\boldsymbol{\Theta}^{n+1})\le J_h^n(\boldsymbol{\Theta}^{n+1})\le J_h^n(\boldsymbol{\Theta}^n)=\mathcal{F}_h(\boldsymbol{\Theta}^n), \tag{3.5}$$

if D_*^n is positive definite, which guarantees the energy stability with respect to the discrete free energy $\mathcal{F}_h(\boldsymbol{\Theta})$. Moreover, by choosing a proper optimization method, we can assure that φ_h^{n+1} stays in the admissible set $\mathcal{S}$.

Although neural networks can be used to construct $\varphi_h(\boldsymbol{x};\boldsymbol{\Theta}(t))$, it might be expensive to compute $\nabla_{\boldsymbol{\Theta}}\varphi_h\otimes\nabla_{\boldsymbol{\Theta}}\varphi_h$ in $\mathsf{D}(\boldsymbol{\Theta})$. Moreover, $\mathsf{D}(\boldsymbol{\Theta})$ is not a sparse matrix and requires a lot of computer memory to store when a deep neural network is used.

To overcome these difficulties, we propose an alternative approach by introducing temporal discretization before spatial discretization. Let τ be the time step size. For the L^2-gradient flow (2.1), given φ^n, which represents the numerical solution at $t^n=n\tau$, one can obtain φ^{n+1} by solving the following optimization problem: finding φ^{n+1} in some admissible set $\mathcal{S}$ such that

$$\varphi^{n+1}=\underset{\varphi\in\mathcal{S}}{\arg\min}\,J^n(\varphi),\quad J^n(\varphi)=\frac{1}{2\tau}\int\eta(\varphi^n)|\varphi-\varphi^n|^2\mathrm{d}\boldsymbol{x}+\mathcal{F}[\varphi]. \tag{3.6}$$

Let $\varphi_h(\boldsymbol{x};\boldsymbol{\Theta})$ be a finite-dimensional approximation of φ with $\boldsymbol{\Theta}\in\mathbb{R}^K$ being the parameter of the spatial discretization (e.g., weights of linear combination in a Galerkin approximation) yet to be determined, then the minimizing movement scheme (3.6) can be written in terms of $\boldsymbol{\Theta}$: finding $\boldsymbol{\Theta}^{n+1}$ such that

$$\boldsymbol{\Theta}^{n+1}=\arg\min_{\boldsymbol{\Theta}\in\mathcal{S}_h} J_h^n(\boldsymbol{\Theta}),\quad J_h^n(\boldsymbol{\Theta})=\frac{1}{2\tau}\int \eta^n|\varphi_h(\boldsymbol{x};\boldsymbol{\Theta})-\varphi_h(\boldsymbol{x};\boldsymbol{\Theta}^n)|^2\mathrm{d}\boldsymbol{x}+\mathcal{F}_h[\boldsymbol{\Theta}]. \tag{3.7}$$

Here $\boldsymbol{\Theta}^n$ is the value of $\boldsymbol{\Theta}$ at time t^n, $\mathcal{F}_h[\boldsymbol{\Theta}]=\mathcal{F}[\varphi_h(\boldsymbol{x};\boldsymbol{\Theta})]$, and $\eta^n=\eta(\varphi_h(\boldsymbol{x};\boldsymbol{\Theta}^n))$.

Remark 3.1. The connection between the minimizing movement scheme (3.7), derived by a temporal-then-spatial approach, and the minimizing movement scheme (3.4), derived by a spatial-then-temporal approach, can be shown with a direct calculation. Indeed, according to the first-order necessary condition for optimality, an optimal solution of the minimization problem (3.7) $\boldsymbol{\Theta}^{n+1}$ satisfies

$$\begin{aligned}\frac{\delta J_h^n(\boldsymbol{\Theta})}{\delta\boldsymbol{\Theta}}\Big|_{\boldsymbol{\Theta}^{n+1}}&=\frac{1}{\tau}\int\eta^n(\varphi_h(\boldsymbol{x};\boldsymbol{\Theta}^{n+1})-\varphi_h(\boldsymbol{x};\boldsymbol{\Theta}^n))\nabla_{\boldsymbol{\Theta}}\varphi_h\Big|_{\boldsymbol{\Theta}^{n+1}}\mathrm{d}\boldsymbol{x}+\frac{\delta\mathcal{F}_h}{\delta\boldsymbol{\Theta}}\Big|_{\boldsymbol{\Theta}^{n+1}}\\&=\int\eta^n\nabla_{\boldsymbol{\Theta}}\varphi_h\Big|_{\boldsymbol{\Theta}^*}\otimes\nabla_{\boldsymbol{\Theta}}\varphi_h\Big|_{\boldsymbol{\Theta}^{n+1}}\mathrm{d}\boldsymbol{x}\frac{\boldsymbol{\Theta}^{n+1}-\boldsymbol{\Theta}^n}{\tau}+\frac{\delta\mathcal{F}_h}{\delta\boldsymbol{\Theta}}\Big|_{\boldsymbol{\Theta}^{n+1}}=0\end{aligned} \tag{3.8}$$

for some $\boldsymbol{\Theta}^*$, where the second equality follows the mean value theorem. In the case of Galerkin methods, as $\nabla_{\boldsymbol{\Theta}}\varphi_h$ is independent on $\boldsymbol{\Theta}$, $\boldsymbol{\Theta}^{n+1}$ is a solution to an implicit Euler scheme for the ODE (3.2) in which $\eta(\varphi_h)$ is treated explicitly.

By choosing a certain neural network to approximate φ, denoted by $\varphi_{\mathrm{NN}}(\boldsymbol{x};\boldsymbol{\Theta})$ ($\boldsymbol{\Theta}$ is used to denote all parameters in the neural network), we can summarize the EVNN scheme for solving L^2-gradient flows in Algorithm 3.1.

It can be noticed that both (3.9) and (3.10) involve integration in the computational domain Ω. This integration is often computed by using a grid-based numerical quadrature or Monte–Carlo/Quasi–Monte–Carlo algorithms [65]. It is worth mentioning that due to the nonconvex nature of the optimization problem and the error in estimating the integration, it might be difficult to find an optimal $\boldsymbol{\Theta}^{n+1}$ at each step. But since we start the optimization procedure with $\boldsymbol{\Theta}^n$, we'll always be able to get a $\boldsymbol{\Theta}^{n+1}$ that lowers the discrete free energy at least on the training set. In practice, the optimization problem can be solved by either deterministic optimization algorithms, such as L-BFGS and gradient descent with Barzilai–Borwein step-size, or stochastic gradient descent algorithms, such as AdaGrad and Adam.

Algorithm 3.1 Numerical algorithm for solving L^2-gradient flows.

For a given initial condition $\varphi_0(\boldsymbol{x})$, compute $\boldsymbol{\Theta}^0$ by solving

$$\boldsymbol{\Theta}^0=\arg\min_{\boldsymbol{\Theta}}\int_\Omega|\varphi_{\mathrm{NN}}(\boldsymbol{x};\boldsymbol{\Theta})-\varphi_0(\boldsymbol{x})|^2\mathrm{d}\boldsymbol{x}. \tag{3.9}$$

At each step, update $\boldsymbol{\Theta}^{n+1}$ by solving the optimization problem

$$\boldsymbol{\Theta}^{n+1}=\arg\min_{\boldsymbol{\Theta}}\left(\frac{1}{2\tau}\int_\Omega\eta^n|\varphi_{\mathrm{NN}}(\boldsymbol{x};\boldsymbol{\Theta})-\varphi_{\mathrm{NN}}(\boldsymbol{x},\boldsymbol{\Theta}^n)|^2\mathrm{d}\boldsymbol{x}+\mathcal{F}[\varphi_{\mathrm{NN}}(\boldsymbol{x};\boldsymbol{\Theta})]\right). \tag{3.10}$$

We have $\varphi_{\mathrm{NN}}(\boldsymbol{x};\boldsymbol{\Theta}^n)$ as a numerical solution at time $t^n=n\tau$.

Remark 3.2. It is straightforward to incorporate other variational high-order temporal discretizations to solve the L^2-gradient flows. For example, a second-order accurate BDF2 scheme can be reformulated as an optimization problem

$$\begin{aligned}\boldsymbol{\Theta}^n = \arg\min_{\boldsymbol{\Theta}} \Bigg(& \frac{\eta}{\tau}\int_\Omega |\varphi_{\mathrm{NN}}(\boldsymbol{x};\boldsymbol{\Theta}) - \varphi_{\mathrm{NN}}(\boldsymbol{x},\boldsymbol{\Theta}^{n-1})|^2 \mathrm{d}\boldsymbol{x} \\ & - \frac{\eta}{4\tau}\int_\Omega |\varphi_{\mathrm{NN}}(\boldsymbol{x};\boldsymbol{\Theta}) - \varphi_{\mathrm{NN}}(\boldsymbol{x},\boldsymbol{\Theta}^{n-2})|^2 \mathrm{d}\boldsymbol{x} + \mathcal{F}[\varphi_{\mathrm{NN}}(\boldsymbol{x};\boldsymbol{\Theta})]\Bigg).\end{aligned} \tag{3.11}$$

A modified Crank–Nicolson time-marching scheme can be reformulated as

$$\begin{aligned}\boldsymbol{\Theta}^n = \arg\min_{\boldsymbol{\Theta}} \Bigg(& \frac{\eta}{\tau}\int_\Omega |\varphi_{\mathrm{NN}}(\boldsymbol{x};\boldsymbol{\Theta}) - \varphi_{\mathrm{NN}}(\boldsymbol{x},\boldsymbol{\Theta}^{n-1})|^2 \mathrm{d}\boldsymbol{x} + \mathcal{F}[\varphi_{\mathrm{NN}}(\boldsymbol{x};\boldsymbol{\Theta})] \\ & + \langle \nabla_\Theta \mathcal{F}[\varphi_{\mathrm{NN}}(\boldsymbol{x};\boldsymbol{\Theta}^{n-1})], \boldsymbol{\Theta} - \boldsymbol{\Theta}^{n-1}\rangle\Bigg).\end{aligned} \tag{3.12}$$

Here we assume that η is a constant for simplicity.

3.2. Lagrangian EVNN scheme for generalized diffusions. In this subsection, we show how to formulate an EVNN scheme in the Lagrangian frame of reference for generalized diffusions.

As discussed previously, a generalized diffusion can be viewed as an L^2-gradient flow of the flow map $\boldsymbol{x}(\mathbf{X},t)$ in the space of diffeomorphisms. Hence, the EVNN scheme for a generalized diffusion can be formulated in terms of a minimizing movement scheme of the flow map given by

$$\Phi^{n+1} = \arg\min_{\Phi\in\mathrm{Diff}} \frac{1}{2\tau}\int |\Phi(\mathbf{X}) - \Phi^n(\mathbf{X})|^2 \rho_0(\mathbf{X})\mathrm{d}\mathbf{X} + \mathcal{F}\left[\Phi_{\#}\rho_0\right], \tag{3.13}$$

where $\Phi^n(\mathbf{X})$ is a numerical approximation of the flow map $\boldsymbol{x}(\mathbf{X},t)$ at $t^n = n\tau$, $\rho_0(\mathbf{X})$ is the initial density, $\mathcal{F}[\rho]$ is the free energy for the generalized diffusion defined in (2.5), and

$$(\Phi_{\#}\rho_0)(\boldsymbol{x}) := \frac{\rho_0(\Phi^{-1}(\boldsymbol{x}))}{\det \mathsf{F}(\Phi^{-1}(\boldsymbol{x}))}, \quad \mathrm{Diff} = \{\Phi : \mathbb{R}^d \to \mathbb{R}^d \mid \Phi \text{ is a diffeomorphism}\}.$$

One can parameterize $\Phi : \mathbb{R}^d \to \mathbb{R}^d$ by a suitable neural network. The remaining procedure is nearly identical to that of the previous subsection. However, it is often difficult to solve this optimization problem directly, and one might need to build a large neural network to approximate Φ^{n+1} when n is large.

