mathematics

MDPI

Article

An Extrinsic Approach Based on Physics-Informed Neural Networks for PDEs on Surfaces

Zhuochao Tang [1,2], Zhuojia Fu [1,2,3,*] and Sergiy Reutskiy [2]

[1] Key Laboratory of Ministry of Education for Coastal Disaster and Protection, Hohai University, Nanjing 210098, China
[2] Center for Numerical Simulation Software in Engineering and Sciences, College of Mechanics and Materials, Hohai University, Nanjing 211100, China
[3] State Key Laboratory of Mechanics and Control of Mechanical Structures, Nanjing University of Aeronautics and Astronautics, Nanjing 210098, China
* Correspondence: paul212063@hhu.edu.cn

Abstract: In this paper, we propose an extrinsic approach based on physics-informed neural networks (PINNs) for solving the partial differential equations (PDEs) on surfaces embedded in high dimensional space. PINNs are one of the deep learning-based techniques. Based on the training data and physical models, PINNs introduce the standard feedforward neural networks (NNs) to approximate the solutions to the PDE systems. Using automatic differentiation, the PDEs information could be encoded into NNs and a loss function. To deal with the surface differential operators in the loss function, we combine the extrinsic approach with PINNs and then express that loss function in extrinsic form. Subsequently, the loss function could be minimized extrinsically with respect to the NN parameters. Numerical results demonstrate that the extrinsic approach based on PINNs for surface problems has good accuracy and higher efficiency compared with the embedding approach based on PINNs. In addition, the strong nonlinear mapping ability of NNs makes this approach robust in solving time-dependent nonlinear problems on more complex surfaces.

Keywords: machine learning; extrinsic; embedding; intrinsic; surfaces; Laplace–Beltrami operator

MSC: 68T07; 65N99; 65M99

Citation: Tang, Z.; Fu, Z.; Reutskiy, S. An Extrinsic Approach Based on Physics-Informed Neural Networks for PDEs on Surfaces. *Mathematics* **2022**, *10*, 2861. https://doi.org/10.3390/math10162861

Academic Editors: Fajie Wang, Ji Lin and Junseok Kim

Received: 4 July 2022
Accepted: 8 August 2022
Published: 11 August 2022

1. Introduction

Various applications in science and engineering, as a matter of fact, refer to solutions of Partial Differential Equations (PDEs) on curved surfaces or more general manifolds. Such applications include the generation of textures [1] or the visualization of vector fields [2] in image processing, flows and solidification [3] on surfaces in fluid dynamics and evolving surfactants [4] on interfaces in biology, etc.

To solve such surface problems, many numerical methods have been put into operation, including the typical finite difference method (FDM), finite element method (FEM), finite volume method (FVM), phase field (PF) method, radial basis function (RBF) collocation method, meshless generalized finite difference method (GFDM), generalized moving least squares (GMLS) method, etc. Generally, these methods cannot be directly used to handle surface problems because the surface differential operators are defined in tangent space rather than Euclidean space. In order to effectively map surface operators, Ruuth et al. [5] put forward the closest point method based on the closest point representation of the surface and then solved embedded PDEs by standard FDM in Euclidean space; further, Piret [6] presented the orthogonal gradients method, which extends the closest point method to a mesh-free version; Hansbo et al. [7] proposed the cut finite element method to solve PDEs on implicit surfaces via level set methods; Cheung et al. [8] combined the unsymmetric Kansa method and embedding conditions (or constant-along-normal conditions)

to construct an overdetermined system for such surface problems; Chen et al. [9,10] used the projection matrix and the idea of pseudospectra to approximate the Laplace–Beltrami operator (also known as surface Laplace operator) only using collocation points on surfaces. These advanced techniques to map surface operators can be roughly divided into three categories: intrinsic approaches [11], embedding approaches [12] and extrinsic approaches [13]. Intrinsic treatment aims to impose global or local parameterization [11] on curved surfaces and then express surface differential operators within new coordinates. The embedding approach aims to make embedding PDE be the analog of the surface PDE and involve only the standard Cartesian operators. The extrinsic approach is to express surface operators in extrinsic form and approximate them extrinsically. In our previous work [14–16], we have combined the meshless GFDM with an extrinsic approach to solve uring pattern formation problems, anomalous diffusion problems and the heat and mass transfer problems on surfaces. The extrinsic approach is numerically proved to be a quite effective treatment.

