Discrete gradient flow approximations of high dimensional evolution partial differential equations via deep neural networks

Communications in Nonlinear Science and Numerical Simulation 117 (2023) 106893

Contents lists available at ScienceDirect

Communications in Nonlinear Science and Numerical Simulation

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

Research paper

Discrete gradient flow approximations of high dimensional evolution partial differential equations via deep neural networks

Emmanuil H. Georgoulis [a,b,c,*,1], Michail Loulakis [b,c,1], Asterios Tsiourvas [d,1]

[a] School of Mathematical and Computer Sciences, Heriot-Watt University, UK
[b] Department of Mathematics, School of Applied Mathematical and Physical Sciences, National Technical University of Athens, Greece
[c] Institute of Applied and Computational Mathematics, Foundation for Research and Technology-Hellas, Greece
[d] Operations Research Center, Massachusetts Institute of Technology, Cambridge, MA, USA

ARTICLE INFO

Article history:
Received 14 April 2022
Received in revised form 21 August 2022
Accepted 14 September 2022
Available online 13 October 2022

Keywords:
Partial differential equations
Deep neural networks
Numerical analysis

ABSTRACT

We consider the approximation of initial/boundary value problems involving, possibly high-dimensional, dissipative evolution partial differential equations (PDEs) using a deep neural network framework. More specifically, we first propose discrete gradient flow approximations based on non-standard Dirichlet energies for problems involving essential boundary conditions posed on bounded spatial domains. The imposition of the boundary conditions is realized weakly via non-standard functionals; the latter classically arise in the construction of Galerkin-type numerical methods and are often referred to as "Nitsche-type" methods. Moreover, inspired by the seminal work of Jordan, Kinderlehrer, and Otto (JKO) Jordan et al. (1998), we consider the second class of discrete gradient flows for special classes of dissipative evolution PDE problems with non-essential boundary conditions. These JKO-type gradient flows are solved via deep neural network approximations. A key, distinct aspect of the proposed methods is that the discretization is constructed via a sequence of residual-type deep neural networks (DNN) corresponding to implicit time-stepping. As a result, a DNN represents the PDE problem solution at each time node. This approach offers several advantages in the training of each DNN. We present a series of numerical experiments which showcase the good performance of Dirichlet-type energy approximations for lower space dimensions and the excellent performance of the JKO-type energies for higher spatial dimensions.

1. Introduction

The numerical approximation of high-dimensional PDEs via collocation-type approaches using artificial neural network (ANN) architectures has received considerable interest in last years [1–6].

This activity appears to have been, at least partially, fueled by the breakthroughs in the use of DNNs in multiple research domains [7,8]. Nonetheless, in the context of scientific computing at large, the use of artificial neural networks is yet in its early stages.

* Corresponding author at: School of Mathematical and Computer Sciences, Heriot-Watt University, UK.
E-mail addresses: georgoulis@math.ntua.gr (E.H. Georgoulis), loulakis@math.ntua.gr (M. Loulakis), atsiour@mit.edu (A. Tsiourvas).

[1] All authors have contributed equally.

https://doi.org/10.1016/j.cnsns.2022.106893

In the context of numerical solution of partial differential equations (PDEs), multiple neural networks and deep learning approaches have been proposed in recent years. We discuss some recent developments that are close in spirit to the contribution below. In [4] a deep learning minimization of the Dirichlet energy is proposed for the numerical approximation of elliptic PDEs, while in [2], the, so-called *deep Galerkin method* (DGM) (and the related *physics informed neural network* (PINN) approach [3]) advocate the minimization of L_2-norm residuals of the PDE and the boundary/initial conditions in a collocation setting; this activity follows from the now classical work [1]. Further, in [6], the authors consider an alternative loss functional implementing weak imposition of boundary conditions in the classical Nitsche fashion from finite element analysis [9], to alleviate the difficulties observed in [4] to enforce conformity in the solution space. In [10], the difference between the Ritz–Galerkin method and the deep learning methods are studied with a focus on the implicit regularity that the second method imposes. In [11] a penalty-free neural network approach for solving a class of second-order boundary value problems on complex geometries is presented that uses a combination of two neural networks, one for the domain and one for the boundary, to solve the problem. Also, in [12] the authors propose a deep neural approximation for high-dimensional elliptic PDEs with boundary conditions for finite domains.

This work is primarily concerned with the development of a class of DNN-based collocation methods for dissipative evolution PDEs, as well as some performance comparison with known alternatives. The basic idea proposed in this work is the use of loss functions stemming from discretized gradient flows [13]. The proposed methods operate in a time-stepping fashion, by seeking to minimize the DNN parameters for each time interval. To showcase the potential generality of this approach, we consider heat-type equations viewed as L_2-gradient flows of the Dirichlet energy and 2-Wasserstein metric gradient flows of the Gibbs–Boltzmann entropy in the spirit of the seminal work of Jordan, Kinderlehrer and Otto (JKO) [14]. A series of numerical experiments using a variant of the ResNET-type architecture from [2] highlight the good performance of the proposed methods. A comparison of the new scheme with the space–time residual approach of the Deep Galerkin Method from [2] is also presented. The use of the JKO functional was advocated in [15], from which the present work originates. We also note the recent manuscript [16] proposing a fully-connected DNN architecture for various gradient flow functionals.

The new ANN-based collocation time-stepping methods proposed in this work use a different ResNET (with the same architecture) for representing the PDE approximation at each time step. This offers several theoretical and practical advantages compared to 'monolithic' space–time residual minimization methods [2,3,5], at least for low-to-medium dimensional evolution problems. More specifically, the "time-instance" ResNETs can be chosen to have a significantly smaller width and depth, compared to full space–time collocation approaches, resulting in both faster performance and better accuracy: this is supported by numerical comparisons presented below. Moreover, the smoothness of the evolution semigroup facilitates the use of the optimized ResNET parameters at time-step n to be used as an excellent initial guess for the optimization taking place at the next time step $n+1$, thereby reducing significantly the number of epochs required in total. This is a particularly salient feature as the optimization steps initiate with the approximation of the known initial condition of the PDE problem (time step 0), i.e., a simple function fitting step not involving differentiation of the ResNET; this is typically achieved to very high accuracy. Having a very accurate representation of the initial condition gives rise to excellent starting values for the optimization to find the approximation at time step 1 and so on. The superior performance of the proposed method is showcased through a series of comparative numerical experiments with known methods.

To arrive at the above complete numerical framework, several developments are also considered for the various individual terms in the loss functions. In particular, for the L_2-norm gradient flow method, we consider weakly imposed Dirichlet-type boundary conditions through a consistent minimization functional in the spirit of Nitsche (cf. also [6], as well as [17] for the same functional in the context of interface modeling in structural mechanics). A crucial methodological development proposed in this work, which may be of independent interest, is the method of computation of the unknown *penalty parameter* required in the weak imposition of Dirichlet boundary conditions. More specifically, in the context of DNNs, the unknown penalty parameter is impossible to estimate directly via so-called inverse estimates as is done in classical finite element methods. To alleviate this difficulty, we propose the use of an implicitly-defined penalty parameter via the DNN approximation of the PDE solution at the previous time-step. We justify our choice of penalty by performing a coercivity analysis of the respective loss functional. Also, for the JKO-type gradient flow, we consider a Sinkhorn–Knopp approximation of the 2-Wasserstein distance and we find that the resulting ResNET for every time-step is approximating the exact solution to an increasing accuracy concerning the spatial dimension d.