To overcome this difficulty, we propose an alternative approach. Instead of seeking an optimal map Φ^{n+1} between ρ^0 and ρ^{n+1}, we seek an optimal map Ψ^{n+1} between ρ^n and ρ^{n+1}. More precisely. we assume that

$$\Phi^{n+1} = \Psi^{n+1} \circ \Psi^n \circ \Psi^{n-1} \cdots \circ \Psi^1.$$

Given ρ^n, one can compute Ψ^{n+1} by solving the following optimization problem:

$$\Psi^{n+1} = \arg\min_{\Psi\in\mathrm{Diff}} \frac{1}{2\tau}\int |\Psi(\boldsymbol{x}) - \boldsymbol{x}|^2 \rho^n(\boldsymbol{x})\mathrm{d}\boldsymbol{x} + \mathcal{F}[\Psi_{\#}\rho^n]. \tag{3.14}$$

The corresponding ρ^{n+1} can then be computed through $\rho^{n+1}(\boldsymbol{x}) = (\Psi_{\#}^{n+1}\rho^n)(\boldsymbol{x})$. An advantage of this approach is that we only need a small size neural network to approximate Ψ^{n+1} at each time step when τ is small.

Remark 3.3. The scheme (3.14) can be viewed as a Lagrangian realization of the JKO scheme (2.9) for the Wasserstein gradient flow, although it is developed based on the L^2-gradient flow structure in the space of diffeomorphism. According to the Benamou–Brenier formulation [5], the Wasserstein distance between two probability densities ρ_1 and ρ_2 can be computed by solving the optimization problem

$$W_2(\rho_1,\rho_2)^2 = \min_{(\rho,\mathbf{u})\in\mathcal{S}} \int_0^1 \int \rho|\mathbf{u}|^2 \mathrm{d}\boldsymbol{x}\mathrm{d}t, \tag{3.15}$$

where the admissible set of $(\rho,\mathbf{u})$ is given by

$$\mathcal{S} = \{(\rho,\mathbf{u}) \mid \rho_t + \nabla\cdot(\rho\mathbf{u}) = 0, \quad \rho(\boldsymbol{x},0) = \rho_1, \quad \rho(\boldsymbol{x},1) = \rho_2\}. \tag{3.16}$$

Hence, one can solve the JKO scheme by solving

$$\begin{aligned} &(\mathbf{u}^*,\hat{\rho}^*) = \arg\min_{(\mathbf{u},\hat{\rho})} \frac{1}{2}\int_0^\tau \int_\Omega \hat{\rho}|\mathbf{u}|^2 \mathrm{d}\boldsymbol{x}\mathrm{d}t + \mathcal{F}[\hat{\rho}(\tau)] \\ &\text{s.t.} \quad \partial_t\hat{\rho} + \nabla\cdot(\hat{\rho}\mathbf{u}) = 0, \quad \hat{\rho}(0) = \rho^n. \end{aligned} \tag{3.17}$$

and letting $\rho^{n+1} = \hat{\rho}^*(\tau)$. In Lagrangian coordinates, (3.17) is equivalent to

$$\boldsymbol{x}^*(\mathbf{X},t) = \arg\min_{\boldsymbol{x}(\mathbf{X},t)} \frac{1}{2}\int_0^\tau \int_\Omega \rho_0(\mathbf{X})|\boldsymbol{x}_t(\mathbf{X},t)|^2 \mathrm{d}\mathbf{X}\mathrm{d}t + \mathcal{F}(\hat{\rho}(\boldsymbol{x},\tau)), \tag{3.18}$$

where $\rho_0 = \rho^n$ and $\hat{\rho}(\boldsymbol{x}(\mathbf{X},\tau),\tau) = \rho_0(\mathbf{X})/\det\mathsf{F}(\mathbf{X},\tau)$. By taking variation of (3.18) with respect to $\boldsymbol{x}(\mathbf{X},t)$, one can show that the optimal condition is $\boldsymbol{x}_{tt}(\mathbf{X},t) = 0$ for $t\in(0,\tau)$, which indicates that $\boldsymbol{x}(\mathbf{X},t) = t(\Psi(\mathbf{X}) - X)/\tau + X$ if $\boldsymbol{x}(\mathbf{X},\tau) = \Psi(\mathbf{X})$. Hence, if Ψ^* is the optimal solution of (3.14), then $\boldsymbol{x}^*(\mathbf{X},t) = t(\Psi^*(\mathbf{X}) - \mathbf{X})/\tau + \mathbf{X}$ is the optimal solution of (3.18).

Remark 3.4. If $\eta(\rho)\neq\rho$, we can formulate the optimization problem (3.14) as

$$\Psi^{n+1} = \arg\min_{\Psi\in\mathrm{Diff}} \frac{1}{2\tau}\int |\Psi(\boldsymbol{x}) - \boldsymbol{x}|^2 \eta(\rho^n(\boldsymbol{x}))\mathrm{d}\boldsymbol{x} + \mathcal{F}[\Psi_{\#}\rho^n]. \tag{3.19}$$

by treating η explicitly. A subtle fact is that $\det\mathsf{F}^n = 1$ since we always start with an identity map.

The numerical algorithm for solving generalized diffusions is summarized in Algorithm 3.2.

Algorithm 3.2 Numerical algorithm for solving generalized diffusions.

- Given $\{\boldsymbol{x}_i^n\}_{i=1}^N$ and the densities ρ_i^n at $t = n\tau$ and $\boldsymbol{x}_i^n$.
- Find $\Psi^{n+1}(\boldsymbol{x}) : \mathbb{R}^d \to \mathbb{R}^d$, by solving the optimization problem (3.14). To guarantee energy stability, we should take Ψ as an approximation to an identity map initially when solving the optimization problem (3.14).
- After obtaining Ψ^{n+1}, update $\{\boldsymbol{x}_i^{n+1}\}_{i=1}^N$ and ρ^{n+1} by

$$\boldsymbol{x}_i^{n+1} = \Psi^{n+1}(\boldsymbol{x}_i^n), \quad \rho_i^{n+1} = \frac{\rho_i^n}{\det(\nabla\Psi^{n+1}(\boldsymbol{x}_i^n))}. \tag{3.20}$$

The next question is how to accurately evaluate the numerical integration in (3.14). Let $\boldsymbol{x}_i^n = \Phi^n(\boldsymbol{x}_i^0)$, for the general free energy (2.5). One way to evaluate the integrations in (3.14) is using

(3.21)
$$J(\Psi) = \frac{1}{2\tau}\sum_{i=1}^N \rho_i^n \|\Psi(\boldsymbol{x}_i^n) - \boldsymbol{x}_i^n\|^2 |\Omega_i^n| + \sum_{i=1}^N \left(\omega\left(\frac{\rho_i^n}{\det(\nabla\Psi(\boldsymbol{x}_i^n))}\right)\det(\nabla\Psi(\boldsymbol{x}_i^n))\right)|\Omega_i^n| + \sum_{i=1}^N V(\Psi(\boldsymbol{x}_i^n))\rho_i^n|\Omega_i^n| + \frac{1}{2}\sum_{i,j=1}^N K(\Psi(\boldsymbol{x}_i^n),\Psi(\boldsymbol{x}_j^n))\rho_i^n\rho_j^n|\Omega_j^n||\Omega_i^n|,$$

where $|\Omega_i^n|$ is the volume of the Voronoi cells associated with the set of points $\{\boldsymbol{x}_i^n\}$, ρ_i^n stands for $\rho^n(\boldsymbol{x}_i^n)$. Here the numerical integration is computed through a piecewisely constant reconstruction of ρ^n based on its values at $\{\Phi^n(\boldsymbol{x}_i^0)\}_{i=1}^N$. However, it is not straightforward to compute $|\Omega_i^n|$, particularly for high-dimensional cases. In the current study, we assume that the initial samples are drawn from ρ_0, one can roughly assume that $\{\boldsymbol{x}_i^n\}$ follows the distribution ρ^n, then according to the Monte–Carlo approach, the numerical integration can be evaluated as

(3.22)
$$J(\Psi) = \frac{1}{2\tau}\left(\frac{1}{N}\sum_{i=1}^N \|\Psi(\boldsymbol{x}_i^n) - \boldsymbol{x}_i^n\|^2\right) + \frac{1}{N}\sum_{i=1}^N\left(f_\omega\left(\frac{\rho_i^n}{\det(\nabla\Psi(\boldsymbol{x}_i^n))}\right) + V(\Psi(\boldsymbol{x}_i^n))\right) + \frac{1}{2N^2}\sum_{i,j=1}^N K\left(\Psi(\boldsymbol{x}_i^n),\Psi(\boldsymbol{x}_j^n)\right),$$

where $f_\omega(\rho) = \omega(\rho)/\rho$. The proposed numerical method can be further improved if one can evaluate the integration more accurately, i.e., have an efficient way to estimate $|\Omega_i^n|$. Alternatively, as an advantage of the neural network-based algorithm, we can get $\rho^n(\boldsymbol{x}) = \rho_0\tilde{\circ}(\Phi^n)^{-1}(\boldsymbol{x}) = (\rho_0\circ(\Phi^n)^{-1}(\boldsymbol{x}))/(\det(\nabla\Phi^n)|_{(\Phi^n)^{-1}(\boldsymbol{x})})\ \forall \boldsymbol{x}$, which enables us to estimate $J(\Psi)$ by resampling. More precisely, assume μ is a distribution that is easy to sample

(3.23)
$$J(\Psi) = \frac{1}{2\tau}\left(\frac{1}{N}\sum_{i=1}^N \|\Psi(\boldsymbol{x}_i^n) - \boldsymbol{x}_i^n\|^2\right)\frac{\rho_i^n}{\mu_i^n} + \frac{1}{N}\sum_{i=1}^N\left(f_\omega\left(\frac{\rho_i^n}{\det(\nabla\Psi(\boldsymbol{x}_i^n))}\right)\right)\frac{\rho_i^n}{\mu_i^n} + V(\Psi(\boldsymbol{x}_i^n))\frac{\rho_i^n}{\mu_i^n} + \frac{1}{2N^2}\sum_{i,j=1}^N K\left(\Psi(\boldsymbol{x}_i^n),\Psi(\boldsymbol{x}_j^n)\right)\frac{\rho_i^n\rho_j^n}{\mu_i^n\mu_j^n},$$

where $\boldsymbol{x}_i \sim \mu$, $\rho_i^n = \rho^n(\boldsymbol{x}_i)$. We will explore this resampling approach in future work.

The remaining question is how to parameterize a diffeomorphism using neural networks. This is discussed in detail in the next subsection.

3.3. Neural network architectures. In principle, the proposed numerical framework is independent of the choice of neural network architectures. However, different neural network architectures may lead to different numerical performances, arising from a balance of approximation (representation power), optimization, and generalization. In this subsection, we briefly discuss several neural network architectures that we use in the numerical experiments.

3.3.1. Neural network architectures for Eulerian methods. For Eulerian methods, one can construct a neural network to approximate an unknown function $f : \mathbb{R}^d \to \mathbb{R}$. Shallow neural networks (two-layer neural networks) approximate f by functions of the form

$$f(\boldsymbol{x};\boldsymbol{\Theta}) = \sum_{i=1}^{N} \alpha_i \sigma(\boldsymbol{\omega}_i \cdot \boldsymbol{x} + b_i) + \alpha_0 = \boldsymbol{\alpha} \cdot \sigma(\boldsymbol{W}\boldsymbol{x} + \boldsymbol{b}) + \alpha_0, \tag{3.24}$$

where $\sigma(\cdot) : \mathbb{R} \to \mathbb{R}$ is a fixed nonlinear activation function, N is the number of hidden nodes (neurons), and $\boldsymbol{\Theta} = \{\alpha_i, \boldsymbol{\omega}_i, b_i\}$ are the neural network parameters to be identified. Typical choices of activation functions include the ReLU $\sigma(x) = \max(x, 0)$, the sigmoid $\sigma(x) = 1/(1+e^{-2x})$ and the hyperbolic tangent function $\tanh(x)$. A DNN can be viewed as a network composed of many hidden layers. More precisely, a DNN with L hidden layers represents a function $f : \mathbb{R}^d \to \mathbb{R}$ by [67]

$$f(\boldsymbol{x};\boldsymbol{\Theta}) = g \circ T^L \circ T^{L-1} \circ \cdots \circ T^{(1)}(\boldsymbol{x}), \tag{3.25}$$

where $g(\boldsymbol{z}) = \sum_{i=1}^{N_L} \gamma_i z_i + \gamma_0$ is a linear map from $\mathbb{R}^{N_l}$ to $\mathbb{R}$, $T^{(l)}$ is a nonlinear map from $\mathbb{R}^{N_{l-1}}$ to $\mathbb{R}^{N_l}$, and $N_0 = d$. The nonlinear map T^l takes the form

$$T^{(l)}(\boldsymbol{x}_{l-1}) = \sigma(\boldsymbol{W}_l \boldsymbol{x}_{l-1} + \boldsymbol{b}_l), \tag{3.26}$$

where $\boldsymbol{W}_l \in \mathbb{R}^{N_{l-1} \times N_l}$, $\boldsymbol{b}_l \in \mathbb{R}^l$, and $\sigma(\cdot)$ is a nonlinear activation function that acts componentwisely on vector-valued inputs.