However, traditional numerical methods inevitably need mesh generation or node generation over the whole computational domain. Additionally, the quality of mesh or node distribution more or less has an influence on numerical accuracy [17]. On the contrary, there is no such concept as mesh quality in machine learning methods [18]. In other words, machine learning methods do not require high-quality meshes, but only require relatively uniform data sampling. To overcome the dilemma that conventional neural network methods lack robustness under the small data regime, Tarkhov et al. [19–22] first introduced the PDE information into neural network models with single hidden layers to solve various mathematical problems. Based on this, Raissi et al. [23,24] recently developed a series of deep neural networks based on physical information named physics-informed neural networks (PINNs). PINNs aim to replace the PDE solution with a feedforward neural network and take advantage of information from PDEs and initial/boundary conditions to form an optimized system explicitly. This explicit system originates from the information based on training data and could also be defined as a terminology loss function. By minimizing this system with respect to the parameters (including weights and biases) defined in NNs, PINNs could find one NN which best describes the physical model governed by the PDEs [25–27]. To specify the differential operators acting on the variables, PINNs employ the automatic differentiation technology and classical chain rule. As a matter of fact, Bihlo et al. [28] have applied PINNs to solve shallow-water equations on the sphere. In that paper, they used the latitude–longitude coordinates; i.e., they imposed one smooth parameterization on sphere and then expressed the shallow-water equations in latitude–longitude coordinates. Apparently, the same operation cannot be conducted on more general surfaces. Additionally, Fang et al. [29] first combined the PINNs with the embedding approach to solve time independent PDEs on surface. However, they only considered some of the embedding conditions, and the numerical accuracy can be further improved by applying complete embedding conditions.

In this paper, our main contribution is to propose an extrinsic approach based on the PINNs to solve surface PDEs on curved surfaces or more general manifolds. Compared with surface-type intrinsic approach, although our method is related to the ambient dimension rather than the surface dimension, its capability of handling more complex surfaces makes it more competitive. In addition, we also combined the embedding approach with PINNs to make a direct comparison with the extrinsic approach with regard to computational efficiency. We introduce the complete embedding conditions, which means that more complex optimization function will be formed, resulting in the inefficiency of the approach. This also shows that extrinsic approach performs well in computational efficiency.

The remainder of the paper is organized as follows: Section 2 gives details on PDEs defined on surfaces, introduces the PINNs and describes their implementation. In Section 3, we demonstrate the effectiveness of PINNs under several numerical examples. In this section, we first illustrate the convergence results by using different parameters in PINNs and test the robustness of PINNs by adopting sundry smooth surfaces. In the same section, we also present a comparison of numerical results by using randomly distributed training points and points that are quasi-uniformly distributed in 3D space. Further, we explore

the potential of PINNs for time-dependent nonlinear problems on more general surfaces. Finally, the conclusions and discussions are summarized in Section 4.

2. Methodology

In this section, a detailed description of surface differential operators involved in PDEs defined on surfaces, the implementation of physics-informed neural networks and their extrinsic treatment for solving surface PDEs are presented. In addition, a brief procedure of PINNs and its distinguishments from other methods are given.

2.1. Continuous Differential Operators on Surfaces and Its Extrinsic Form

The main difference between surface PDEs defined on surfaces and standard PDEs posed in some bounded domains with flat geometries is that the curvatures of surfaces play vital roles in physical models governed by the PDEs. We first pay attention to the differential operators posed on some sufficiently smooth, connected and compact surface $S \subset \mathbb{R}$ with no boundary and $dim(S) = d - 1$. The dimension $d = 3$ is taken into consideration for notational simplicity, and any other cases with higher d could be extended simply. To specify the relationship between surface differential operators and standard Euclidean differential operators, we denote the unit outward normal vector at any $\mathbf{x} \in S$ as $\mathbf{n} = (n^x, n^y, n^z)$ and the corresponding projection matrix to the tangent space as

$$\mathbf{P}(x) = (\mathbf{I}_3 - \mathbf{n}\mathbf{n}^T) \in \mathbb{R}^{3\times 3}, \tag{1}$$

where $\mathbf{I}$ is the 3-by-3 identity matrix. Then, the surface gradient operator ∇_S could be defined in terms of the standard Euclidean gradient ∇ via projections as

$$\nabla_S := \mathbf{P}\nabla, \tag{2}$$

and similarly, the Laplace–Beltrami operator (also known as surface Laplace operator) Δ_S could be defined as

$$\Delta_S := \nabla_S \cdot \nabla_S. \tag{3}$$

The Laplace–Beltrami operator could be regarded as a divergence-gradient operator. By introducing the extrinsic idea and substituting Equation (2) into Equation (3), the extrinsic (Euclidean) form [8] of the surface gradient operator and Laplace–Beltrami operator acting on any sufficiently smooth function could be derived as

$$\nabla_S u := \nabla u - \mathbf{n}\partial_{\mathbf{n}} u, \tag{4}$$

$$\Delta_S u := \Delta u - H_S \partial_{\mathbf{n}} u - \partial_{\mathbf{n}}^{(2)} u. \tag{5}$$

in which $\partial_{\mathbf{n}} u = \mathbf{n}^T \nabla u$, $\partial_{\mathbf{n}}^{(2)} u := \mathbf{n}^T J(\nabla u)\mathbf{n}$ and $H_S = \text{trace}\big(J(\mathbf{n})(\mathbf{I} - \mathbf{n}\mathbf{n}^T)\big)$. Here, J means the Jacobian operator in Euclidean space. Obviously, Euclidean space is the one we are most familiar with, and most algorithms are also developed in Euclidean space. Once the extrinsic (Euclidean) form is obtained, the approximations of surface operators are conducted naturally. It should be noted here that the Euclidean way is just one of the extrinsic treatments, and this way makes the approximation be implemented in the ambient dimension rather than the surface dimension.