The remainder of this work is structured as follows. In Section 2 we discuss the basic model problem and the (continuous) variational interpretations we shall be concerned with for the design of the approximation algorithms. In Section 3, we discuss various approaches for the design of DNN-based numerical methods and we present the Nitsche functional idea, which is, in turn, used in Section 4 for the design of the *discrete gradient flow deep Nitsche method* proposed in this work. A series of numerical experiments are presented in Section 5. Finally, in Section 6, we present a discrete JKO-type gradient flow for the heat problem, which applies also to other Fokker–Planck type equations.

2. Model problem and variational interpretations

In the functional analytic setting of a Gel'fand triple $\mathcal{V} \subset \mathcal{H} \subset \mathcal{V}^*$, we are primarily concerned with finding an approximate solution $u : (0,T] \times \Omega \to \mathbb{R}$, for a domain $\Omega \subset \mathbb{R}^d$, $d \in \mathbb{N}$, and a $T > 0$, satisfying

$$\begin{aligned} \partial_t u + \mathcal{L}u &= F, && \text{in } (0,T] \times \Omega \\ u(0,\cdot) &= u_0, && \text{in } \Omega \\ u &= g_D, && \text{on } (0,T] \times \Gamma_D, \\ \mathbf{n} \cdot \nabla u &= g_N, && \text{on } (0,T] \times \Gamma_N, \end{aligned} \tag{1}$$

with $\mathcal{L} : \mathcal{V} \to \mathcal{V}^*$ denoting a second-order elliptic self-adjoint partial differential operator admitting a variational minimization interpretation, $\mathbf{n}$ denoting the unit outward normal vector to $\partial\Omega$, $\Gamma_D \cup \Gamma_N = \partial\Omega$, the Dirichlet and Neumann parts of the boundary, respectively. We assume that $u_0 \in \mathcal{H}$, $g \in \partial\mathcal{H}$ is sufficiently smooth to be well-defined in the sense of traces in a suitable space $\partial\mathcal{H}$ in what follows, and $F \in L_2(0,T;\mathcal{V}^*)$, in the usual notation of Böchner spaces. We stress, nevertheless, that $\mathcal{L}$ being self-adjoint and/or of second order is not an essential restriction for the developments concerning L_2-gradient flows below. For simplicity of the exposition, however, we shall confine ourselves to the 'canonical' cases $\mathcal{H} = L_2(\Omega)$, $\partial\mathcal{H} = L_2(\partial\Omega)$, and $\mathcal{V} = H^1(\Omega)$ or $\mathcal{V} = H^1_0(\Omega)$ (using standard notation for Lebesgue and Hilbertian Sobolev spaces) and $\mathcal{L} = -\Delta$, with Δ denoting the Laplacian with respect to the spatial variables. The latter choices give rise to the classical heat problem with various boundary conditions.

Upon considering a subdivision of the time interval $[0,T]$ into time steps $I_k := (t_{k-1}, t_k]$, $k = 1, \dots, N_t$, for $0 =: t_0 < t_1 < \cdots < t_{N_t} := T$, for $N_t \in \mathbb{N}$, (1) can be discretized in time as follows: let $u^0 := u_0$ and, for $k = 1, \dots, N_t$, seek approximations u^k such that

$$\frac{u^k - u^{k-1}}{\tau_k} + \mathcal{L}u^k = F^k, \tag{2}$$

with $\tau_k := t_k - t_{k-1}$, and $F^k := F(t_k, \cdot)$, together with the boundary conditions to ensure closure.

Now, assume that there exists an energy functional $I : \mathcal{V} \to \mathbb{R}$ whose respective Euler–Lagrange operator is given by $\mathcal{L}$. In the 'canonical' case, the classical Dirichlet energy

$$I(w) = \frac{1}{2}\int_\Omega |\nabla w|^2 \, \mathrm{d}x \tag{3}$$

corresponds to the Euler–Lagrange operator $\mathcal{L} = -\Delta$ when $\Gamma_N = \emptyset$, $g_D = 0$, and $F = 0$. Moreover, the minimizer

$$w = \arg\min_{w \in \mathcal{V}} \Big(I(w) - \int_\Omega F w \, \mathrm{d}x \Big)$$

within the space $\mathcal{V} = H^1_0(\Omega)$ is unique from the Lax–Milgram Lemma.

An important, yet now classical, observation for what follows is that (2) is the Euler–Lagrange equation of the following variational minimization problem as follows: for $u^0 := u_0$ and $k = 1, \dots, N_t$, seek approximations u^k defined by

$$u^k = \arg\min_{w \in \mathcal{V}} \left(\frac{1}{2}\|w - u^{k-1}\|^2_{L_2(\Omega)} + \tau_k \Big(I(w) - \int_\Omega F^k w \, \mathrm{d}x \Big) \right). \tag{4}$$

Thus, instead of solving (2) directly, we can instead construct numerical methods based on solving (4) instead.

Another possibility to write a variational minimization problem for (2) for the particular case $\Omega = \mathbb{R}^d$, is the celebrated JKO-functional introduced by Jordan, Kinderlehrer and Otto in [14] for the case where the solution u (and the initial condition u_0) refer to probability density functions: find

$$u^k = \arg\min_{p \in K} \left(\frac{1}{2}\mathcal{W}(u^{k-1}, p)^2 + \tau_k \int_{\mathbb{R}^d} p \log p \, \mathrm{d}x \right), \tag{5}$$

over all $p \in K$, where K is the set of all probability densities on $\mathbb{R}^d$ having finite second moments; here $\mathcal{W}$ represents the 2-*Wasserstein metric* (or *distance*) $\mathcal{W}(\mu_1, \mu_2)$ between two probability measures μ_1, μ_2 on $\mathbb{R}^d$ given by

$$\mathcal{W}(\mu_1, \mu_2) = \left(\inf_{p \in P(\mu_1, \mu_2)} \int_{\mathbb{R}^d \times \mathbb{R}^d} |x - y|^2 p(\mathrm{d}x, \, \mathrm{d}y) \right)^{1/2}; \tag{6}$$

here $P(\mu_1, \mu_2)$ is the set of probability measures on $\mathbb{R}^d \times \mathbb{R}^d$ with marginals μ_1, μ_2. (A probability measure p is in $P(\mu_1, \mu_2)$ if and only if for each Borel subset $A \subset \mathbb{R}^d$, $p(A \times \mathbb{R}^d) = \mu_1(A)$ and $p(\mathbb{R}^d \times A) = \mu_2(A)$ hold.) It is proved in [14] that, given $u^0 \in K$, there exists a unique solution of the scheme (5). Moreover, it is proved that the energy functional is convex and that an appropriate interpolation of the solution to (5) for $k = 1, \dots, N_t$ converges to the unique solution of the heat equation. We note that the result in [14] holds for a more general class of Fokker–Planck type equations by suitably modifying the entropy.