Another widely used class of DNN model is residual neural network (ResNet). A typical K-block ResNet approximates an unknown function $f(\boldsymbol{x}) : \mathbb{R}^d \to \mathbb{R}$ by

$$f_K(\boldsymbol{x};\boldsymbol{\Theta}) = g(\boldsymbol{z}_K(\boldsymbol{x})), \tag{3.27}$$

where $g(\boldsymbol{z}) = \sum_{i=1}^{N} \gamma_i z_i + \gamma_0$ is a linear map from $\mathbb{R}^N \to \mathbb{R}$ and $z_K(\boldsymbol{x}) : \mathbb{R}^d \to \mathbb{R}^N$ is a nonlinear map defined through

$$\boldsymbol{z}_0 = \boldsymbol{V}\boldsymbol{x}, \quad \boldsymbol{z}_k = \sigma_2\left(\boldsymbol{\alpha}_k T_k^{L_k} \circ T_k^{L-1} \circ \cdots \circ T_k^{(1)}(\boldsymbol{z}_{k-1}) + \boldsymbol{z}_{k-1}\right), \quad k = 1, 2, \ldots, K. \tag{3.28}$$

Here L_i is the number of fully connected layer in ith block, $T_k^{(l)}$ is the same nonlinear map defined in (3.26) with $\boldsymbol{W}_l^k \in \mathbb{R}^{M \times N}$, $\boldsymbol{b}_l^k \in \mathbb{R}^M$, $\boldsymbol{V} \in \mathbb{R}^{N \times d}$, $\boldsymbol{\alpha}_i \in \mathbb{R}^{N \times M}$ and $\sigma_i(\cdot)$ is an elementwise activation function. The model parameters are $\boldsymbol{\Theta} = \{\boldsymbol{\gamma}, \boldsymbol{\alpha}_i, \boldsymbol{W}_l^k, \boldsymbol{b}_l^k, \boldsymbol{V}\}$. The original ResNet [29] takes σ_2 as a nonlinear activation function such as ReLU. Later studies indicate that one can also take σ_2 as the identity function [18, 30]. Then at an infinite length, i.e., $L \to \infty$, (3.28) corresponds to the ODE

$$\frac{\mathrm{d}\boldsymbol{z}}{\mathrm{d}t} = f(\boldsymbol{z}), \quad \boldsymbol{z}_0 = \boldsymbol{x}. \tag{3.29}$$

Compared with fully connected DNN models, which may suffer from numerical instabilities in the form of exploding or vanishing gradients [19, 28, 40], very deep ResNet can be constructed to avoid these issues.

Given that the proposed numerical scheme employs neural networks with time-dependent parameters to approximate the solution of a gradient flow, there is no need to employ a deep neural network. In all numerical experiments for L^2-gradient flows, we utilize ResNet with only a few blocks. The detailed settings for each numerical experiment will be described in the next section. We'll compare the numerical performance of different neural network architectures in future work.

3.3.2. Neural network architectures for Lagrangian methods. The proposed Lagrangian method seeks for a neural network to approximate a diffeomorphism from $\mathbb{R}^d$ to $\mathbb{R}^d$. This task involves two main challenges: one is ensuring that the map is a diffeomorphism, and the other is efficiently and robustly computing the deformation tensor F and its determinant $\det \mathsf{F}$. Fortunately, various neural network architectures have been proposed to approximate a transport map. Examples include planar flow [64], auto-regressive flow [39], continuous-time diffusive flow [69], neural spline flow [17], and convex potential flow [32].

One way to construct a neural network $\mathbb{R}^d \to \mathbb{R}^d$ for approximating a diffeomorphism is to use a planar flow. A K-layer planar flow is defined by $T = T_K \circ \cdots T_1 \circ T_0$, where $T_k : \mathbb{R}^d \to \mathbb{R}^d$ is given by

$$\boldsymbol{x}^{k+1} = T_k(\boldsymbol{x}^k) = \boldsymbol{x}^k + \boldsymbol{u}_k h(\boldsymbol{w}_k^{\mathrm{T}} \boldsymbol{x}^k + b_k). \tag{3.30}$$

Here, $\boldsymbol{w}_k, \boldsymbol{u}_k \in \mathbb{R}^d$, $b_k \in \mathbb{R}$, and h is a smooth, elementwise, nonlinear activation function such as tanh. Direct computation shows that

$$J_k = \det(\nabla T_k) = 1 + h'(\boldsymbol{w}_k^{\mathrm{T}} \boldsymbol{x}^k + b_k) \boldsymbol{u}_k^{\mathrm{T}} \boldsymbol{w}_k. \tag{3.31}$$

Clearly, T_k is a diffeomorphism if $h'(\boldsymbol{w}_k^{\mathrm{T}} \boldsymbol{x}^k + b_k) \boldsymbol{u}_k^{\mathrm{T}} \boldsymbol{w}_k < 1 \ \forall \boldsymbol{x}_k$. The determinant of the transport map can be computed as $\det(\nabla T) = J_K J_{K-1} \dots J_0$. and we have $\varphi(T(\boldsymbol{x})) = \frac{\varphi^n(\boldsymbol{x})}{\det(\nabla T)}$.

One limitation of the planar flow is its potential lack of expressive power. Another commonly employed neural network architecture for approximating a map is the convex potential flow [32], which defines a diffeomorphism via the gradient of a strictly convex function that is parameterized by an Input Convex Neural Network (ICNN) [2]. A fully connected, K-layer ICNN can be written as

$$\boldsymbol{z}_{l+1} = \sigma_l(\boldsymbol{W}_l \boldsymbol{z}_l + \boldsymbol{A}_l \mathbf{X} + \boldsymbol{b}_l), \quad l = 0, 1, \dots K-1, \tag{3.32}$$

where $\boldsymbol{z}_0 = 0$, $\boldsymbol{W}_0 = 0$, $\boldsymbol{\Theta} = \{\boldsymbol{W}_l, \boldsymbol{A}_l, \boldsymbol{b}_l\}$ are the parameters to be determined, and σ_l is the nonlinear activation function. As proved in [2], if all entries $\boldsymbol{W}_l$ are nonnegative, and all σ_l are convex and nondecreasing, then $f(\mathbf{X}; \boldsymbol{\Theta})$ is a convex function with respect to $\mathbf{X}$. Hence $\nabla_{\mathbf{X}} f(\mathbf{X}; \boldsymbol{\Theta})$ provides a parametric approximation to a flow map. In the current study, we adopt the concept of the convex potential flow to develop the Lagrangian EVNN method. We'll explore other types of neural network architectures in future work.

Remark 3.5. It is worth mentioning that the gradient of a convex function f only defines a subspace of diffeomorphism, and it is unclear whether the optimal solution of (3.14) belongs to this subspace.

4. Numerical experiments. In this section, we test the proposed EVNN methods for various L^2-gradient flows and generalized diffusions. To evaluate the accuracy of different methods, we define the l^2-error

$$\left(\frac{1}{N} \sum_{i=1}^{N} \left| \varphi_{\mathrm{NN}}(\boldsymbol{x}_i) - \varphi_{\mathrm{ref}}(\boldsymbol{x}_i) \right|^2 \right)^{1/2}$$

and the relative l_2-error

$$\left(\sum_{i=1}^{N} \left| \varphi_{\mathrm{NN}}(\boldsymbol{x}_i) - \varphi_{\mathrm{ref}}(\boldsymbol{x}_i) \right|^2 / \sum_{i=1}^{N} \left| \varphi_{\mathrm{ref}}(\boldsymbol{x}_i) \right|^2 \right)^{1/2}$$

between the neural network solution φ_{NN} and the corresponding reference solution φ_{ref} on a set of test samples $\{\boldsymbol{x}_i\}_{i=1}^N$. The reference solution $\varphi_{\mathrm{ref}}(\boldsymbol{x})$ is either an analytic solution or a numerical solution obtained by a traditional numerical method. In some numerical examples, we also plot pointwise absolute errors $|\varphi_{\mathrm{ref}} - \varphi_{\mathrm{NN}}|$ at the grid points.

4.1. Poisson equations. Although the proposed method is designed for evolutionary equations, it can also be used to compute equilibrium solutions for elliptic problems. In this subsection, we compare the performance of the EVNN method with two classical neural network-based algorithms, PINN and DRM, in the context of solving Poisson equations.

We first consider a two-dimensional (2D) Poisson equation with a Dirichlet boundary condition

$$\begin{cases} -\Delta u(\boldsymbol{x}) = f(\boldsymbol{x}), & \boldsymbol{x} \in \Omega \subset \mathbb{R}^d, \\ u(\boldsymbol{x}) = g(x), & \boldsymbol{x} \in \partial\Omega. \end{cases} \tag{4.1}$$

Since EVNN is developed for evolution equations, we solve the following L^2-gradient flow

$$\frac{\mathrm{d}}{\mathrm{d}t}\left(\int_\Omega \frac{1}{2}|\nabla u|^2 - f(\boldsymbol{x})u(\boldsymbol{x})\mathrm{d}\boldsymbol{x} + \lambda\int_{\partial\Omega}|u(\boldsymbol{x}) - g(\boldsymbol{x})|^2\mathrm{d}S\right) = -\int_\Omega |u_t|^2\mathrm{d}\boldsymbol{x} \tag{4.2}$$

to get a solution of the Poisson equation (4.1). Here, $\lambda\int_{\partial\Omega}|u(\boldsymbol{x}) - g(\boldsymbol{x})|^2\mathrm{d}S$ is the surface energy that enforces the Dirichlet boundary condition. The corresponding gradient flow equation is given by

$$u_t = \Delta u(x) + f(\boldsymbol{x}), \tag{4.3}$$

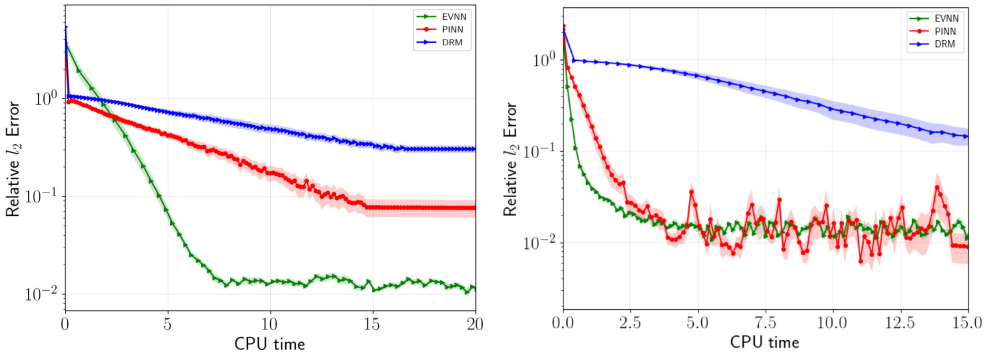
subject to a Robin boundary condition $u(\boldsymbol{x}) - g(\boldsymbol{x}) = \frac{1}{2\lambda}\frac{\partial u}{\partial \boldsymbol{n}}$, where $\boldsymbol{n}$ is the outer normal of $\partial\Omega$. One can recover the original Dirichlet boundary condition by letting $\lambda \to \infty$. Such a penalty approach is also used in PINN and DRM, and we take $\lambda = 500$ in all numerical experiments below.

We consider the following two cases:

- Case 1: $\Omega = (0,\pi)\times(-\pi/2,\pi/2)$, $f(\boldsymbol{x}) = 2\sin x\cos y$ and $g(\boldsymbol{x}) = 0$. The exact solution is $u(\boldsymbol{x}) = \sin x\cos y$.
- Case 2: $\Omega = \{(x,y) \mid |\boldsymbol{x}| \le 1\}$, $f(\boldsymbol{x}) = \frac{\pi^2}{4}\sin\left(\frac{\pi}{2}(1-|\boldsymbol{x}|)\right) + \frac{\pi}{2|\boldsymbol{x}|}\cos\left(\frac{\pi}{2}(1-|\boldsymbol{x}|)\right)$ and $g(\boldsymbol{x}) = 0$. The exact solution is $u(\boldsymbol{x}) = \sin(\frac{\pi}{2}(1-|\boldsymbol{x}|))$.