For better understanding, we give one example to derive the explicit expression of surface differential operators on the unit sphere. Simplifying with the surface $S = x^2 + y^2 + z^2 - 1$, one could naturally obtain the unit normal vector $[x \quad y \quad z]^T$. Putting this into Equations (4) and (5), the extrinsic surface differential operators are represented by

$$\nabla_S = \begin{bmatrix} 1-x^2 & -xy & -xz \\ -xy & 1-y^2 & -yz \\ -xz & -yz & 1-z^2 \end{bmatrix} \begin{bmatrix} \partial_x \\ \partial_y \\ \partial_z \end{bmatrix} = \begin{bmatrix} (1-x^2)\partial_x - xy\partial_y - xz\partial_z \\ -xy\partial_x + (1-y^2)\partial_y - yz\partial_z \\ -xz\partial_x - yz\partial_y + (1-z^2)\partial_z \end{bmatrix}, \tag{6}$$

$$\Delta_S = (1-x^2)\partial_{xx} + (1-y^2)\partial_{yy} + (1-z^2)\partial_{zz} - 2xy\partial_{xy} - 2xz\partial_{xz} - 2yz\partial_{yz} - 2x\partial_x - 2y\partial_y - 2z\partial_z. \tag{7}$$

Once Equations (6) and (7) have been obtained, the approximation for surface operators defined on smooth surfaces could be expressed using some existing methods. For other surfaces, the normal information is different, hence the difference in Equations (6) and (7).

2.2. Physics-Informed Neural Networks (PINNs)

The main aim of PINNs is to approximate the solutions to PDEs. Like other numerical methods, the standard PINNs is derived in standard Euclidean space. In this section, we focus on introducing the basic idea of PINNs and how it solves PDEs on surfaces extrinsically. We use the steady state convective diffusion reaction equation

$$\Big(a\Delta_S - \vec{\mathbf{b}}\cdot\nabla_S + c\Big)u(x,y,z) = f(x,y,z) \tag{8}$$

with the certain coefficients $a, \vec{\mathbf{b}}, c$ as an illustration.

In the PINNs, there are three different ways to construct the approximate solutions $u(x,y,z)$ to the PDEs [26]. Due to fact that the PDE (8) defined on closed surfaces has no boundary conditions, the direct construction of the approximate solutions is employed in this work as an output of neural networks (NN), namely, $\tilde{u}(x,y,z) = u_{NN}(\mathbf{x};\boldsymbol{\mu})$, $\mathbf{x} \in S(\boldsymbol{\mu} = \{\mathbf{W},\mathbf{B}\})$. The NN, which is parameterized with finitely many weights $\mathbf{W}$ and biases $\mathbf{B}$, acts as a surrogate model of the PDE model to approximate the mapping from the spatial coordinates to the solutions of equation. One NN usually contains multiple hidden layers to obtain more accurate solutions. Here, PINNs seek to optimize the NN's parameters composed of weights and biases by minimizing the so-called loss function. Usually, the loss function is defined as the sum of mean squared error from both governing equations (PDEs) and boundary conditions on the training points. For PDEs defined on surfaces without boundary conditions, the loss function is expressed by the NN parameter $\boldsymbol{\mu}$ as

$$Loss(\boldsymbol{\mu}) = \frac{1}{N}\sum_{k=1}^{N}\Big[\Big(a\Delta_S - \vec{\mathbf{b}}\cdot\nabla_S + c\Big)\tilde{u}(\mathbf{x}_k) - f(\mathbf{x}_k)\Big]^2, \tag{9}$$

in which N is the total number of the training points. By substituting Equation (4) and Equation (5) into Equation (9), the loss function in extrinsic form finally could be derived under Cartesian coordinate by the NN parameter as

$$Loss(\boldsymbol{\mu}) = \frac{1}{N}\sum_{k=1}^{N}\Big[\Big(a(\Delta - H_S\partial_{\mathbf{n}} - \partial_{\mathbf{n}}^{(2)}) - \vec{\mathbf{b}}\cdot(\nabla - \mathbf{n}\partial_{\mathbf{n}}) + c\Big)u_{NN}(\mathbf{x}_k;\boldsymbol{\mu}) - f(\mathbf{x}_k)\Big]^2. \tag{10}$$