An alternative way of defining the Wasserstein distance is as the infimum taken over all random variables X, Y such that X has distribution μ_1 and Y has distribution μ_2, which is the infimum over all possible couplings of the random variables X and Y, namely,

$$\mathcal{W}^2(\mu_1, \mu_2) = \inf \mathbb{E}[|X - Y|^2]. \tag{7}$$

The above variational principles are possible upon understanding subtle properties of the diffusion operators involved, in conjunction with respective boundary conditions. Alternatively, one can always consider the *space–time* 'least squares' minimization problem related to (1):

$$u = \arg \min_{w \in L_2(0,T;\mathcal{V})} \left(\|\partial_t w + \mathcal{L} w - F\|^2_{L_2(0,T;\mathcal{H})} + \|\alpha w + \beta \mathbf{n} \cdot \nabla w - g\|^2_{L_2(0,T;\partial\mathcal{H})} + \|w(0,\cdot) - u_0\|^2_{\mathcal{H}} \right), \tag{8}$$

which can be used for general classes of PDE problems. We remark that $\mathcal{H}$ here can accept weighted inner products; see [2] for details.

Although we focus on the case of the heat problem for simplicity of the exposition, we stress that more general cases of PDE problems in the form of (1) can be treated by the methodologies presented below. For instance, any dissipative PDE whose spatial operator is the Euler–Lagrange operator of a suitable energy functional $I(\cdot)$ is admissible for representation by (4), while the JKO theory applies to a general family of linear Fokker–Planck equations also, cf., [14].

3. Numerical solution of PDE problems via ANNs

Perhaps the most common paradigm in employing artificial neural networks (ANNs) for the numerical solution of PDE problems is the collocation one, whereby the solution sought is modeled by an appropriate ANN architecture and the ANN parameters are optimized to approximate the exact solution through suitably defined loss functionals.

The recent, yet already highly popular, methodologies of the *Deep Galerkin Method* (DGM) of Sirignano and Spiliopoulos in [2] and the original *Physics Informed Neural Network* (PINN) approach of Raissi, Perdikaris & Karniadakis [3] employ various network architectures to model the approximate solution and minimize through the 'least-squares' space–time variational problem (8). These methods follow the collocation framework originally proposed in [1] and they are flexible in terms of the problems to which they can be applied. At the same time, DGM and PINN approaches may not always capture the fine properties of the respective PDE problems, as they are confined to performing minimization within certain function spaces only. In contrast, using more specialized energy functionals, such as the Dirichlet energy (3) one restricts the class of problems but allows for minimization in more relevant solution spaces for the problem at hand.

For example, if we consider the elementary case of an (elliptic) Poisson problem (i.e., no time variable) with essential boundary conditions:

$$-\Delta u = F \quad \text{in} \quad \Omega, \qquad u = g \quad \text{on} \quad \partial\Omega \tag{9}$$

within DGM or the PINN approach of [3], it is proposed to find an ANN U (of some given architecture) by minimizing the loss functional

$$\|\Delta U + F\|^2_{L_2(\Omega)} + \|U - g\|^2_{L_2(\partial\Omega)} \to \min$$

i.e., minimization of the PDE residual takes place in the L_2-norm and the boundary conditions are also enforced in the corresponding sense. In contrast, using the Dirichlet energy instead, we can compute another ANN approximation $\tilde{U}$ by minimizing:

$$I(\tilde{U}) - \int_\Omega F\tilde{U}\,\mathrm{d}x = \int_\Omega \left(\frac{1}{2}|\nabla\tilde{U}|^2 - F\tilde{U} \right) \mathrm{d}x \to \min \tag{10}$$

for $\tilde{U} = g$ on $\partial\Omega$. Standard variational calculus arguments show that $\tilde{U}$ is aiming to minimize the error $\|\nabla(u - \tilde{U})\|_{L_2(\Omega)}$ or, equivalently, the functional

$$\|\nabla(u - \tilde{U})\|^2_{L_2(\Omega)} = \|\Delta\tilde{U} + F\|^2_{H^{-1}(\Omega)} \to \min$$

under the condition $\tilde{U} = g$ on $\partial\Omega$, i.e., minimization takes place in a weaker norm. This may be advantageous in many situations of interest and/or higher-dimensional problems. The strategy of considering optimization of ANNs under (10) constitutes the, so-called, *Deep Ritz Method* of E and Yu [4]. A key practical challenge for the deep Ritz method is the requirement for the strong imposition of essential boundary conditions. Strongly imposing Dirichlet-type conditions appear to be introducing some stiffness in simulations due to the global nature of the ANN approximants.

The issue of imposition of essential boundary conditions in energy minimization problems of the type (10) can be addressed by altering the loss functional to facilitate the weak imposition of essential boundary conditions. More specifically, by seeking ANN minimizer $\hat{U}$ of the modified Dirichlet energy functional

$$\int_\Omega \left(\frac{1}{2}|\nabla\hat{U}|^2 - F\hat{U} \right) \mathrm{d}x - \int_{\partial\Omega} \mathbf{n} \cdot \nabla\hat{U}(\hat{U} - g)\,\mathrm{d}s + \int_{\partial\Omega} \frac{\gamma}{2}(\hat{U} - g)^2\,\mathrm{d}s \to \min \tag{11}$$

for $\hat{U} \in H^1(\Omega)$, involving a so-called, *penalty parameter* γ, which should be defined by the user. Notice that when $\hat{U} = g$ the last two terms of the last functional disappear. The second term on the right-hand side of the functional in (11) is *not* sign-definite; yet its inclusion is important to retain the consistency of (11) with respect to the PDE problem (9). To facilitate convexity of the functional, at least in finite-dimensional subspaces of $H^1(\Omega)$, the third term in (9) is included. To ensure successful 'stabilization' (i.e., coercivity in a suitable norm), the penalty parameter γ has to be chosen appropriately; this means in practical terms that it should be chosen large enough to 'stabilize' effectively but not too large to introduce stiffness in the numerical method. Note that, formally, we expect $\hat{U} \to \tilde{U}$ as $\gamma \to \infty$.

4. A discrete gradient flow deep Nitsche method

Approximating an elliptic boundary value problem by minimizing (11) for a residual-type network $\hat{U}$ was discussed in [6]; this method is referred to as the *deep Nitsche method*. To carry out the error analysis in [6], the validity of a Bernstein-type inequality (trace inverse estimate) for ANNs was assumed which, in turn, determines the penalty parameter γ. Given the parameter-dependent growth of ANNs, it is not clear if such an assumption is realistic or not. Here, we will propose a practical rectification of this state of affairs, which can give a concrete choice of the penalty parameter γ, which appears to be effective in stabilizing the method in all numerical examples we have performed.