We employ a 1-block ResNet with 20 hidden nodes and one hidden layer for all cases. The total number of parameters is 501. We apply Xavier Initialization [26] to initialize the weights of neural networks in all cases. To evaluate the integration in all methods, we generate 2500 samples in Ω using a Latin hypercube sampling (LHS) and 50 samples on each boundary of $\partial\Omega$ using a one-dimensional (1D) uniform sampling for case 1. For case 2, we generate 2500 samples in $(-1,1)^2$ using LHS, but only use the samples satisfies $x^2 + y^2 < 1$ as training samples. Additionally, we generate 200 training samples $(\cos\theta_i, \sin\theta_i)_{i=1}^{200}$ on the boundary, with $\{\theta_i\}_{i=1}^{200}$ being generated by a uniform distribution on $(0, 2\pi)$. For PINN and DRM, we use Adam to minimize the loss function and use a different set of samples at each iteration. For EVNN, we use a different set of samples at each time step and employ an L-BFGS to solve the optimization problem (3.10).

Due to randomness arising from the stochastic initialization of the neural network and the sampling process, we repeated each method 10 times and plotted the mean and standard error of the relative l_2-errors. Figure 4.1(a) shows the relative l_2-errors



(a) $u(x,y) = \sin(x)\cos(y)$ **(b)** $u(\boldsymbol{x}) = \sin(\frac{\pi}{2}(1-|\boldsymbol{x}|))$

FIG. 4.1. *Relative l^2 error with respect to CPU time for different methods for 2D Poisson equations.*

of different methods with respect to the CPU time for both cases. The training was executed on a MacBook Pro equipped with an M2 chip. In case 1, the relative l_2 error of each method is evaluated on a test set comprising a uniform grid of 101×101 points within the domain $(0,1)^2$. For case 2, the test set is defined as the collection of points $\{(x_i, y_j) | x_i^2 + y_j^2 < 1\}$, where (x_i, y_j) forms a uniform 201×201 grid on $(-1,1)^2$. The CPU time is computed by averaging the time of 10 trials at nth iteration. The numerical results clearly show that the proposed method is more efficient than both PINN and DRM. It significantly enhances the efficiency and test accuracy of DRM through the introduction of time relaxation. While PINN may achieve better test accuracy in case 2, it exhibits a larger standard error in both cases.

Next, we consider a high-dimensional Poisson equation

$$-\Delta u = f(\boldsymbol{x}), \quad \boldsymbol{x} \in \Omega = (-1,1)^d, \tag{4.4}$$

with a homogeneous Neumann boundary condition $\frac{\partial u}{\partial n}|_{\partial\Omega} = 0$. We take $f(\boldsymbol{x}) = \pi^2 \sum_{k=1}^d \cos(\pi x_k)$. Similar numerical examples are tested in [21, 53]. The exact solution to this high-dimensional Poisson equation is $u(\boldsymbol{x}) = \sum_{k=1}^d \cos(\pi x_k)$. Following [53], we solve an L^2-gradient flow associated with the free energy

$$\mathcal{F}[u] = \int_{(-1,1)^d} \frac{1}{2}|\nabla u|^2 - fu \, \mathrm{d}\boldsymbol{x} + \lambda \left(\int_{(-1,1)^d} u \, \mathrm{d}\boldsymbol{x} \right)^2, \tag{4.5}$$

where the last term enforces $\int_{[-1,1]^d} u \, \mathrm{d}\boldsymbol{x} = 0$. We take $\lambda = 0.5$ as in [53] in all numerical tests.

Since it is difficult to get an accurate estimation of high-dimensional integration, we choose Adam to minimize (3.10) at each time step and use a different set of samples at each Adam step. This approach allows us to explore the high-dimensional space with a low computational cost. The samples are drawn using LHS. Table 4.1 shows the final relative l^2-error with different neural network settings in different spatial dimensions. The relative l^2-error is evaluated on a test set that comprises 40000 samples generated by LHS. The model is trained with 200 outer iterations ($\tau = 0.01$) and at most 200 iterations for inner optimization (3.10). An early stop criterion was applied if the l_2-norm of the model parameters between two consecutive epochs is less than 10^{-6}. The first column of Table 4.1 specifies the numerical setting as follows: the

TABLE 4.1
Relative l^2-error of high-dimensional Poisson equation with different settings in different dimensions. The first column shows the number of residual blocks, the number of nodes in each fully connected layer, and the number of samples.

Setting	$d=4$	$d=8$	$d=16$	$d=32$
(2, 10, 1000)	0.024	0.046	0.623	0.867
(3, 60, 1000)	0.042	0.048	0.077	0.117
(3, 60, 10000)	0.018	0.022	0.036	0.077

entry $(2, 10)$ means a ResNet with two residual blocks, each with two fully connected layers having 10 nodes is used, and 1000 represents the number of samples drawn in each epoch. As we can see, the proposed method can achieve comparable results with results reported in similar examples in previous work by DRM [21, 53]. It can be observed that increasing the width of the neural network improves test accuracy in high dimensions, as it enhances the network's expressive power. Furthermore, increasing the number of training samples also significantly improves test accuracy in high dimensions. We'll explore the effects of τ, the number of samples, and neural network architecture for the high-dimensional Poisson equation in future work.

4.2. L^2-gradient flow. In this subsection, we apply the EVNN scheme to two L^2-gradient flows to demonstrate its energy stability and numerical accuracy.

4.2.1. Heat equation. We first consider an initial-boundary value problem of a heat equation:

$$\begin{cases} u_t = \Delta u(\mathbf{x}), \quad \mathbf{x} \in \Omega = (0,2)^2, \quad t \in (0,T], \\ u(\boldsymbol{x},t) = 0, \quad \boldsymbol{x} \in \partial\Omega, \quad t \in (0,T], \\ u(\boldsymbol{x},0) = \sin\left(\frac{\pi}{2}x_1\right)\sin\left(\frac{\pi}{2}x_2\right), \quad \boldsymbol{x} \in \Omega = (0,2)^2, \end{cases} \tag{4.6}$$

which can be interpreted as an L^2-gradient flow satisfying the energy-dissipation law

$$\frac{\mathrm{d}}{\mathrm{d}t}\left(\int_\Omega \frac{1}{2}|\nabla u|^2 \,\mathrm{d}\boldsymbol{x} + \lambda \int_{\partial\Omega} |u|^2 \mathrm{d}S\right) = -\int |u_t|^2 \,\mathrm{d}\boldsymbol{x}. \tag{4.7}$$

Again, a surface energy term is added to enforce the Dirichlet boundary condition.

In the numerical simulation, we take $\tau = 0.01$ and use a 1-block ResNet with 20 nodes in each layer. The nonlinear activation function is chosen to be tanh. The total number of parameters is 501. To achieve energy stability, we fix the training samples to eliminate the numerical fluctuation arising from estimating the integration in (3.10). The set of training samples comprises a 301×301 uniform grid on $(0,2)^2$, and an additional 1000 uniformly spaced grid points on each edge of the boundary. To test the accuracy of the solution over time, we compare the EVNN solution with a numerical solution obtained by a finite difference method (FDM). We apply the standard central finite difference and an implicit Euler method to obtain the FDM solution. The numerical scheme can be reformulated as an optimization problem:

$$\phi_h^{n+1} = \arg\min_{u_h \in \mathcal{C}} \frac{1}{2\tau}\|u_h - u_h^n\|_h^2 + \mathcal{F}_h(u_h), \quad \mathcal{F}_h(u_h) = \frac{1}{2}\langle \nabla_h u_h, \nabla_h u_h\rangle_*. \tag{4.8}$$

Here, $\mathcal{C} = \{u_{i,j} \mid 0 < i < I, \quad 0 < j < J\}$ is the set of grid functions defined on a uniform mesh, ∇_h is the discrete gradient, $\mathcal{F}_h$ is the discrete free energy, $\|\cdot\|_h$ is the discrete L_2-norm induced by the discrete L_2-inner product $\langle f, g\rangle = h^2 \sum_{i=1}^{N-1} f_{i,j} g_{i,j}$, and $\langle\cdot,\cdot\rangle_*$

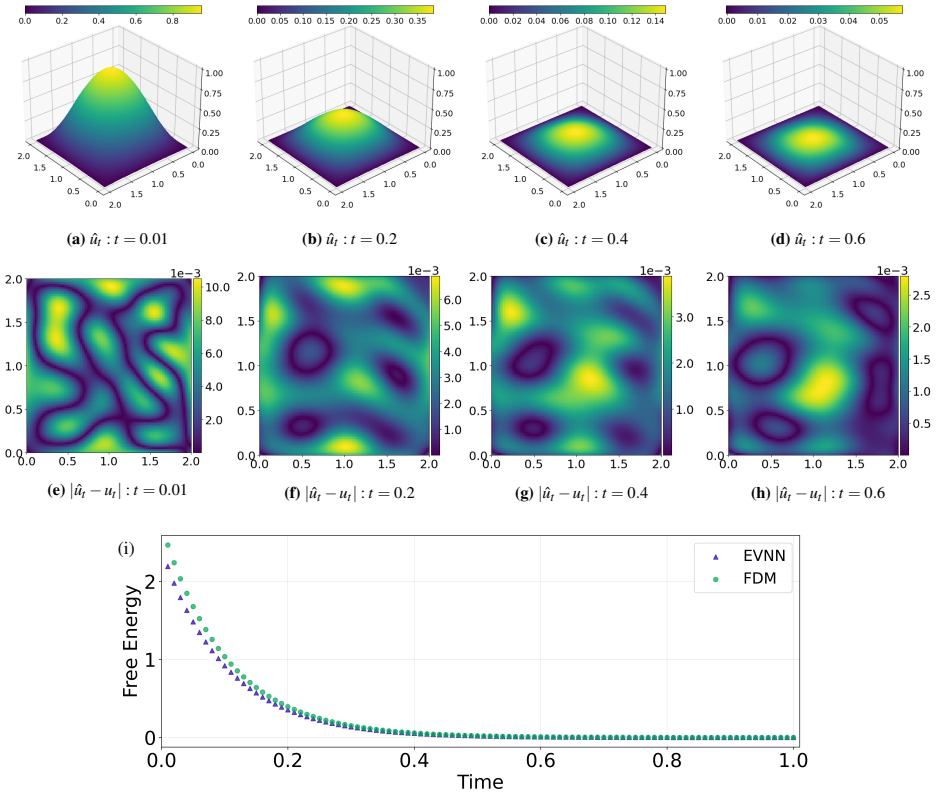


FIG. 4.2. *Numerical results for the heat equation.* (a)–(d) *are the EVNN solutions at* $t = 0.01, 0.2, 0.4,$ *and* 0.6. (e)–(h) *are the absolute differences between the EVNN solutions and the FDM solutions.* (i) *The evolution of discrete free energy with respect to time for both the EVNN and FDM solutions.*

is the discrete inner product defined on the staggered mesh points [49, 66]. We use a 101×101 uniform grid and take $\tau = 0.01$ in the FDM simulation. Figure 4.2(a)–(d) shows the EVNN solution at $t = 0.01, 0.2, 0.4,$ and 0.6. The corresponding pointwise absolute errors are shown in Figure 4.2(e)–(h). The evolution of discrete free energy (evaluated on 101×101 uniform grid) for both methods is shown in Figure 4.2(i). Clearly, the neural-network-based algorithm achieves energy stability and the result is consistent with the FDM. It is worth mentioning that the size of the optimization problem in the neural network-based algorithm is much smaller than that in the FDM, although (4.8) is a quadratic optimization problem and can be solved directly by solving a linear equation of $u_{i,j}$.

4.2.2. Allen–Cahn equation. Next, we consider an Allen–Cahn type equation, which is extensively utilized in phase-field modeling, and becomes a versatile technique to solve interface problems arising from different disciplines [15]. In particular, we focus on the following Allen–Cahn equation on $\Omega = (-1, 1)^2$:

$$
\begin{cases}
\dfrac{\partial \varphi}{\partial t}(\boldsymbol{x}, t) = -\left(\dfrac{F'(\varphi)}{\varepsilon^2} - \triangle \varphi(\boldsymbol{x}, t) + 2W \left(\displaystyle\int \varphi \mathrm{d}\boldsymbol{x} - A \right) \right), & \boldsymbol{x} \in \Omega, \quad t > 0, \\
\varphi(\boldsymbol{x}, t) = -1, & \boldsymbol{x} \in \partial\Omega, \quad t > 0, \\
\varphi(\boldsymbol{x}, 0) = -\tanh(10(\sqrt{4x_1^2 + x_2^2} - 0.5)), & \boldsymbol{x} \in \Omega.
\end{cases} \tag{4.9}
$$

where $\varphi(\boldsymbol{x},t)$ is the phase-field variable, constants ϵ, W, and A are model parameters. The Allen–-Cahn equation (4.9) can be regarded as an L^2-gradient flow, derived from an energy-dissipation law

(4.10)
$$\frac{\mathrm{d}}{\mathrm{d}t}\mathcal{F}[\varphi]=-\int_\Omega |\varphi_t|^2\mathrm{d}\boldsymbol{x},\quad \mathcal{F}[\varphi]=\int_\Omega \frac{1}{2}\left|\nabla\varphi\right|^2+\frac{1}{4\varepsilon^2}\left(\varphi^2-1\right)^2\mathrm{d}\boldsymbol{x}+W\left(\int_\Omega \varphi\mathrm{d}\boldsymbol{x}-A\right)^2.$$

Here, the last term in the energy is a penalty term for the volume constraint $\int\varphi\mathrm{d}\boldsymbol{x}=A$, as the standard Allen–Cahn equation does not preserve the volume fraction $\int\varphi\mathrm{d}\boldsymbol{x}$ over time. In the numerical simulation, we take $A=-(4-\pi r^2)+\pi r^2, r=0.5, \frac{1}{\epsilon^2}=100$, and $W=1000$.