As mentioned in Section 1, the embedding approach based on PINNs is also discussed in this work for comparison with the extrinsic approach. As can be seen in Equations (4) and (5), the surface operators could be completely equal to the standard operator with the constraints $\partial_{\mathbf{n}}u = 0$ and $\partial_{\mathbf{n}}^{(2)}u = 0$. The constraints are embedding conditions. Therefore, the loss function in embedding form could be written as

$$\begin{aligned} Loss(\boldsymbol{\mu}) =& \frac{1}{N}\sum_{k=1}^{N}\Big[\Big(a\Delta - \vec{\mathbf{b}}\cdot\nabla + c\Big)u_{NN}(\mathbf{x}_k;\boldsymbol{\mu}) - f(\mathbf{x}_k)\Big]^2 \\ &+ \frac{1}{N}\sum_{k=1}^{N}\big[\partial_{\mathbf{n}}u_{NN}(\mathbf{x}_k;\boldsymbol{\mu})\big]^2 + \frac{1}{N}\sum_{k=1}^{N}\Big[\partial_{\mathbf{n}}^{(2)}u_{NN}(\mathbf{x}_k;\boldsymbol{\mu})\Big]^2. \end{aligned} \tag{11}$$

For PINNs, it is easy to add only two constraints to the optimization function as Equation (11). Although the extrinsic treatment needs many computations, as shown in Equations (4) and (5), they could be pre-computed for a certain surface before the "training", just like Equations (6) and (7) for a unit sphere. Then, the surface operators could be regarded as some specified operators defined in Euclidean space. Once they have been obtained, the loss function could be expressed explicitly only using governing equation

without any constraints. Compared with the embedding treatment having extra constraints, the loss function in extrinsic form is simpler in the "training" process.

Then, the original problem (8) becomes an optimization problem, namely,

$$\boldsymbol{\mu}^* = \arg\min_{\boldsymbol{\mu}} Loss(\boldsymbol{\mu}) \tag{12}$$

in which the $\boldsymbol{\mu}^*$ represents the optimal parameters.

Herein the automatic differentiation technique and the chain rule are used in loss function to compute the spatial derivatives of $u_{NN}(\mathbf{x};\boldsymbol{\mu})$. For time-dependent problems, the approximation could be regarded as $u_{NN}(\mathbf{x},t;\boldsymbol{\mu})$, and the temporal derivative could be realized in two ways: similar treatment as a spatial derivative and individual time integration using the method of lines. Then, different optimization algorithms can be used to solve Equation (12). This optimization process is called "training". Additionally, we use multiple sets of initial NN parameters $\boldsymbol{\mu}$ in the following numerical examples to avoid its uncertainty.

2.3. The Procedure of the Extrinsic Approach Based on PINNs

To better understand this extrinsic numerical framework for approximating the surface PDEs and compare it with traditional numerical methods, pseudocode is demonstrated in this section. We first give the steps of some methods, involving linear algebra such as FEM; RBF collocation methods; meshless GFDM; etc. In the implementation of these methods, the process is more or less divided into five steps briefly: firstly, generate the meshes/collocation points on surfaces; secondly, construct the approximate solutions based on respective approximation theory; thirdly, form the stiffness matrix or basis matrix for each mesh/point extrinsically; fourthly, assemble the information on each mesh/point and then obtain a discrete system with respect to the PDE model on surfaces; lastly, solve the algebraic system by using linear solver.

Differently, the pseudocode of the extrinsic approach based on PINNs could be summarized in Algorithm 1.

Algorithm 1 The extrinsic approach based on PINNs.

Require: The training datasets including a group of spatial coordinates and the corresponding solutions; the prescribed number of width and depth in NN; the initialized NN parameters; the convergence tolerance ε and number of iterations N_i;
Ensure: The surrogate NN model with optimized parameters;
1: Construct the NN with initialized parameters;
2: Specify the training sets for governing equation;
3: Specify the loss function in extrinsic form considering the governing equation;
4: **repeat**
5: $n \leftarrow n+1, n < N_i$;
6: Optimization: compute Equation (12);
7: Update the loss value;
8: **until** Loss value $< \varepsilon$
9: Determine the optimal parameters;
10: Substitute test datasets and then acquire the posterior error.

The concept of datasets in PINNs is somewhat similar to the that of collocation points [17]—namely, the PINNs are also meshless. It inherits the advantages of both meshless and neural network methods. In addition, although the numerical accuracy of PINNs in the present study on surface PDEs is usually not as high as those of some collocation methods such as RBF collocation methods, the PINNs is easy implement because neural networks can directly be used to deal with nonlinear problems without introducing iterative algorithms. These two advantages over traditional methods make PINNs quite attractive recently.