Further, the new practical version of minimization using (11) mentioned above will be combined with (4) to establish a new ANN framework, based on deep residual-type architectures, for the approximation of solutions to (1). The approximate solution will be, therefore, expressed as a sequence of ANNs $\hat{U}^k$, each representing the approximate solution in the time interval I_k. This point of view has several potential salient features compared to the generic 'monolithic' space–time residual approximation proposed in [2,3]. In particular, the minimization takes place in the natural $L_2(H^1)$-norm setting, and the boundary conditions are treated weakly to alleviate numerical stiffness. Furthermore, starting from fitting an ANN approximation of the known initial condition of the PDE problem, not involving differentiation of the network, the method progresses iteratively in computing $\hat{U}^k$, $k = 1, \dots, N_t$, having the same architecture as the initial condition ANN. As such, we can use the ANN parameters fitted in the previous time step as an initial guess for the optimization in the current one; this leads to a significant reduction in epochs needed to achieve a given accuracy. The ANN architecture used in the numerical experiments follows the residual architecture proposed by Sirignano and Spiliopoulos in [2].

4.1. Method definition

We consider the PDE problem: find $u \in L_2(0, T; H^1(\Omega))$ such that:

$$\begin{aligned} \partial_t u - \nabla \cdot (A \nabla u) &= F, && \text{in } (0,T] \times \Omega, \\ u(0, \cdot) &= u_0, && \text{in } \Omega, \\ u &= g_D, && \text{on } (0,T] \times \Gamma_D, \\ \mathbf{n} \cdot A \nabla u &= g_N, && \text{on } (0,T] \times \Gamma_N, \end{aligned} \tag{12}$$

in the distributional sense, for a symmetric, uniformly positive definite and bounded diffusion tensor $A \in [L^\infty(\Omega)]^{d \times d}$, viz., there exist $\lambda_1, \lambda_d > 0$ (the smallest and the largest eigenvalue, respectively,) such that

$$\lambda_1 |\xi|^2 \le \xi^T A \xi \le \lambda_d |\xi|^2, \quad \text{for all } \xi \in \mathbb{R}^d. \tag{13}$$

We select a subdivision of the time-interval $[0, T]$, reading $0 = t_0 < t_1 < \cdots < T_{N_t}$ and we seek approximations u^k to $u(t_k, \cdot)$ given by

$$u^k = \arg \min_{w \in H^1(\Omega)} \Big(\frac{1}{2} \| w - u^{k-1} \|^2_{L_2(\Omega)} + \tau_k N(w) \Big), \tag{14}$$

with

$$N(w) := \int_\Omega \Big(\frac{1}{2} |\sqrt{A} \nabla w|^2 - F w \Big) \, \mathrm{d}x - \int_{\Gamma_D} \mathbf{n} \cdot A \nabla w (w - g_D) \, \mathrm{d}s + \int_{\Gamma_D} \frac{\sigma}{2} (w - g_D)^2 \, \mathrm{d}s - \int_{\Gamma_N} g_N w \, \mathrm{d}s.$$

We note carefully, that the second and third terms in the definition of N vanish when N is evaluated at the exact solution of (12); their presence, however, becomes important when w is not satisfying the boundary condition exactly. In particular, the second term ensures the consistency of the functional with respect to the PDE problem (12), while the third term penalizes w when its value on the Dirichlet part of the boundary Γ_D differs from the Dirichlet boundary condition. A similar in spirit, functional has appeared within the context of minimization problems in the presence of interfaces in [17]. We stress that more general classes of problems can be treated in this context, e.g., nonlinear reaction–diffusion equations with mixed boundary conditions, etc. We refrain from providing the full generality here to focus on the key mathematical and algorithmic issues.

A crucial aspect of any deep learning based algorithm is its *architecture*: clever choices of architectures, may help exploit a prior knowledge about the application and therefore improve the performance. In [2], the authors propose an interesting architecture, similar to Recurrent Neural Networks (RNNs) and, more specifically, similar to the Long Short-Term Memory

(LSTM) architecture [18]. LSTM networks are well-suited for time (and space) dependent data, since there can be lags of unknown duration between important events. Furthermore, LSTMs deal with the vanishing gradient problem that can be encountered when training traditional RNNs [19]. The shape of the solution $u(t,x)$ for $t < T$, although smooth, to the presence of diffusion, may rapidly change in certain spatial regions. For d_{in} as input dimension and d_{out} as output dimension, we consider $\boldsymbol{x} \in \mathbb{R}^{d_{in}}$, $W^1 \in \mathbb{R}^{n_1 \times d_{in}}$, $b^1 \in \mathbb{R}^{n_1}$, $U^{\cdot,l} \in \mathbb{R}^{n_{l+1} \times d_{in}}$, $W^{\cdot,l} \in \mathbb{R}^{n_{l+1} \times n_l}$, $b^{\cdot,l} \in \mathbb{R}^{n_l}$, for $l = 1, \dots, L$, and $W \in \mathbb{R}^{d_{out}, n_{L+1}}$, $b \in \mathbb{R}^{d_{out}}$, where n_l is the dimension of S_l that is equal to the number of neurons at each block. The architecture proposed in [2] reads:

$$
\begin{aligned}
S^1 &= \sigma(W^1 \boldsymbol{x} + b^1) \\
Z^l &= \sigma(U^{z,l} \boldsymbol{x} + W^{z,l} S^l + b^{z,l}), \quad l = 1, \dots, L \\
G^l &= \sigma(U^{g,l} \boldsymbol{x} + W^{g,l} S^l + b^{g,l}), \quad l = 1, \dots, L \\
R^l &= \sigma(U^{r,l} \boldsymbol{x} + W^{r,l} S^l + b^{r,l}), \quad l = 1, \dots, L \\
H^l &= \sigma(U^{h,l} \boldsymbol{x} + W^{h,l} (S^l \odot R^l) + b^{h,l}), \quad l = 1, \dots, L \\
S^{l+1} &= (1 - G^l) \odot H^l + Z^l \odot S^l, \quad l = 1, \dots, L \\
f(t, \boldsymbol{x}; \theta) &= W S^{L+1} + b
\end{aligned}
\tag{15}
$$

where $\boldsymbol{x} = (t, x)$, the number of blocks are $L + 1$ and $\odot$ is the element-wise multiplication operator. The parameters of this architecture are:

$$
\theta = \left\{ W^1, b^1, \left(U^{z,l}, W^{z,l}, b^{z,l}\right)_{l=1}^{L}, \left(U^{g,l}, W^{g,l}, b^{g,l}\right)_{l=1}^{L}, \left(U^{r,l}, W^{r,l}, b^{r,l}\right)_{l=1}^{L}, \left(U^{h,l}, W^{h,l}, b^{h,l}\right)_{l=1}^{L}, W, b \right\}
$$

Each block parameter consists of M neurons/units with element-wise non-linearity function $\sigma : \mathbb{R}^M \to \mathbb{R}^M$ defined as $\sigma(z) = \big(\phi(z_1), \dots, \phi(z_M)\big)$ where $\phi : \mathbb{R} \to \mathbb{R}$ is the activation function.