Due to the complexity of the initial condition, we utilized a larger neural netwok compared to the one employed in the previous subsection. Our choice was a 1-block ResNet, which has two fully connected layers, each containing 20 nodes. The total number of parameters is 921. The nonlinear activation function is chosen to be tanh. The set of training samples comprises a 301×301 uniform grid on $(-1,1)^2$, and an additional 1000 uniformly spaced grid points on each edge of the boundary. To test the numerical accuracy of the EVNN scheme for this problem, we also solve (4.9) by a finite element method, which approximates the phase-field variable φ by a piecewise linear function $\varphi_h(\boldsymbol{x},t)=\sum_{i=1}^N\gamma_i(t)\psi_i(\boldsymbol{x})$, where $\psi_i(X)$ are hat functions supported on the computational mesh. Inserting φ_h into the continuous energy–dissipation law (4.10), we get a discrete energy–dissipation law with the discrete energy and dissipation given by

$$\mathcal{F}_h^{\mathrm{FEM}}(\boldsymbol{\gamma})=\sum_{e=1}^{N_e}\int_{\tau_e}\frac{1}{2}\left|\sum_{i=1}^N\gamma_i\nabla\psi_i(\boldsymbol{x})\right|^2+\frac{1}{\epsilon^2}\sum_{i=1}^N(\gamma_i^2-1)^2\psi_i(\boldsymbol{x})\mathrm{d}\boldsymbol{x},$$

and $\mathcal{D}_h^{\mathrm{FEM}}(\boldsymbol{\gamma},\boldsymbol{\gamma}')=\sum_{e=1}^{N_e}\int_{\tau_e}|\sum_{i=1}^N\gamma_i'(t)\psi_i(\boldsymbol{x})|^2\mathrm{d}\boldsymbol{x}$, respectively. Here τ_e is used to denote a finite element cell, and N_e is the number of cells. This form of discretization was used in [79], which constructs a piecewise linear approximation to the nonlinear term in the discrete energy. We can update γ_i at each time step by solving the optimization problem (3.7), i.e.,

$$\boldsymbol{\gamma}^{n+1}=\arg\min_{\boldsymbol{\gamma}\in\mathbb{R}^N}\frac{1}{2\tau}\mathsf{D}(\boldsymbol{\gamma}-\boldsymbol{\gamma}^n)\cdot(\boldsymbol{\gamma}-\boldsymbol{\gamma}^n)+\mathcal{F}_h(\boldsymbol{\gamma}),\tag{4.11}$$

where $\mathsf{D}_{ij}=\int_\Omega\psi_i\psi_j\mathrm{d}\boldsymbol{x}$ is the mass matrix. The optimization problem (4.11) is solved by L-BFGS in our numerical implementation.

The simulation results are summarized in Figure 4.3. It clearly shows that our method can achieve comparable results with the FEM. Numerical simulation of phase-field type models is often challenging. To capture the dynamics of the evolution of the thin diffuse interface, the mesh size should be much smaller than ϵ, the width of the diffuse interface. Traditional numerical methods often use an adaptive or moving mesh approach to overcome this difficulty. In contrast, a neural network-based numerical scheme has a mesh-free feature. The number of parameters of the neural network can be much smaller than the number of samples needed to resolve the diffuse interface. Consequently, the dimension of the optimization problem in the neural network-based scheme is much smaller than that in the FEM scheme without using adaptive or moving meshes.

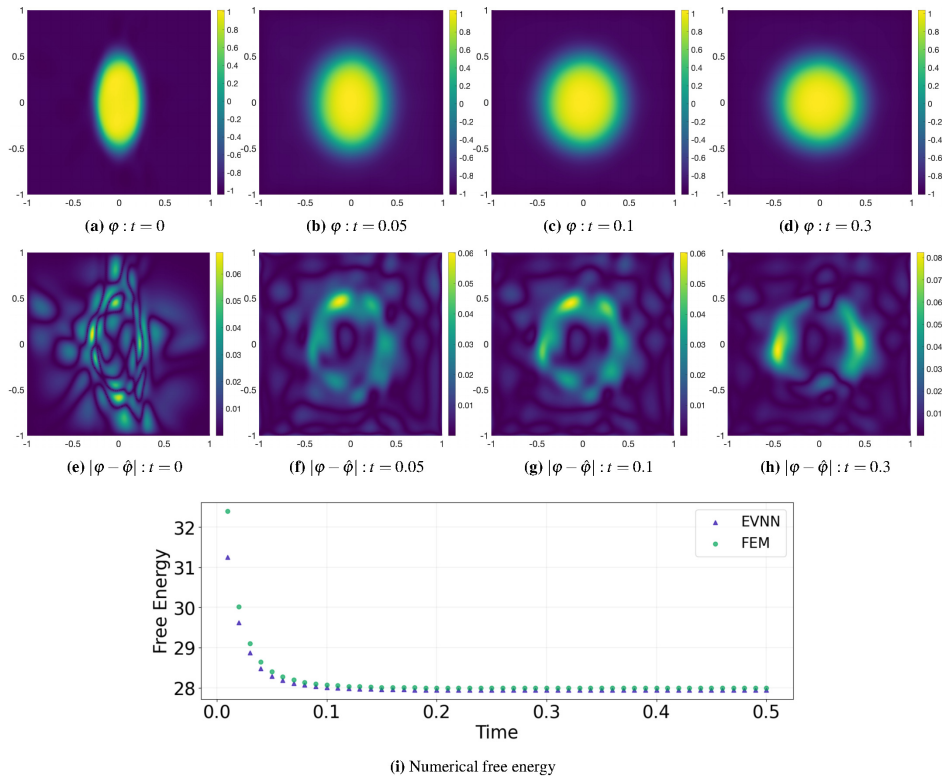


FIG. 4.3. *Numerical results for the Allen–Cahn equation with a volume constraint.* (a)–(d) *EVNN solutions at* $t = 0, 0.05, 0.1,$ *and* 0.3*, respectively.* (e)–(h) *Absolute differences between the EVNN solutions and the FEM solutions at* $t = 0, 0.05, 0.1,$ *and* 0.3*, respectively.* (i) *Evolution of numerical free energies with respect to time for both the FEM and EVNN solutions.*

4.3. Generalized diffusions. In this subsection, we apply the proposed Lagrangian EVNN scheme to solving a Fokker–Planck equation and a porous medium equation.

4.3.1. Fokker–Planck equation. We first consider a Fokker–Planck equation

$$\begin{cases} \rho_t = \nabla \cdot (\nabla \rho + \rho \nabla V), & \boldsymbol{x} \in \mathbb{R}^d, \quad t \in (0, T], \\ \rho(\boldsymbol{x}, 0) = \rho_0(\boldsymbol{x}), & \boldsymbol{x} \in \mathbb{R}^d, \end{cases} \tag{4.12}$$

where $V(\boldsymbol{x})$ is a prescribed potential energy. The Fokker–Planck equation can be viewed as a generalized diffusion satisfying the energy-dissipation law

$$\frac{\mathrm{d}}{\mathrm{d}t} \int_{\mathbb{R}^d} \rho \ln \rho + \rho V(\boldsymbol{x}) \mathrm{d}\boldsymbol{x} = - \int_{\mathbb{R}^d} \rho |\mathbf{u}|^2 \mathrm{d}\boldsymbol{x}, \tag{4.13}$$

where the probability density ρ satisfies the kinematics $\rho_t + \nabla \cdot (\rho \mathbf{u}) = 0$.

We test our numerical scheme for solving the Fokker–Planck equation in two dimensions and four dimensions, respectively. For both numerical experiments, we adopt the augmented-ICNN proposed in [32]. We use a 1 ICNN block of six fully connected layers with 32 hidden nodes in each layer. The activation function is chosen as the Gaussian-Softplus function $\sigma(x) = \sqrt{\frac{2}{\pi}}(\sqrt{2}x \int_0^{x/\sqrt{2}} e^{-t^2} dt + e^{-\frac{x^2}{2}} + \sqrt{\frac{\pi}{2}}x)$. In addition, we use L-BFGS to solve the optimization problem at each time step.

In the 2D case, we consider $V(\boldsymbol{x})$ as follows:

$$V = \frac{1}{2}(\boldsymbol{x} - \mu_{target})^T \Sigma_{target}^{-1}(\boldsymbol{x} - \mu_{target}) \quad \text{with} \quad \mu_{target} = \left(\frac{1}{3}, \frac{1}{3}\right),$$

and $\Sigma_{target} = \begin{bmatrix} \frac{5}{8} & -\frac{3}{8} \\ -\frac{3}{8} & \frac{5}{8} \end{bmatrix}$. The initial condition $\rho_0(\boldsymbol{x})$ is set to be the 2D standard Gaussian $\mathcal{N}(\mathbf{0}, \mathbf{I})$. The exact solution of (4.12) takes the following analytical form [33]:

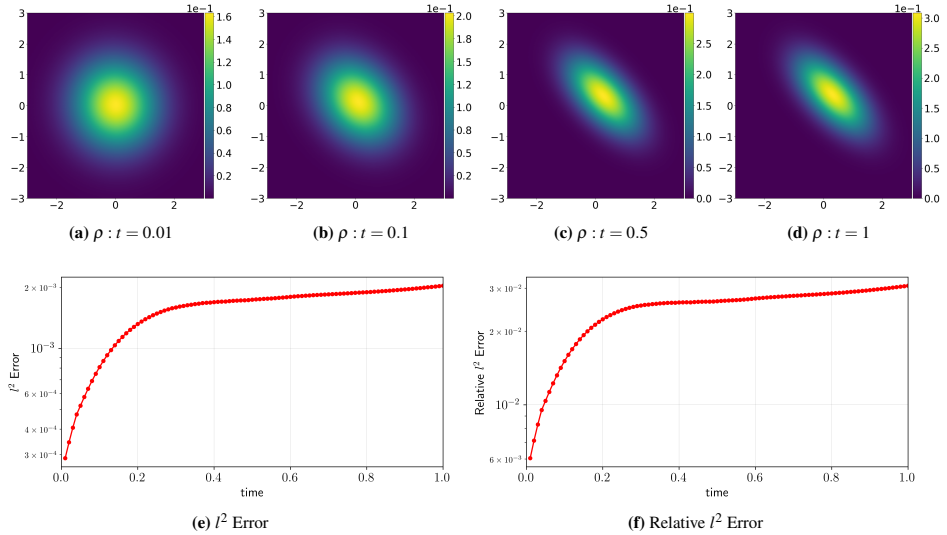
$$\rho(\boldsymbol{x}, t) \sim \mathcal{N}(\mu(t), \Sigma(t)), \tag{4.14}$$

where $\mu(t) = (1 - \mathbf{e}^{-4t})\mu_{target}$ and $\Sigma(t) = \begin{bmatrix} \frac{5}{8} + \frac{3}{8} \times e^{-8t} & -\frac{3}{8} + \frac{3}{8} \times e^{-8t} \\ -\frac{3}{8} + \frac{3}{8} \times e^{-8t} & \frac{5}{8} + \frac{3}{8} \times e^{-8t} \end{bmatrix}$. We draw 10000 samples from $\rho_0(\boldsymbol{x})$ as the training set. As an advantage of the neural network-based algorithm, we can compute $\rho^n(\boldsymbol{x})$ point-wisely through a function composition $\rho^n(\boldsymbol{x}) = \rho_0 \tilde{\circ} (\Phi^n)^{-1}(\boldsymbol{x})$, where $(\Phi^n)^{-1}(\boldsymbol{x})$ can be computed by solving a convex optimization problem, i.e., $\Phi^{-1}(\boldsymbol{x}) = \arg\min_{\boldsymbol{y}} \Phi(\boldsymbol{y}) - \boldsymbol{x}^{\mathrm{T}}\boldsymbol{y}$. Figure 4.4(a) - (d) shows the predicted density on a 301×301 uniform grid on $(-3, 3)^2$. The absolute and relative l_2-errors on the grid are shown in Figures 4.4(e)-(f). It can be noticed that the relative l^2-error for the predicted solution is around 10^{-2}, which is quite accurate given the limited number of samples used.