3. Numerical Examples

In this section, several different examples are provided. We first explore the convergence and the accuracy of PINNs for Equation (8) on the unit sphere, and then more surfaces and nonlinear PDEs are taken into consideration to verify its robustness. To quantify the accuracy and effectiveness of our approximate solutions, we introduce the L_2 error measures as follows.

$$L_2 = \sqrt{\sum_{k=1}^{N} (u(\mathbf{x}_k) - \tilde{u}(\mathbf{x}_k))^2} / \sqrt{\sum_{k=1}^{N} (u(\mathbf{x}_k))^2} \tag{13}$$

where $u(\mathbf{x}_k), \tilde{u}(\mathbf{x}_k)$ represent the reference solution and approximate solution at the k-th point. To avoid the uncertainty of different initializations for the network parameters $\boldsymbol{\mu}$ and find an optimal neural network as much as possible, we employed the L-BFGS optimization method and plot the mean for the solution errors from the 10 runs, which we adopted as a new metric of convergence. The Xavier initialization and hyperbolic tangent activation function were taken into consideration, and all the tests were implemented in Python on laptop with CPU i5-8265U @1.60 GHz and RAM 8.00 GB.

Example 1. *Convergence and accuracy test on a unit sphere*

In this example, we used Equation (8), and the coefficients were chosen as $a = 1$, $\vec{\mathbf{b}} = [1\ 1\ 1]^T, c = 5$, and the reference solution was assembled by trigonometric function, which is expressed as

$$u(x, y, z) = \sin x \sin y \sin z. \tag{14}$$

The force term was simply obtained by substituting the reference solution into the equation. A total number of 2500 points were chosen to be distributed on the unit sphere, as shown in Figure 1. Here, we selected N points randomly from these quasi-uniform points and the corresponding solutions from Equation (14) as training data, and all these 2500 points were regarded as test points to test the convergence of PINNs. As derived above in Equations (6) and (7), the loss function on this unit sphere could be obtained easily.

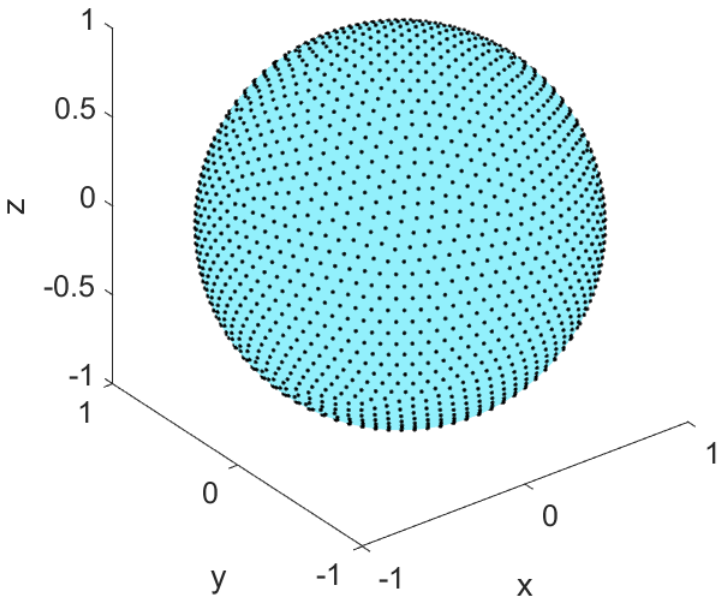
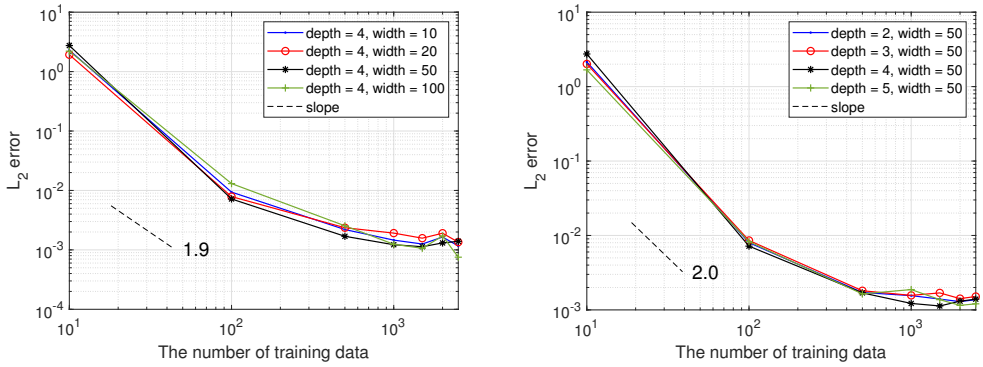