The aforementioned mathematical description of the architecture appears to be complicated at first sight. Each layer takes as an input the batch inputs $\boldsymbol{x}$ (for instance, a set of randomly sampled time-space points) and the output S^l of the previous layer/block. The final block S^{L+1} passes through a linear transformation to produce the final output $y = f(t, \boldsymbol{x}; \theta)$. Compared to a Multilayer Perceptron (MLP) of the same number of neurons/units, the number of parameters in each hidden layer of the above architecture is approximately eight times greater than the same number in a common dense layer. This observation derives from the fact that each layer in (15) has 8 weight matrices and 4 bias vectors.

4.2. The choice of the penalty parameter γ

A crucial aspect of the success of the proposed method is the judicious choice of the penalty parameter γ. In the context of finite element methods with piecewise polynomial approximants on triangulations and Nitsche-type imposition of boundary conditions, the choice of the penalty parameter is determined by the constant of, so-called, *inverse estimates*. To showcase the use of such estimates, we consider momentarily a triangulation $\mathcal{T}(\mathcal{N})$ of Ω into mutually non-overlapping d-dimensional simplices with $\mathcal{N}$ denoting the set of nodes (vertices) with respect to which the triangulation is defined. (If Ω has curved boundaries we approximate the domain, as is standard in finite element methods.) Let also S_h be the space of continuous piecewise polynomial functions of degree at most p, subordinate to the triangulation. Then, there exists a constant $C > 0$, independent of the triangulation granularity, (depending though on the polynomial degree and, possibly on elemental shapes,) such that the estimate

$$
\|\sqrt{h} v\|_{L^2(\Gamma_D)} \leq C \|v\|_{L^2(\Omega)}, \tag{16}
$$

holds, with h the local mesh function, representing the local maximum distance between nodes in the triangulation. In particular, in the vicinity of the boundary Γ_D, h is proportional to the distance of the locally nearest non-boundary node from Γ_D. We now describe why inverse estimates such as (16) prescribe suitable choices for σ, if we seek to minimize (14) within S_h. To that end, we have, respectively, for $u_h \in S_h$,

$$
\begin{aligned}
N(u_h) &\geq \int_\Omega \left(\frac{1}{2} |\sqrt{A} \nabla u_h|^2 - F u_h \right) \mathrm{d}x - \|\sqrt{Ah} \nabla u_h\|_{L^2(\Gamma_D)} \|\sqrt{A} \mathbf{n} | h^{-1/2} (u_h - g_D)\|_{L^2(\Gamma_D)} \\
&\quad + \int_{\Gamma_D} \frac{\sigma}{2} (u_h - g_D)^2 \, \mathrm{d}s - \int_{\Gamma_N} g_N u_h \, \mathrm{d}s \\
&\geq \int_\Omega \left(\frac{1}{4} |\sqrt{A} \nabla u_h|^2 - F u_h \right) \mathrm{d}x + \frac{1}{2} \int_{\Gamma_D} \left(\sigma - \frac{2C^2 \lambda_d^2}{\lambda_1 h} \right) (u_h - g_D)^2 \, \mathrm{d}s - \int_{\Gamma_N} g_N u_h \, \mathrm{d}s,
\end{aligned}
$$

using (13) and (16) by observing that ∇u_h is still a piecewise polynomial over the same triangulation, along with the inequality $\alpha\beta \leq \alpha^2/4 + \beta^2$ for $\alpha, \beta \in \mathbb{R}$ in the last step. Therefore, to ensure the positivity of the quadratic terms and, therefore, uniqueness of the solution, we should select $\sigma \geq 2C^2 \lambda_d^2 / (\lambda_1 h)$. At the other end of the spectrum, the penalty parameter σ should *not* be chosen 'too' large, as this will effectively annihilate the advantages brought by the use of weak imposition of boundary conditions.

Unfortunately, inverse estimates such as (16) are not available for general (deep) ANNs and, therefore, it is not *a priori* clear how to select γ in (14) for ANN functions. Nevertheless, a good choice of the penalty parameter γ is essential for the success of the proposed method. To address this we now provide a 'semi-heuristic' construction to facilitate a practical choice for γ. Following the standard practice, in the implementation below, we approximate the integrals in the minimization principles by quadrature rules based on random point clouds $\mathcal{N}_I := \{x_n^I\}_{n=1}^{\mathrm{N}_I}$ contained in Ω according to a uniform distribution, $\mathcal{N}_D := \{x_n^D\}_{n=1}^{\mathrm{N}_D} \subset \Gamma_D$ and $\mathcal{N}_N := \{x_n^N\}_{n=1}^{\mathrm{N}_N}$; in practice, we select the uniform distribution, unless there is a specific modeling reason to make a different choice. We note that one of the two boundary point clouds may be void if the respective boundary condition is not enforced. For uniformly distributed point cloud distributions, we approximate the integrals, as is standard, in a Monte Carlo fashion:

$$\int_\Omega f\,\mathrm{d}x \approx \frac{|\Omega|}{\mathrm{N}_I}\sum_{n=1}^{\mathrm{N}_I} f(x_n^I), \qquad \int_{\Gamma_D} g\,\mathrm{d}s \approx \frac{|\Gamma_D|}{\mathrm{N}_D}\sum_{n=1}^{\mathrm{N}_D} g(x_n^D), \qquad \int_{\Gamma_N} g\,\mathrm{d}s \approx \frac{|\Gamma_N|}{\mathrm{N}_N}\sum_{n=1}^{\mathrm{N}_N} g(x_n^N).$$

Using the above, we define the approximate Nitsche functional $\tilde{N}(w, w) \approx N(w)$, given by

$$\begin{aligned}\tilde{N}(w; v) := {} & \frac{|\Omega|}{\mathrm{N}_I}\sum_{n=1}^{\mathrm{N}_I}\Big(\frac{1}{2}\big|\sqrt{A}(x_n^I)\nabla w(x_n^I)\big|^2 - F(x_n^I)w(x_n^I)\Big)\\ & - \frac{|\Gamma_D|}{\mathrm{N}_D}\sum_{n=1}^{\mathrm{N}_D}\big(\mathbf{n}(x_n^D)\cdot A(x_n^D)\nabla w(x_n^D)(w(x_n^D) - g_D(x_n^D))\big)\\ & + \frac{|\Gamma_D|}{\mathrm{N}_D}\sum_{n=1}^{\mathrm{N}_D}\frac{\gamma(x_n^D; v)}{2}(w(x_n^D) - g_D(x_n^D))^2 - \frac{|\Gamma_N|}{\mathrm{N}_N}\sum_{n=1}^{\mathrm{N}_N} g_N(x_n^N)w(x_n^N),\end{aligned} \tag{17}$$

and we seek to minimize instead the loss functional given by (14) with $\tilde{N}$ replacing N. Note that we allow the penalty γ to depend independently on an ANN v, which may differ from w; this is non-standard and will be discussed in detail below.