In the 4D case, we take $V(\boldsymbol{x})$ to be

$$V = \frac{1}{2}(\boldsymbol{x} - \mu_{target})^T \Sigma_{target}^{-1}(\boldsymbol{x} - \mu_{target}) \quad \text{with} \quad \mu_{target} = \left(\frac{1}{3}, \frac{1}{3}, 0, 0\right),$$

and $\Sigma_{target} = \begin{bmatrix} \frac{5}{8} & -\frac{3}{8} \\ -\frac{3}{8} & \frac{5}{8} \end{bmatrix} \bigoplus \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$. The initial condition $\rho_0(\boldsymbol{x})$ is set to be a 4D standard normal distribution $\mathcal{N}(\mathbf{0}, \mathbf{I})$. The exact solution of (4.12) follows a normal distribution with $\mu(t) = ((1 - \mathbf{e}^{-4t})\frac{1}{3}, (1 - \mathbf{e}^{-4t})\frac{1}{3}, 0, 0)$, and $\Sigma(t) = \begin{bmatrix} \frac{5}{8} + \frac{3}{8} \times e^{-8t} & -\frac{3}{8} + \frac{3}{8} \times e^{-8t} \\ -\frac{3}{8} + \frac{3}{8} \times e^{-8t} & \frac{5}{8} + \frac{3}{8} \times e^{-8t} \end{bmatrix} \bigoplus$



(a) $\rho : t = 0.01$ (b) $\rho : t = 0.1$ (c) $\rho : t = 0.5$ (d) $\rho : t = 1$

(e) l^2 Error (f) Relative l^2 Error

FIG. 4.4. *Numerical results for the 2D Fokker–Planck equation.* (a)–(d) *Predicted solution on a* 301×301 *uniform mesh on* $(-3, 3)^2$ *at* $t = 0, 0.1, 0.5,$ *and* 1*, respectively.* (e)–(f) *The absolute and relative* l_2*-errors of the solution over time, evaluated on the uniform mesh.*

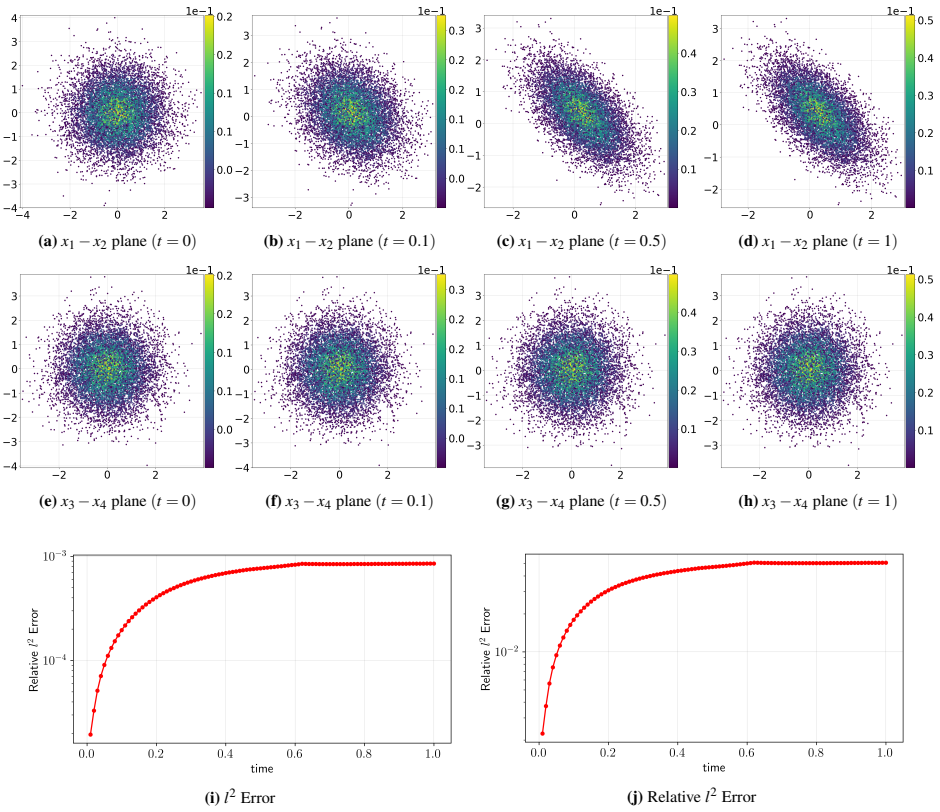


FIG. 4.5. *Numerical results for the four-dimensional (4D) Fokker–Planck equation.* (a)–(d) *Distribution of samples, projected in* x_1-x_2 *plane, as well as its weight at* $t = 0, 0.1, 0.5,$ *and* 1, *respectively.* (e)–(h) *Distribution of samples, projected in* x_3-x_4 *plane, as well as its weight at* $t = 0, 0.1, 0.5,$ *and* 1, *respectively.* (i)–(j) *The* l_2*-errors and the relative* l^2*-errors of the solution over time, evaluated in the training set.*

$\left[\begin{smallmatrix} 1 & 0 \\ 0 & 1 \end{smallmatrix}\right]$. Here $\bigoplus$ is the direct sum. Same as the 2D case, we draw 10000 initial samples from $\rho_0(\boldsymbol{x})$. Figure 4.5 shows the evolution of samples as well the values of the numerical solution on individual samples over space and time. Interestingly, the relative l^2-error in 4D case is similar to the 2D case. The dimension-independent error bound suggests the potential of the current method for handling higher-dimensional problems.

Fokker–Planck type equations have a wide range of application in machine learning. One of the fundamental tasks in modern statistics and machine learning is to estimate or generate samples from a target distribution $\rho^*(\boldsymbol{x})$, which might be completely known, partially known up to a normalizing constant, or empirically given by samples. Examples include Bayesian inference [7], numerical integration [54], space-filling design [61], density estimation [69], and generative learning [59]. These problems can be transformed as an optimization problem, which is to seek for a $\rho^{\text{opt}} \in \mathcal{Q}$ by solving an optimization problem

$$\rho^{\text{opt}} = \arg\min_{\rho \in \mathcal{Q}} D(\rho || \rho^*), \tag{4.15}$$

where $\mathcal{Q}$ is the admissible set, $D(p||q)$ is a dissimilarity function that assesses the differences between two probability measures p and q. The classical dissimilarities

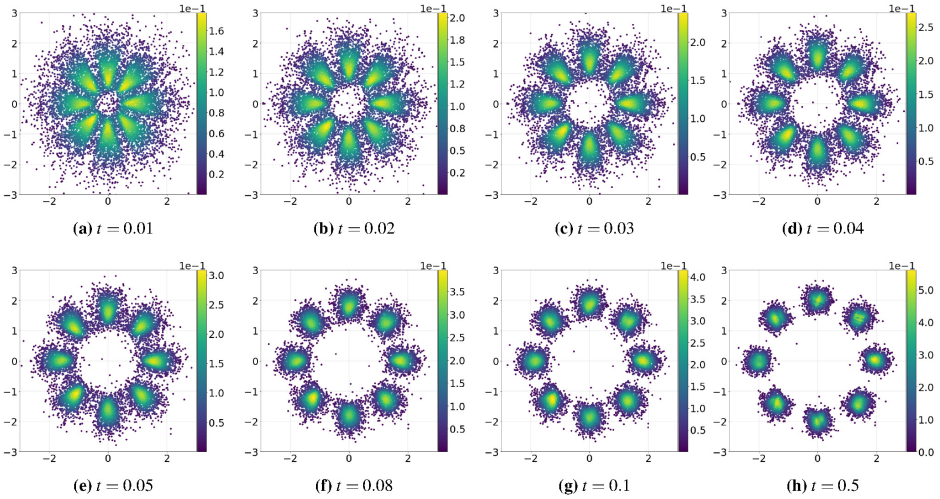


FIG. 4.6. *The distribution of samples and their weight at different times for the Fokker–Planck equation with a complicated* $V(\boldsymbol{x})$*, which corresponds to an eight-component mixture Gaussian distribution. The initial* $10,000$ *samples are drawn from a standard normal distribution* $\mathcal{N}(\mathbf{0}, \mathbf{I})$.

include the Kullback–Leibler (KL) divergence and the Maximum Mean Discrepancy (MMD). The optimal solution of the optimization problem can be obtained by solving a Fokker–Planck type equation, given by

$$\rho_t = \nabla \cdot (\rho \nabla \mu), \quad \mu = \frac{\delta D(\rho||\rho^*)}{\delta \rho}. \tag{4.16}$$

The developed numerical approach has potential applications in these machine learning problems.

To illustrate this point, we consider a toy problem that is widely used in the machine learning literature [32]. The goal is to sample from a target distribution ρ^*, which is known up to a normalizing constant. We take the dissimilarity function as the KL divergence $\mathrm{KL}(\rho||\rho^*) = \int_\Omega \rho(\boldsymbol{x}) \ln(\frac{\rho}{\rho^*}) \mathrm{d}\boldsymbol{x}$, then (4.16) is reduced to the Fokker–Planck equation (4.12) with $V(\boldsymbol{x}) = -\ln \rho^*$. In the numerical experiment, we take the target distribution $\rho^*(\boldsymbol{x}) = \frac{1}{8}\sum_{i=1}^{8} N(\boldsymbol{x}|\boldsymbol{\mu}_i, \boldsymbol{\Sigma})$, an eight component mixture Gaussian distribution, where $\boldsymbol{\mu}_1 = (0, 4)$, $\boldsymbol{\mu}_2 = (2.8, 2.8)$, $\boldsymbol{\mu}_3 = (4, 0)$, $\boldsymbol{\mu}_4 = (-2.8, 2.8)$, $\boldsymbol{\mu}_5 = (-4, 0)$, $\boldsymbol{\mu}_6 = (-2.8, -2.8)$, $\boldsymbol{\mu}_7 = (0, -4)$, $\boldsymbol{\mu}_8 = (2.8, -2.8)$, and $\boldsymbol{\Sigma} = \mathrm{diag}(0.2, 0.2)$. The simulation results are summarized in Figure 4.6, in which we first draw 10000 samples from a standard normal distribution $\mathcal{N}(\mathbf{0}, \mathbf{I})$ and show the distribution of samples as well as their weights at different time. It can be noticed that the proposed neural network-based algorithm can generate a weighted sample from a complicated target distribution.

4.3.2. Porous medium equation. Next, we consider a porous medium equation (PME), $\rho_t = \Delta \rho^\alpha$,where $\alpha > 1$ is a constant. The PME is a typical example of nonlinear diffusion equations. One important feature of the PME is that the solution of the PME has a compact support at any time $t > 0$ if the initial data has a compact support. The free boundary of the compact support moves outward with a finite speed, known as the property of finite speed propagation [71]. As a consequence, numerical simulations of the PME are often difficult to conduct using Eulerian methods, which may fail to capture the movement of the free boundary and suffer from

numerical oscillations [50]. In a recent work [50], the authors developed a variational Lagrangian scheme using a finite element method. Here we show the ability of the Lagrangian EVNN scheme to solve the PME with a free boundary. Following [50], we employ the energy-dissipation law

$$\frac{\mathrm{d}}{\mathrm{d}t}\int_{\mathbb{R}^d}\frac{\alpha}{(\alpha-1)(\alpha-2)}\rho^{\alpha-1}\mathrm{d}\boldsymbol{x}=-\int_{\mathbb{R}^d}|\mathbf{u}|^2\mathrm{d}\boldsymbol{x} \tag{4.17}$$

to develop the EVNN scheme.