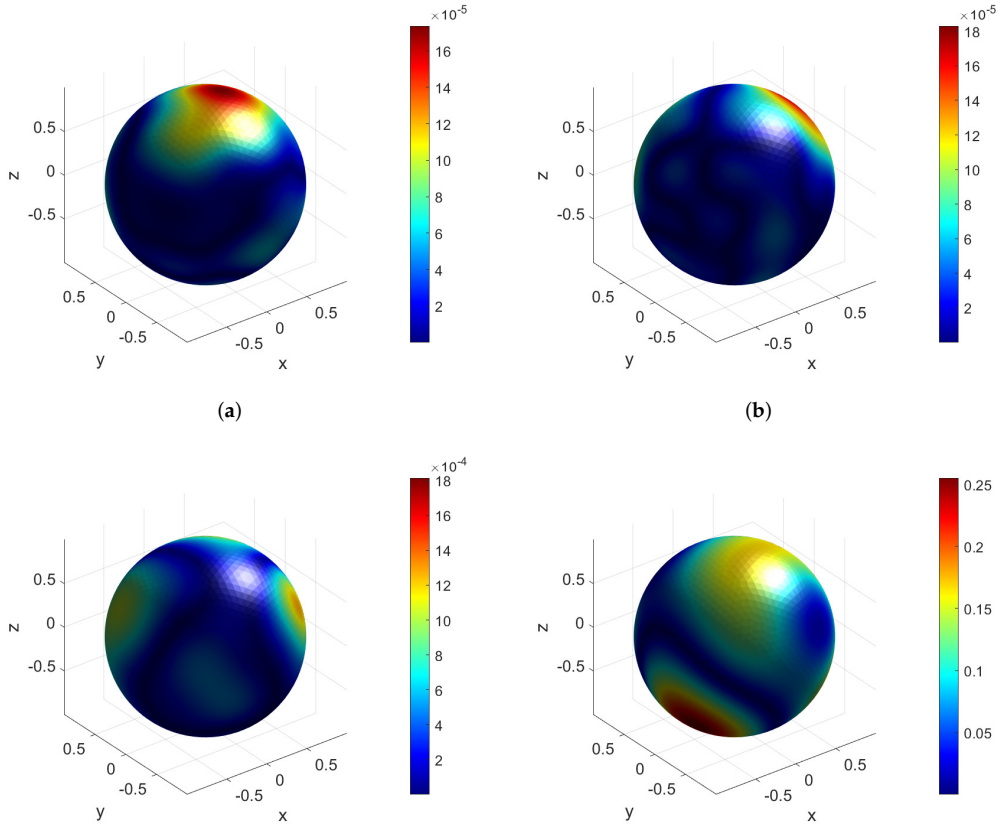
Figure 1. Sketch of the quasi-uniform points distributed on the unit sphere: the point sets could be obtained by using the minimum energy (ME) algorithm [30].

Since we had no idea of how sensitive PINNs approximations are to surface differentiation operators, we attempted to use various NNs with different numbers of hidden layers (also known as the depth of the NN; e.g., four hidden layers' mean depth is five) and neurons (also known as the width of the NN; e.g., 20 neurons' mean width is 20). Figure 2 shows the convergence results, and Figure 3 indicates some snapshots of error distribution by using different parameters. Tables 1 and 2 give some numerical results using smaller width and depth for solving linear problems on surfaces.



(**a**) (**b**)

Figure 2. Example 1: Convergence results by using (**a**) different widths and (**b**) different depths.



(**a**) (**b**)

(**c**) (**d**)

Figure 3. Example 1: Random few snapshots of absolute error distribution under width = 50 and depth = 4 by using (**a**) 2500 training data points; (**b**) 1500 training data points; (**c**) 100 training data points; (**d**) 10 training data points.

Table 1. Example 1: L_2 error and CPU time using different depths under width = 50 and 2000 training data.

Depth	**2**	**3**	**4**	**5**
L_2 error	1.12×10^{-3}	1.41×10^{-2}	1.34×10^{-3}	1.21×10^{-3}
CPU time	19.96 (s)	29.42 (s)	76.65 (s)	102.26 (s)

Table 2. Example 1: L_2 error and CPU time using different widths under depth = 4, with 2000 training data points.

Width	**3**	**5**	**10**	**20**	**50**	**100**
L_2 error	4.06×10^{-2}	1.02×10^{-2}	1.44×10^{-3}	1.91×10^{-3}	1.34×10^{-3}	1.69×10^{-3}
CPU time	5.47 (s)	8.54 (s)	17.67 (s)	34.63 (s)	76.65 (s)	152.72 (s)

As seen in Figures 2 and 3, we used, respectively, 2, 3, 4 and 5 hidden layers with 10, 20, 50 and 100 neurons to test convergence and accuracy of PINNs in solving Equation (8). The distribution of error could be affected by many factors, such as the width/depth of NNs, the initializations of NNs and the potential noise of training data. Numerical results converged at around $10^{-4} \sim 10^{-3}$ with convergence rates of 1.9 and 2.0. Apparently, we could obtain similar results by using different depths and widths when the number of training data reached 500 or more. In Table 1, we can see that for linear surface PDEs, a network with one hidden layer works fine, and it has the advantages of simplicity and speed of operation. In Table 2, we can see that when using 3 or 5, the numerical accuracy would be reduced to around 10^{-2}. To connect numerical results in Table 2 with those in Table 1, we further considered the case with depth = 2 and width = 3, and its L_2 error is 0.71. We could summarize that the depth, width and number of training data indeed influence the numerical results. Additionally, for surface linear problems, using smaller width and depth is more suitable due to its higher efficiency and for surface nonlinear problems, width and depth should be increased correspondingly. This shows PINN approximation has good adaptability to surface differential operators. Furthermore, we particularly plot the error distribution in Figure 3 to visualize the results, which shows good accuracy of PINNs for explicitly solving surface PDEs.