A key attribute of the proposed approach is the estimation of the penalty parameter σ, which we now consider. Standard estimation of the indefinite term yields

$$\begin{aligned}&\Big|\frac{|\Gamma_D|}{\mathrm{N}_D}\sum_{n=1}^{\mathrm{N}_D}\big(\mathbf{n}(x_n^D)\cdot A(x_n^D)\nabla w(x_n^D)(w(x_n^D) - g_D(x_n^D))\big)\Big|\\ &\le \frac{|\Omega|}{\mathrm{N}_I}\sum_{n=1}^{\mathrm{N}_D}\frac{|\Gamma_D|\mathrm{N}_I\lambda_d^2}{|\Omega|\mathrm{N}_D\tau(x_n^D)}\big|\nabla w(x_n^D)\big|^2 + \frac{|\Gamma_D|}{\mathrm{N}_D}\sum_{n=1}^{\mathrm{N}_D}\frac{\tau(x_n^D)}{4}(w(x_n^D) - g_D(x_n^D))^2,\end{aligned} \tag{18}$$

for any $\tau(x_n^D) > 0$, $x_n^D \in \mathcal{N}_D$. Now, let $y_n \equiv y_n(x_n^D) \in \mathcal{N}_I$ denote the nearest point to $x_n^D \in \mathcal{N}_D$, such that $|\nabla w(y_n)| \neq 0$, that has not been selected in a previous boundary point. Combining (18), along with the uniform ellipticity, and selecting

$$\gamma(x_n^D; w) \ge \tau(x_n^D) := 8\frac{|\Gamma_D|\mathrm{N}_I\lambda_d^2}{|\Omega|\mathrm{N}_D\lambda_1}\frac{\big|\nabla w(x_n^D)\big|^2}{\big|\nabla w(y_n)\big|^2}, \tag{19}$$

noting the dependence of γ on w, we deduce

$$\begin{aligned}\tilde{N}(w; w) \ge {} & \frac{|\Omega|}{\mathrm{N}_I}\sum_{n=1}^{\mathrm{N}_I}\Big(\frac{1}{4}\big|\sqrt{A}(x_n^I)\nabla w(x_n^I)\big|^2 - F(x_n^I)w(x_n^I)\Big)\\ & + \frac{|\Gamma_D|}{\mathrm{N}_D}\sum_{n=1}^{\mathrm{N}_D}\frac{\gamma(x_n^D; w)}{4}(w(x_n^D) - g_D(x_n^D))^2 - \frac{|\Gamma_N|}{\mathrm{N}_N}\sum_{n=1}^{\mathrm{N}_N} g_N(x_n^N)w(x_n^N),\end{aligned} \tag{20}$$

with the second argument in $\tilde{N}$ referring to the (nonlinear) dependence of γ on w.

Therefore, the coercivity of $\tilde{N}(w; w)$ is possible upon selecting the penalty γ according to (19). Note that, as it stands, the choice of γ is *nonlinear* with respect to the parameters θ. This is non-standard in Nitsche-type/weak imposition of boundary conditions. Here, however, we use this concept to introduce some quantitative information on the size of γ, which is crucial for the success of the method as we shall see below. In a practical method, to avoid back-propagating the additional w-nonlinearity introduced in the penalty γ, in the gradient descent step we use the value $u^k(\theta_{m-1})$ to evaluate the penalty instead, viz., we implement $\tilde{N}(u^k(\theta_m); u^k(\theta_{m-1}))$ at the current step; we refer to Algorithm 1 for details of the implemented process. Finally, we note that the values θ_0 used are the ones from the previous time-step.

5. Numerical experiments

We will now present a comprehensive numerical experiment showcasing the performance of the proposed "Nitsche-type" method used to approximate solutions to linear heat-type parabolic problems for $d \in [2, 20]$. More specifically, we will demonstrate the good performance of the combined use of the functional $\tilde{N}$ above with the choice of penalty given by (19) in conjunction with the architecture described in (15).

We evaluate the performance of the time deep Nitsche method where for $k = 1, \dots, N_t$ we compute ANN u^k using

$$u^k = \arg \min_{w \in H^1(\Omega)} \left(\frac{|\Omega|}{2\mathrm{N}_I} \sum_{n=1}^{\mathrm{N}_I} (w - u^{k-1})^2(x_n^I) + \tau_k \tilde{N}(w; w) \right). \tag{21}$$

Therefore, in contrast to the DGM/PINN approach [2,3,5], whereby a unique 'space–time' ANN/ResNET is computed, the present method constructs one ResNET at each time-step. As a result, the computed "time-instance" ResNETs can be chosen to have a significantly smaller width and depth, compared to full space–time collocation approaches, resulting in faster performance. Moreover, the optimized ResNET parameters at time-step $k-1$ can be used as excellent initializations for the next time step k, thereby reducing significantly the number of epochs required. Note that this process starts at $k = 0$ with the approximation of the initial condition, i.e., a simple function fitting step not involving differentiation of the ResNET; this is typically achieved to very high accuracy. As a result, we expect the approximation u^1 to be also good, and so on. The pseudocode of the algorithm for the time deep Nitsche method is given in Algorithm 1.

Algorithm 1 The Time Deep Nitsche Method

Initialize randomly the parameter set θ_0 and set the learning rate schedule a_n. Initialize time step τ.
Initialize ANN approximating the initial condition $u^0 = \arg\min_{w \in H^1(\Omega)} \|w - u_0\|^2_{L_2(\Omega)}$.
for $k = 1, 2, \dots, N_t$ **do**
  Create ANN u^k equal to u^{k-1} from the previous time step.
  for each epoch $m = 1, 2, \dots$ **do**
    Generate point clouds $\mathcal{N}_I := \{x_n^I\}_{n=1}^{\mathrm{N}_I} \subset \Omega$, $\mathcal{N}_D := \{x_n^D\}_{n=1}^{\mathrm{N}_D} \subset \Gamma_D$, and $\mathcal{N}_N := \{x_n^N\}_{n=1}^{\mathrm{N}_N} \subset \Gamma_N$.
    Calculate $L(\theta_m) := \frac{|\Omega|}{\mathrm{N}_I} \sum_{n=1}^{\mathrm{N}_I} (u^k(\theta_m) - u^{k-1})^2(x_n^I) + \tau_k \tilde{N}(u^k(\theta_m); u^k(\theta_{m-1}))$.
    Take a descent step at the random batch of generated points using SGD: $\theta_{m+1} = \theta_m - a_n \nabla_\theta L(\theta_m)$.
    Exit when $||\theta_{m+1} - \theta_m||$ is small enough.
  end for
end for
Return u^k solution for every time step.

To test the Deep Nitsche Method, we consider (12) with $A = I_{d\times d}$, and u_0 and f so that

$$u(t, \boldsymbol{x}) = \sin(t) \sin(x_1 + x_2 + \cdots + x_d) \tag{22}$$

is the exact solution, with $\boldsymbol{x} = [x_1, x_2, \dots, x_d]^T \in [0, 1]^d =: \Omega$, for $t \in (0, 1]$ and d denotes the spatial dimensions; we take $d = 2, 3, 5, 10$ and 20. Finally, we consider a uniform time-step $\tau = 0.01$.