We take $\alpha=4$ in the simulation. To test the numerical accuracy of the EVNN scheme for solving the PME, we consider a 2D Barenblatt–Pattle solution of the PME. The Barenblatt–Pattle solution in d-dimensional space is given by

$$B_\alpha(\boldsymbol{x},t)=t^{-k}\left[\left(C_0-\frac{k(\alpha-1)}{2d\alpha}\frac{|\boldsymbol{x}|^2}{t^{2k/d}}\right)_+\right]^{1/(\alpha-1)},\quad \boldsymbol{x}\in\mathbb{R}^d, \tag{4.18}$$

where $k=(\alpha-1+2/d)^{-1}$, $u_+:=\max(u,0)$, and C_0 is a constant that is related to the initial mass. This solution is radially symmetric, self-similar, and has compact support $|\boldsymbol{x}|\le\xi_\alpha(t)$ for any finite time, where $\xi_\alpha(t)=\sqrt{\frac{2d\alpha C_0}{k(\alpha-1)}}t^{k/d}$. We take the Barenblatt solution (4.18) with $C_0=0.1$ at $t=0.1$ as the initial condition, and compare the numerical solution at T with the exact solution $B_\alpha(\boldsymbol{x},T+0.1)$. We choose $\{x_i^0\}_{i=1}^N$ as a set of uniform grid points inside a disk $\{(x,y)\mid\sqrt{x^2+y^2}\le\xi_\alpha(0.1)\}$. To visualize the numerical results, we also draw 500 points distributed with equal arc length on the initial free boundary and evolve them using the trained ICNN. The numerical results with $\tau=0.005$ are shown in Figure 4.7. It can be noticed that the proposed neural network-based algorithm can well approximate the Barenblatt–Pattle solution and capture the movement of the free boundary.

We note that numerical methods to solve generalized diffusions (1.4) can also be formulated in the Eulerian frame of reference based on the notion of Wasserstein gradient flow [35]. However, it is often challenging to compute the Wasserstein distance

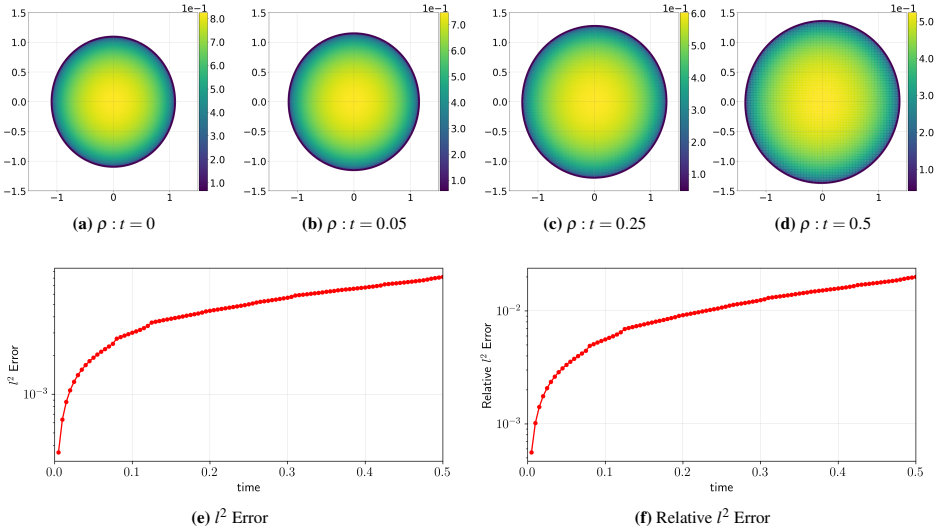


FIG. 4.7. *Numerical results for the porous medium equation.* (a)–(d) *The numerical solution at* $t=0,0.05,0.25,$ *and* 0.5*, respectively.* (e)–(f) *The* l_2*-errors of the solution.*

between two probability densities efficiently with high accuracy, which often requires solving an additional min-max problem or minimization problem [5, 6, 9, 33]. In a recent study [33], the authors initially employ a fully connected neural network to approximate ρ and subsequently utilize an additional ICNN to calculate the Wasserstein distance between two probability densities. Compared with their approach, our method is indeed more efficient and accurate.

5. Conclusion. In this paper, we develop structure-preserving EVNN schemes for simulating L^2-gradient flows and generalized diffusions by utilizing neural networks as a tool for spatial discretization, within the framework of the discrete energetic variational approach. These numerical schemes are directly constructed based on a prescribed continuous energy-dissipation law for the system, without the need for the underlying PDE. The incorporation of mesh-free neural network discretization opens up exciting possibilities for tackling high-dimensional gradient flows arising in different applications in the future.

Various numerical experiments are presented to demonstrate the accuracy and energy stability of the proposed numerical schemes. In our future work, we will explore the effects of different neural network architectures, sampling strategies, and optimization methods, followed by a detailed numerical analysis. Additionally, we intend to employ EVNN schemes to investigate other complex fluid models, including the Cahn–Hilliard equation and Cahn–Hilliard–Navier–Stokes equations, as well as solve machine learning problems such as generative modeling and density estimation.

Appendix A. In this appendix, we provide the detailed computation for the EnVarA derivation of a generalized diffusion. Due to the kinematics relation (2.4), the action functional $\mathcal{A}[\boldsymbol{x}]$ can be written as

$$\begin{aligned}\mathcal{A}[\boldsymbol{x}] = &-\int_0^T \int_{\Omega_0} \omega\left(\frac{\rho_0(\mathbf{X})}{\det \mathsf{F}}\right) \det \mathsf{F} + V(\boldsymbol{x}(\mathbf{X},t))\rho_0(\mathbf{X}) \\ &+ \frac{1}{2}\rho_0(\mathbf{X})\left(\int_{\Omega_0} K(\boldsymbol{x}(\mathbf{X},t),\boldsymbol{y}(\mathbf{X},t))\rho_0(\mathbf{X})\mathrm{d}\mathbf{X}\right)\mathrm{d}\mathbf{X}\mathrm{d}t\end{aligned}$$

in Lagrangian coordinates. A direct computation leads to (see [24] for more detailed computations)

$$\begin{aligned}\delta\mathcal{A} = &-\int_0^T \int_{\Omega_0} \left[-\omega'\left(\frac{\rho_0(\mathbf{X})}{\det \mathsf{F}}\right)\cdot\frac{\rho_0(\mathbf{X})}{\det \mathsf{F}} + \omega\left(\frac{\rho_0(\mathbf{X})}{\det \mathsf{F}}\right)\right] \times \left(\mathsf{F}^{-\mathrm{T}} : \nabla_{\mathbf{X}}\delta\boldsymbol{x}\right)\det \mathsf{F} \\ &+ \rho_0(\mathbf{X})\nabla V(\boldsymbol{x})\cdot\delta\boldsymbol{x} + \rho_0(\mathbf{X})\int \rho_0(\mathbf{X})\nabla K(\boldsymbol{x},\boldsymbol{y})\cdot\delta\boldsymbol{x}\mathrm{d}\mathbf{X}\ \ \mathrm{d}\mathbf{X}\mathrm{d}t,\end{aligned}$$

where $\delta\boldsymbol{x}(\mathbf{X},t)$ is the test function satisfying $\tilde{\delta\boldsymbol{x}}\cdot\mathbf{n} = 0$ with $\mathbf{n}$ being the unit outward normal of $\partial\Omega$ and$\tilde{\delta\boldsymbol{x}}(\boldsymbol{x}(\mathbf{X},t),t) = \delta\boldsymbol{x}(\mathbf{X},t)$. We omit the tilde when there is no ambiguity. Pushing forward to the Eulerian frame, we have

$$\begin{aligned}\delta\mathcal{A} &= -\int_0^T \int_\Omega \left(-\omega'(\rho)\rho + \omega\right)\nabla\cdot(\delta\boldsymbol{x}) + (\rho\nabla V + \rho(\nabla_{\boldsymbol{x}} K * \rho))\cdot\delta\boldsymbol{x}\ \mathrm{d}\boldsymbol{x}\mathrm{d}t \\ &= -\int_0^T \int_\Omega \left(\nabla\left(\omega'(\rho)\rho - \omega\right) + \rho\nabla V + \rho(\nabla_{\boldsymbol{x}} K * \rho)\right)\cdot\delta\boldsymbol{x}\ \mathrm{d}\boldsymbol{x}\mathrm{d}t \\ &= -\int_0^T \int_\Omega \rho\nabla\left(\omega'(\rho) + V + (K * \rho)\right)\cdot\delta\boldsymbol{x}\ \mathrm{d}\boldsymbol{x}\mathrm{d}t.\end{aligned}$$

Hence,

$$\frac{\delta \mathcal{A}}{\delta \boldsymbol{x}} = -\rho \nabla \mu, \quad \mu = \frac{\delta \mathcal{F}}{\delta \rho} = \omega'(\rho) + V(\boldsymbol{x}) + K * \rho,$$

where μ is called the chemical potential. Since $\mathcal{D} = \frac{1}{2}\int \eta(\rho)|\mathbf{u}|^2 \mathrm{d}\boldsymbol{x}$, it is easy to compute that $\frac{\delta \mathcal{D}}{\delta \mathbf{u}} = \eta(\rho)\mathbf{u}$. So the force balance condition leads to

$$\eta(\rho)\mathbf{u} = -\rho \nabla \mu, \quad \mu = \omega'(\rho) + V(\boldsymbol{x}) + K * \rho.$$

REFERENCES

[1] S. Adams, N. Dirr, M. Peletier, and J. Zimmer, *Large deviations and gradient flows*, Philos. Trans. Roy. Soc. A, 371 (2013), 20120341.

[2] B. Amos, L. Xu, and J. Z. Kolter, *Input convex neural networks*, in Proceedings of the International Conference on Machine Learning, PMLR, 2017, pp. 146–155.

[3] A. Baron, *Universal approximation bounds for superposition of a sigmoid function*, IEEE Trans. Inform. Theory, 39 (1993), pp. 930–945.

[4] R. Bellman, *Dynamic Programming*, Dover Publications, Mineola, NY, 1957.

[5] J.-D. Benamou and Y. Brenier, *A computational fluid mechanics solution to the Monge-Kantorovich mass transfer problem*, Numer. Math., 84 (2000), pp. 375–393.

[6] J.-D. Benamou, G. Carlier, and M. Laborde, *An augmented lagrangian approach to wasserstein gradient flows and applications*, ESAIM Proc. Surveys, 54 (2016), pp. 1–17.

[7] D. M. Blei, A. Kucukelbir, and J. D. McAuliffe, *Variational inference: A review for statisticians*, J. Amer. Statist. Assoc., 112 (2017), pp. 859–877.

[8] J. Bruna, B. Peherstorfer, and E. Vanden-Eijnden, *Neural Galerkin schemes with active learning for high-dimensional evolution equations*, J. Comput. Phys., 496 (2024), 112588.

[9] J. A. Carrillo, K. Craig, L. Wang, and C. Wei, *Primal dual methods for Wasserstein gradient flows*, Found. Comput. Math., 22 (2022), pp. 389–443.

[10] G. Cybenko, *Approximation by superpositions of a sigmoidal function*, Math. Control Signals Systems, 2 (1989), pp. 303–314.

[11] P.-G. De Gennes and J. Prost, *The Physics of Liquid Crystals*, Internat. Ser. Monogr. Phys. 83, Oxford University Press, 1993.

[12] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, *Bert: Pre-training of Deep Bidirectional Transformers for Language Understanding*, preprint, https://arxiv.org/abs/1810.04805, 2018.

[13] T. Dockhorn, *A Discussion on Solving Partial Differential Equations Using Neural Networks*, preprint, https://arxiv.org/abs/1904.07200, 2019.

[14] M. Doi, *Onsager's variational principle in soft matter*, J. Phys. Condens. Matter, 23 (2011), 284118.

[15] Q. Du and X. Feng, *The phase field method for geometric moving interfaces and their numerical approximations*, Handb. Numer. Anal. 21, Elsevier/North-Holland, Amsterdam, 2020, pp. 425–508.

[16] Y. Du and T. A. Zaki, *Evolutional deep neural network*, Phys. Rev. E, 104 (2021), 045303.

[17] C. Durkan, A. Bekasov, I. Murray, and G. Papamakarios, *Neural spline flows*, in Proceedings of the 33rd Conference on Neural Information Processing Systems, Adv. Neural Inform. Process. Syst., 32, 2019, pp. 7511–7522.

[18] W. E, J. Han, and A. Jentzen, *Algorithms for solving high dimensional pdes: From nonlinear Monte Carlo to machine learning*, Nonlinearity, 35 (2022), pp. 278–310.

[19] W. E, C. Ma, and L. Wu, *Machine learning from a continuous viewpoint,* I, Sci. China Math., 63 (2020), pp. 2233–2266.

[20] W. E, C. Ma, L. Wu, and S. Wojtowytsch, *Towards a mathematical understanding of neural network-based machine learning: What we know and what we don't*, CSIAM Trans. Appl. Math., 1 (2020), pp. 561–615.