In addition, we further compare the extrinsic approach with the embedding approach both based on the PINNs. As mentioned in Equation (11), the embedding approach needs other constraints, and in Table 3, one can find that the accuracies of the two techniques show almost no difference, but the computational time varies a lot. This is because the additional constraints of embedding conditions make loss function (11) a more non-convex function. Numerical results prove that PINNs combined with the extrinsic technique is more efficient.

Table 3. Example 1: L_2 error and CPU time by using extrinsic and embedding approaches with different numbers of training data under width = 50 and depth = 4.

N	**1000**	**1500**	**2000**	**2500**
Extrinsic	1.02×10^{-3}	9.88×10^{-4}	1.49×10^{-3}	9.36×10^{-4}
	32.70 (s)	65.42 (s)	102.26 (s)	108.66 (s)
Embedding	3.51×10^{-3}	4.80×10^{-3}	2.17×10^{-3}	1.90×10^{-3}
	113.93 (s)	237.02 (s)	312.26 (s)	418.55 (s)

Example 2. *Results on more general surfaces*

In this example, we attempted to test the robustness of PINNs by solving PDEs on more general surfaces, and made a direct comparison by using quasi-uniform distributed training data and randomly distributed training data as shown in Figure 4. The parametric equations or implicit expressions of some surfaces used in this or the following example, including Torus, a constant distance product (CDP) surface, Bretzel2, Orthocircle, Red Blood Cell (RBC) and tooth surface, are provided as

(1) Tours:

$$S = \left(1 - \sqrt{x^2 + y^2}\right)^2 + z^2 - \frac{1}{9}; \tag{15}$$

(2) CDP:

$$\begin{aligned} S = &\sqrt{(x-1)^2 + y^2 + z^2}\sqrt{(x+1)^2 + y^2 + z^2}\sqrt{x^2 + (y-1)^2 + z^2} \\ &\sqrt{x^2 + (y+1)^2 + z^2} - 1.1; \end{aligned} \tag{16}$$

(3) Bretzel2:

$$S = \left(x^2(1-x^2) - y^2\right)^2 + \frac{1}{2}z^2 - \frac{1}{40}; \tag{17}$$

(4) Orthocircle:

$$\begin{aligned} S = &\left((x^2 + y^2 - 1)^2 + z^2\right)\left((y^2 + z^2 - 1)^2 + x^2\right) \\ &\left((z^2 + x^2 - 1)^2 + y^2\right) - 0.075^2\left(1 + 3(x^2 + y^2 + z^2)\right); \end{aligned} \tag{18}$$

(5) RBC:

$$S = \begin{cases} x = 1.15\cos(\lambda)\cos(\theta), \\ y = 1.15\sin(\lambda)\cos(\theta), \\ z = 0.5\sin(\lambda)\left(0.24 + 2.3\cos(\theta)^2 - 1.3\cos(\theta)^4\right), \end{cases} \quad -\pi \le \lambda \le \pi, \frac{-\pi}{2} \le \theta \le \frac{\pi}{2}. \tag{19}$$

(6) Tooth:

$$S = x^8 + y^8 + z^8 - (x^2 + y^2 + z^2); \tag{20}$$

In this test, the coefficients in Equation (8) were set as $a = 1, \vec{\mathbf{b}} = [1\ 1\ 1]^T, c = 1$, and the reference solution was changed to $u(x, y, z) = \sin x \cos y \sin z$. We first employed Torus by using 500 quasi-uniform training data and by using 500 randomly distributed training data to make a comparison.