To realize the method, we uniformly sample $600d$ points from the domain and 600 from each boundary edge, using 3 blocks with 50 neurons each, and with $ReLU(\cdot)$ activation. We select the penalty at each iteration k to be

$$\gamma^k = \max_{x_n^D \in \mathcal{N}_D} 500 \cdot \frac{|\Gamma_D| \mathrm{N}_I \lambda_d^2}{|\Omega| \mathrm{N}_D \lambda_1} \frac{|\nabla w(x_n^D)|^2}{|\nabla w(y_n)|^2}. \tag{23}$$

The weights of the initial condition are initialized using Xavier initialization [20]. The initial network is trained for 50,000 epochs for u^1, using the following learning rate schedule

$$lr(epochs) = \begin{cases} 10^{-2}, \, 1 \le epochs < 10{,}000 \\ 10^{-3}, \, 10{,}000 \le epochs < 40{,}000 \\ 10^{-4}, \, epochs \ge 40{,}000. \end{cases} \tag{24}$$

The subsequent networks u^k are trained via 2000 epochs $k = 1, \dots, N_t := 1/\tau$ with a constant learning rate equal to 10^{-3} for the first 500 epochs and 10^{-4} for the rest. The performance of the method in the above problem is presented in Table 1. We note the 'scalability' of the method with respect to the dimension d in terms of errors: the metrics taken for $d = 2$ are comparable to metrics taken for $d = 20$.

To assess the potential practical advantages of the proposed time deep Nitsche Method, we compare it against the general purpose DGM approach in which a discrete version of (8) is solved for the same PDE problem. For the solution we use uniformly sampled $600 \cdot (d+1)$ points from the domain and 600 from each boundary edge, using 3 blocks with 50 neurons each, along with the $ReLU(\cdot)$ activation function, the weights are initialized via the Xavier initialization [20] and the network is trained for 200,000 epochs; the above are in accordance to the implementation in [2]. We use the same

Table 1
Results from the time deep Nitsche method with penalty chosen as in (23).

d	L^2 relative error	max error	mean error
2	$1.6e{-2}$	$9.3e{-3}$	$2.9e{-3}$
3	$4.7e{-3}$	$7.2e{-3}$	$1.3e{-3}$
5	$2.0e{-3}$	$1.5e{-3}$	$2.9e{-4}$
10	$3.5e{-3}$	$1.9e{-3}$	$2.4e{-4}$
20	$4.2e{-3}$	$3.4e{-3}$	$3.7e{-4}$

Table 2
Results from the Deep Galerkin Method.

d	L^2 relative error	max error	mean error
2	$9.9e{-2}$	$8.7e{-2}$	$3.4e{-2}$
3	$9.3e{-2}$	$1.0e{-1}$	$3.2e{-2}$
5	$1.2e{-1}$	$1.2e{-1}$	$2.8e{-2}$
10	$4.5e{-1}$	$2.9e{-1}$	$1.5e{-1}$
20	$4.7e{-1}$	$4.2e{-1}$	$1.3e{-1}$

learning rate and the same total number of epochs to make a fair comparison. The performance of the DGM approach is given in Table 2.

By observing the tables we observe that the time deep Nitsche method outperforms DGM by an order of magnitude for this particular PDE problem. Moreover, the time deep Nitsche method appears to perform equally well as the input dimension increases, while the performance of the DGM drops as the dimension increases. This behavior is persistent with respect to the inherent randomness of the optimization steps. We conclude, therefore, that, at least for this particular PDE problem and with the above architectures and method parameters the time deep Nitsche method is at least as competitive as DGM/PINN while using significantly fewer computational resources in the optimization step.

6. A discrete gradient flow based on the JKO functional

Finally, for the case $F = 0$, $\Gamma_D = \emptyset$, $g_N = 0$, i.e., for the pure homogeneous Neumann problem, we also present results from an implementation of seeking approximations based on minimizing the JKO-functional (5). The challenge of evaluating the 2-Wasserstein distance is addressed via the Sinkhorn–Knopp Algorithm [21]. More specifically, we use the algorithm proposed in [22]. The discretization scheme is the following

$$\frac{1}{2}\mathcal{W}^2(p^{k-1}, p) + h\int_{\Omega} p\log p \tag{25}$$

over the appropriate class of densities $p \in K$ where K is the set of all probability densities in Ω having finite second moments and $\mathcal{W}(\cdot,\cdot)$ the 2-Wasserstein distance. Based upon the aforementioned scheme we can derive a learning algorithm for neural networks that solves the heat equation. The algorithm is given below.

Algorithm 2 The Deep Wasserstein Method

Initialize randomly the parameter set θ_0 and set the learning rate schedule a_n. Initialize time step τ.
Initialize ANN approximating the initial condition $u^0 = \arg\min_{w\in H^1(\Omega)} \|w - u_0\|^2_{L_2(\Omega)}$.
for $k = 1, 2, \ldots, N_t$ **do**
  Create ANN u^k equal to u^{k-1} from the previous time step.
  for each epoch $m = 1, 2, \ldots$ **do**
    Generate interior point cloud $\mathcal{N}_I := \{x_n^I\}_{n=1}^{N_I} \subset \Omega$.
    Calculate $L(\theta_m) = \frac{1}{2}d(u^{k-1}, u^k(\theta_m))^2 + \tau\int_{\Omega} u^k(\theta_m)\log u^k(\theta_m)$.
    Take a descent step at the random batch of generated points using SGD: $\theta_{m+1} = \theta_m - a_n\nabla_\theta L(\theta_m)$.
    Exit when $||\theta_{m+1} - \theta_m||$ is small enough.
  end for
end for
Return u^k solution for every time step.

We use our method to solve a parabolic type PDE with Neumann boundary conditions. The problem has the following analytical solution

$$u(t, \boldsymbol{x}) = \frac{1}{2}(e^{-d\pi^2 t}\prod_{i=1}^{d}\cos(\pi x_i) + 2), \tag{26}$$

Table 3
Results for the deep Wasserstein method.

d	L^2 Relative Error	Maximum Error	Mean Error
2	8.7e−2	1.0e−1	8.5e−2
3	1.7e−1	4.8e−1	1.3e−1
5	8.6e−2	4.2e−1	5.3e−2
10	8.2e−3	3.9e−2	6.8e−3
20	4.0e−3	6.8e−3	4.0e−3
40	2.1e−3	4.2e−3	1.9e−3
50	2.5e−3	1.6e−2	2.5e−3

Table 4
Results for the Deep Galerkin Method.

d	L^2 relative error	max error	mean error
2	3.7e−2	7.3e−2	3.2e−2
3	3.4e−2	6.8e−2	3.1e−2
5	3.7e−2	7.3e−2	3.4e−2
10	6.3e−2	9.0e−2	6.3e−2
20	1.2e−1	1.5e−1	1.2e−1
40	3.6e−1	7.5e−1	3.4e−1
50	3.6e−1	9.9e−1	3.5e−1

where $\boldsymbol{x} = [x_1, x_2, \ldots, x_d]^T \in [0, 1]^d$. We solve this problem for $d = 2, 3, 5, 10, 20, 40$ and 50 dimensions for $t \in [0, T = 1]$ and $\tau = h = 0.01$.