[21] W. E and B. Yu, *The deep ritz method: A deep learning-based numerical algorithm for solving variational problems*, Commun. Math. Stat., 6 (2018), pp. 1–12.

[22] B. Eisenberg, Y. Hyon, and C. Liu, *Energy variational analysis of ions in water and channels: Field theory for primitive models of complex ionic fluids*, J. Chem. Phys., 133 (2010), 104104.

[23] J. L. Ericksen, *Introduction to the Thermodynamics of Solids*, Appl. Math. Sci. 275, Springer, 1998.

[24] M.-H. Giga, A. Kirshtein, and C. Liu, *Variational modeling and complex fluids*, in Handbook of Mathematical Analysis in Mechanics of Viscous Fluids, Springer, Cham, 2018, pp. 73–113.
[25] E. D. Giorgi, *Movimenti minimizzanti*, in Proceedings of the Conference on Aspetti e problemi della Matematica oggi, Lecce, 1992.
[26] X. Glorot and Y. Bengio, *Understanding the difficulty of training deep feedforward neural networks*, in Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics, JMLR Workshop and Conference Proceedings, 2010, pp. 249–256.
[27] J. Han, A. Jentzen, and W. E, *Solving high-dimensional partial differential equations using deep learning*, Proc. Natl. Acad. Sci., 115 (2018), pp. 8505–8510, https://doi.org/10.1073/pnas.1718942115.
[28] B. Hanin, *Which neural net architectures give rise to exploding and vanishing gradients?*, in NIPS'18: Proceedings of the 32nd International Conference on Neural Information Processing Systems, 2018, pp. 580–589.
[29] K. He, X. Zhang, S. Ren, and J. Sun, *Deep residual learning for image recognition*, in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2016, pp. 770–778.
[30] K. He, X. Zhang, S. Ren, and J. Sun, *Identity mappings in deep residual networks*, in Proceedings of the European Conference on Computer Vision, Springer, 2016, pp. 630–645.
[31] G. Hinton, L. Deng, D. Yu, G. E. Dahl, A.-r. Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, T. N. Sainath, and B. Kingsbury, *Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups*, IEEE Signal Proc. Mag., 29 (2012), pp. 82–97.
[32] C.-W. Huang, R. T. Chen, C. Tsirigotis, and A. Courville, *Convex Potential Flows: Universal Probability Distributions with Optimal Transport and Convex Optimization*, preprint, https://arxiv.org/abs/2012.05942, 2020.
[33] H. J. Hwang, C. Kim, M. S. Park, and H. Son, *The Deep Minimizing Movement Scheme*, preprint, https://arxiv.org/abs/2109.14851, 2021.
[34] K. Jiang and P. Zhang, *Numerical methods for quasicrystals*, J. Comput. Phys., 256 (2014), pp. 428–440.
[35] R. Jordan, D. Kinderlehrer, and F. Otto, *The variational formulation of the Fokker–Planck equation*, SIAM J. Math. Anal., 29 (1998), pp. 1–17,.
[36] E. F. Keller and L. A. Segel, *Initiation of slime mold aggregation viewed as an instability*, J. Theor. Biol., 26 (1970), pp. 399–415.
[37] E. Kharazmi, Z. Zhang, and G. E. Karniadakis, *hp-vpinns: Variational physics-informed neural networks with domain decomposition*, Comput. Methods Appl. Mech. Engrg., 374 (2021), 113547.
[38] Y. Khoo, J. Lu, and L. Ying, *Solving parametric PDE problems with artificial neural networks*, Eur. J. Appl. Math., 32 (2021), pp. 421–435.
[39] D. P. Kingma, T. Salimans, R. Jozefowicz, X. Chen, I. Sutskever, and M. Welling, *Improved variational inference with inverse autoregressive flow*, in NIPS'16: Proceedings of the 30th International Conference on Neural Information Processing Systems, Adv. Neural Inf. Process. Syst. 29 2016, pp. 4743–4751.
[40] J. F. Kolen and S. C. Kremer, *Gradient flow in recurrent nets: The difficulty of learning longterm dependencies*, IEEE Press, 2001, pp. 237–243, https://doi.org/10.1109/9780470544037.ch14.
[41] A. Krizhevsky, I. Sutskever, and G. E. Hinton, *Imagenet classification with deep convolutional neural networks*, Comm. ACM, 60 (2017), pp. 84–90.
[42] I. E. Lagaris, A. Likas, and D. I. Fotiadis, *Artificial neural networks for solving ordinary and partial differential equations*, IEEE Trans. Neural Networks, 9 (1998), pp. 987–1000.
[43] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, *Gradient-based learning applied to document recognition*, Proc. IEEE, 86 (1998), pp. 2278–2324.
[44] R. Lifshitz and H. Diamant, *Soft quasicrystals–why are they stable?*, Philos. Mag., 87 (2007), pp. 3021–3030.
[45] F. H. Lin and C. Liu, *Static and dynamic theories of liquid crystals*, J. Partial Differential Equations, 14 (2001), pp. 289–330.
[46] S. Lisini, D. Matthes, and G. Savaré, *Cahn–Hilliard and thin film equations with nonlinear mobility as gradient flows in weighted-Wasserstein metrics*, J. Differential Equations, 253 (2012), pp. 814–850.

[47] G. Litjens, T. Kooi, B. E. Bejnordi, A. A. A. Setio, F. Ciompi, M. Ghafoorian, J. A. Van Der Laak, B. Van Ginneken, and C. I. Sánchez, *A survey on deep learning in medical image analysis*, Med. Image Anal., 42 (2017), pp. 60–88.
[48] C. Liu and H. Sun, *On energetic variational approaches in modeling the nematic liquid crystal flows*, Discrete Contin. Dyn. Syst., 23 (2009), pp. 455–475.
[49] C. Liu, C. Wang, and Y. Wang, *A structure-preserving, operator splitting scheme for reaction-diffusion equations with detailed balance*, J. Comput. Phys., 436 (2021), 110253.
[50] C. Liu and Y. Wang, *On Lagrangian schemes for porous medium type generalized diffusion equations: A discrete energetic variational approach*, J. Comput. Phys., 417 (2020), 109566.
[51] C. Liu and Y. Wang, *A variational Lagrangian scheme for a phase-field model: A discrete energetic variational approach*, SIAM J. Sci. Comput., 42 (2020), pp. B1541–B1569, https://doi.org/10.1137/20M1326684.
[52] L. Lu, X. Meng, Z. Mao, and G. E. Karniadakis, *DeepXDE: A deep learning library for solving differential equations*, SIAM Rev., 63 (2021), pp. 208–228, https://doi.org/10.1137/19M1274067.
[53] Y. Lu, J. Lu, and M. Wang, *A priori generalization analysis of the deep Ritz method for solving high dimensional elliptic partial differential equations*, in Proceedings of the Conference on Learning Theory, PMLR, 2021, pp. 3196–3241.
[54] T. Müller, B. McWilliams, F. Rousselle, M. Gross, and J. Novák, *Neural importance sampling*, ACM Trans. Graphics (TOG), 38 (2019), 145.
[55] J. Noh, Y. Wang, H.-L. Liang, V. S. R. Jampani, A. Majumdar, and J. P. Lagerwall, *Dynamic tuning of the director field in liquid crystal shells using block copolymers*, Phys. Rev. Res., 2 (2020), 033160.
[56] T. Ohta and K. Kawasaki, *Equilibrium morphology of block copolymer melts*, Macromolecules, 19 (1986), pp. 2621–2632.
[57] L. Onsager, *Reciprocal relations in irreversible processes.* I., Phys. Rev., 37 (1931), pp. 405–426.
[58] L. Onsager, *Reciprocal relations in irreversible processes.* II., Phys. Rev., 38 (1931), pp. 2265–2279.
[59] G. Papamakarios, E. T. Nalisnick, D. J. Rezende, S. Mohamed, and B. Lakshminarayanan, *Normalizing flows for probabilistic modeling and inference*, J. Mach. Learn. Res., 22 (2021), 57.
[60] M. A. Peletier, *Variational Modelling: Energies, Gradient Flows, and Large Deviations*, preprint, https://arxiv.org/abs/1402.1990, 2014.
[61] L. Pronzato and W. G. Müller, *Design of computer experiments: Space filling and beyond*, Stat. Comput., 22 (2012), pp. 681–701.
[62] M. Raissi, P. Perdikaris, and G. E. Karniadakis, *Physics Informed Deep Learning (Part* I*): Data-driven Solutions of Nonlinear Partial Differential Equations*, preprint, https://arxiv.org/abs/1711.10561, 2017.
[63] L. Rayleigh, *Some general theorems relating to vibrations*, Proc. Lond. Math. Soc., 4 (1873), pp. 357–368.
[64] D. Rezende and S. Mohamed, *Variational inference with normalizing flows*, in Proceedings of the International Conference on Machine Learning, PMLR, 2015, pp. 1530–1538.
[65] G. M. Rotskoff, A. R. Mitchell, and E. Vanden-Eijnden, *Active Importance Sampling for Variational Objectives Dominated by Rare Events: Consequences for Optimization and Generalization*, preprint, https://arxiv.org/abs/2008.06334, 2020.
[66] A. J. Salgado and S. M. Wise, *Classical Numerical Analysis: A Comprehensive Course*, Cambridge University Press, 2022.
[67] Z. Shen, H. Yang, and S. Zhang, *Nonlinear Approximation via Compositions*, preprint, https://arxiv.org/abs/1902.10170, 2019.
[68] J. Sirignano and K. Spiliopoulos, *DGM: A deep learning algorithm for solving partial differential equations*, J. Comput. Phys., 375 (2018), pp. 1339–1364.
[69] E. G. Tabak and E. Vanden-Eijnden, *Density estimation by dual ascent of the log-likelihood*, Commun. Math. Sci., 8 (2010), pp. 217–233.
[70] B. P. van Milligen, V. Tribaldos, and J. Jiménez, *Neural network differential equation and plasma equilibrium solver*, Phys. Rev. Lett., 75 (1995), pp. 3594–3597.
[71] J. L. Vázquez, *The Porous Medium Equation: Mathematical Theory*, Oxford University Press, 2007.
[72] H. Wang, T. Qian, and X. Xu, *Onsager's variational principle in active soft matter*, Soft Matter, 17 (2021), pp. 3634–3653, https://doi.org/10.1039/D0SM02076A.

[73] Y. Wang, J. Chen, C. Liu, and L. Kang, *Particle-based energetic variational inference*, Stat. Comput., 31 (2021), 34.
[74] Y. Wang and C. Liu, *Some recent advances in energetic variational approaches*, Entropy, 24 (2022), pp. 721.
[75] Y. Wang, C. Liu, P. Liu, and B. Eisenberg, *Field theory of reaction-diffusion: Law of mass action with an energetic variational approach*, Phys. Rev. E, 102 (2020), 062147.
[76] Y. Wang, T.-F. Zhang, and C. Liu, *A two species micro–macro model of wormlike micellar solutions and its maximum entropy closure approximations: An energetic variational approach*, J. Non-Newton. Fluid Medh., 293 (2021), 104559.
[77] Q. Wei, Y. Jiang, and J. Z. Y. Chen, *Machine-learning solver for modified diffusion equations*, Phys. Rev. E, 98 (2018), 053304.
[78] E. Weinan, *Machine Learning and Computational Mathematics*, preprint, https://arxiv.org/abs/2009.14596, 2020.
[79] J. Xu, Y. Li, S. Wu, and A. Bousquet, *On the stability and accuracy of partially and fully implicit schemes for phase field modeling*, Comput. Methods Appl. Mech. Engrg., 345 (2019), pp. 826–853.
[80] S. Xu, P. Sheng, and C. Liu, *An energetic variational approach for ion transport*, Commun. Math. Sci., 12 (2014), pp. 779–789.
[81] S. Xu, Z. Xu, O. V. Kim, R. I. Litvinov, J. W. Weisel, and M. S. Alber, *Model predictions of deformation, embolization and permeability of partially obstructive blood clots under variable shear flow*, J. R. Soc. Interface, 14 (2017).
[82] X. Xu, Y. Di, and M, *Variational method for liquids moving on a substrate*, Phys. Fluids, 28 (2016), 087101.
[83] P. Yue, J. J. Feng, C. Liu, and J. Shen, *A diffuse-interface method for simulating two-phase flows of complex fluids*, J. Fluid Mech., 515 (2004), pp. 293–317.
[84] Y. Zang, G. Bao, X. Ye, and H. Zhou, *Weak adversarial networks for high-dimensional partial differential equations*, J. Comput. Phys., 411 (2020), 109409.