We uniformly sample $100 \cdot d$ points from the domain, we used 3 blocks with 50 neurons each, we applied as activation function the $ReLU(\cdot)$ and we initialized the weights using Xavier initialization [20]. The initial network is trained for 4000 epochs for u^0, using the following learning rate schedule

$$lr(epochs) = \begin{cases} 10^{-3}, 1 \leq epochs < 2{,}000 \\ 10^{-4}, epochs \geq 2{,}000. \end{cases} \tag{27}$$

The subsequent networks u^k are trained for 100 epochs with constant learning rate equal to 10^{-5}. The performance of this method is given in Table 3. By observing the results, we conclude that the deep Wasserstein method solves the particular PDE efficiently, especially in higher dimensions in relatively few epochs.

We now compare the deep Wasserstein method against the general purpose DGM approach for the same PDE problem. For the solution we use uniformly sampled $100 \cdot (d + 1)$ points from the domain and 100 from each boundary edge, using 3 blocks with 50 neurons each, along with the $ReLU(\cdot)$ activation function. The weights are initialized via the Xavier initialization [20] and the network is trained for 15,000 epochs; the above is in accordance with the implementation in [2]. We use the same learning rate and the same total number of epochs with the deep Wasserstein method in an effort to make a fair comparison. The performance of the DGM approach is given in Table 4.

We observe that the time deep Nitsche method outperforms DGM for input dimension $d > 5$ by an order of magnitude for this particular PDE problem. This behavior is, generally speaking, persistent with respect to the random nature of the optimization step. The deep Wasserstein method appears to be robust with respect to the space dimension, while the performance of the DGM appears to degenerate as the dimension increases. We conclude, therefore, that, at least for this particular PDE problem and with the above architectures and method parameters the deep Wasserstein method is at least as competitive as DGM/PINN for high dimensional problems. Finally, we note that it would be interesting to also examine the different functionals presented in [16] using the ResNET architecture employed in this work.

Declaration of competing interest

The authors declare the following financial interests/personal relationships which may be considered as potential competing interests: Emmanuil Georgoulis reports financial support was provided by Helenic Foundation of Research and Innovation. Emmanuil Georogulis reports financial support was provided by The Leverhulme Trust. Emmanuil Georgoulis reports financial support was provided by Engineering and Physical Sciences Research Council. Michail Loulakis reports financial support was provided by Helenic Foundation of Research and Innovation.

Data availability

No data was used for the research described in the article.

Acknowledgments

This research work was supported by the Hellenic Foundation for Research and Innovation, Greece (H.F.R.I.) under the "First Call for H.F.R.I. Research Projects to support Faculty members and Researchers and the procurement of high-cost research equipment grant" (Project Numbers: 3270, 1034 and 2152). Also, EHG wishes to acknowledge the financial support of The Leverhulme Trust (grant number RPG-2021-238) and of EPSRC, UK (grant number EP/W005840/1).

References

[1] Lagaris Isaac E, Likas Aristidis, Fotiadis Dimitrios I. Artificial neural networks for solving ordinary and partial differential equations. IEEE Trans Neural Netw 1998;9(5):987–1000.
[2] Sirignano Justin, Spiliopoulos Konstantinos. DGM: A deep learning algorithm for solving partial differential equations. J Comput Phys 2018;375:1339–64.
[3] Raissi M, Perdikaris P, Karniadakis GE. Physics-informed neural networks: a deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. J Comput Phys 2019;378:686–707.
[4] E Weinan, Yu Bing. The Deep Ritz method: A deep learning-based numerical algorithm for solving variational problems. 2017, arXiv:1710.00211, [cs, stat].
[5] Hutzenthaler Martin, Jentzen Arnulf. Numerical approximations of stochastic differential equations with non-globally Lipschitz continuous coefficients. Mem Amer Math Soc 2015;236(1112):v+99.
[6] Liao Yulei, Ming Pingbing. Deep Nitsche method: Deep Ritz method with essential boundary conditions. 2019, arXiv:1912.01309, [cs, math].
[7] Goodfellow Ian, Bengio Yoshua, Courville Aaron. Deep learning. The MIT Press; 2016.
[8] LeCun Yann, Bengio Yoshua, Hinton Geoffrey. Deep learning. Nature 2015;521(7553):436–44.
[9] Nitsche J. Über ein Variationsprinzip zur Lösung von Dirichlet-Problemen bei Verwendung von Teilräumen, die keinen Randbedingungen unterworfen sind. Abh Math Semin Univ Hambg 1971;36(1):9–15.
[10] Wang Jihong, Xu Zhi-Qin John, Zhang Jiwei, Zhang Yaoyu. Implicit bias with Ritz-Galerkin method in understanding deep learning for solving PDEs. 2020, arXiv:2002.07989, [cs, math].
[11] Sheng Hailong, Yang Chao. PFNN: A penalty-free neural network method for solving a class of second-order boundary-value problems on complex geometries. 2020, arXiv:2004.06490, [cs, math].
[12] Grohs Philipp, Herrmann Lukas. Deep neural network approximation for high-dimensional elliptic PDEs with boundary conditions. 2020, arXiv:2007.05384, [cs, math, stat].
[13] Ambrosio Luigi, Gigli Nicola, Savaré Giuseppe. Gradient flows in metric spaces and in the space of probability measures. Lectures in mathematics ETH Zürich, 2nd ed.. Basel: Birkhäuser Verlag; 2008, p. x+334.
[14] Jordan Richard, Kinderlehrer David, Otto Felix. The variational formulation of the Fokker–Planck equation. SIAM J Math Anal 1998;29(1):1–17.
[15] Tsiourvas Asterios. Solving high dimensional PDEs using Machine Learning [MSc Thesis], National Technical University of Athens; 2020, https://dspace.lib.ntua.gr/xmlui/handle/123456789/52792.
[16] Hwang Hyung Ju, Kim Cheolhyeong, Park Min Sue, Son Hwijae. The deep minimizing movement scheme. 2021.
[17] Makridakis Charalambos, Mitsoudis Dimitrios, Rosakis Phoebus. On atomistic-to-continuum couplings without ghost forces in three dimensions. Appl Math Res Express 2014;2014(1):87–113.
[18] Hochreiter Sepp, Schmidhuber Jürgen. Long Short-Term Memory. Neural Comput 1997;9(8):1735–80, _eprint: http://dx.doi.org/10.1162/neco.1997.9.8.1735.
[19] Pascanu Razvan, Mikolov Tomas, Bengio Yoshua. Understanding the exploding gradient problem. 2012, CoRR, abs/1211.5063.
[20] Glorot Xavier, Bengio Yoshua. Understanding the difficulty of training deep feedforward neural networks. In: Teh Yee Whye, Titterington Mike, editors. Proceedings of the thirteenth international conference on artificial intelligence and statistics. Proceedings of machine learning research, vol. 9, Chia Laguna Resort, Sardinia, Italy: PMLR; 2010, p. 249–56.
[21] Knight Philip A. The Sinkhorn–Knopp Algorithm: Convergence and applications. SIAM J Matrix Anal Appl 2008;30(1):261–75.
[22] Cuturi Marco. Sinkhorn distances: Lightspeed computation of optimal transport. In: Burges CJC, Bottou L, Welling M, Ghahramani Z, Weinberger KQ, editors. Advances in neural information processing systems. vol. 26, Curran Associates, Inc.; 2013.