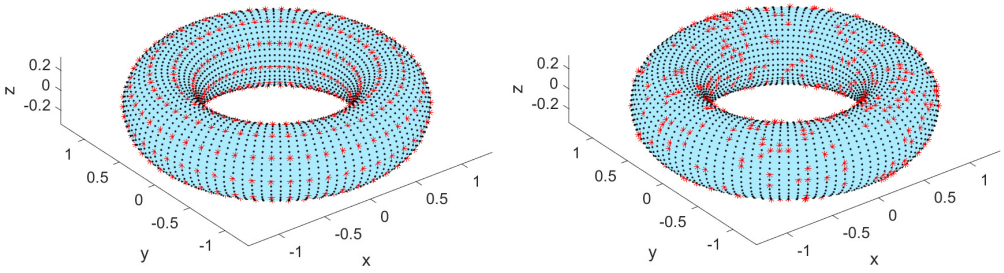
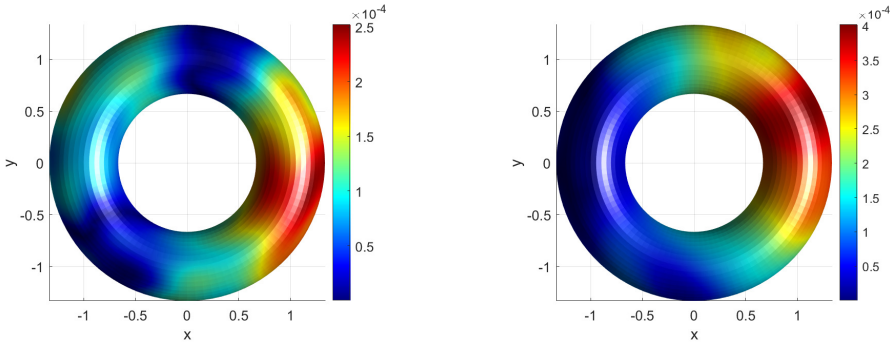


Figure 4. Example 2: Two different selections of training data on Torus: (**a**) quasi-uniform training data; (**b**) random training data generated by combined multiple recursive generator algorithm: red "*" points are selected training data; black points are test points.

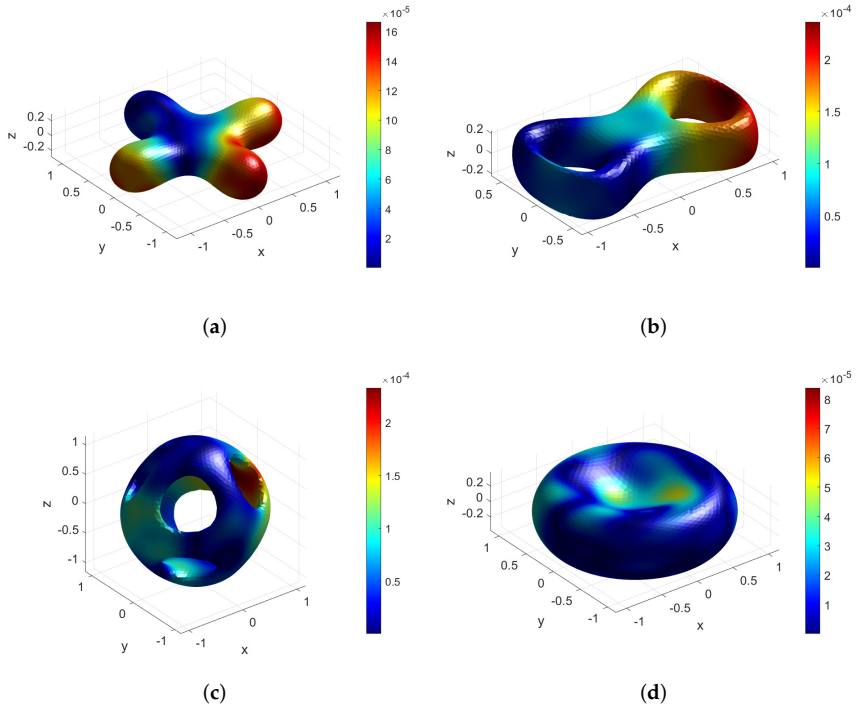
It can be found from Figure 5 that the distribution of training data slightly affected the numerical results. Although uniform sampling of the training dataset is always good for results, PINNs are superior to some typical numerical methods to some extent for solving PDEs on high dimensional surfaces because for PINNs combined with the extrinsic approach, only training data are required, rather than generating high quality meshes or regular points. Additionally, the distribution of training data influences the results little.



(**a**) (**b**)

Figure 5. Example 2: Snapshots of absolute error distribution under width = 50 and depth = 4: (**a**) by using quasi-uniform training data and (**b**) by using randomly distributed training data.

In addition, distribution numerical errors and L_2 errors on different surfaces are given, respectively, in Figure 6 and Table 4. The number of training points was chosen as 500, and the total numbers of points corresponding to CDP, Breztel2, Orthocircle and RBC were 3996, 3690, 4286 and 4000. When dealing with PDEs defined on high-dimensional complex surfaces, PINNs combined with the extrinsic approach show good stability and robustness.



(**a**) (**b**)

(**c**) (**d**)

Figure 6. Example 2: Snapshots of absolute error distribution under width = 50 and depth = 4 on various surfaces: (**a**) CDP, (**b**) Bretzel2, (**c**) Orthocircle, (**d**) RBC.

Table 4. Example 2: L_2 error on different surfaces under width = 50 and depth = 4.

Surfaces	CDP	Bretzel2	Orthocircle	RBC
L_2 error	1.18×10^{-3}	1.51×10^{-3}	4.20×10^{-3}	2.37×10^{-3}

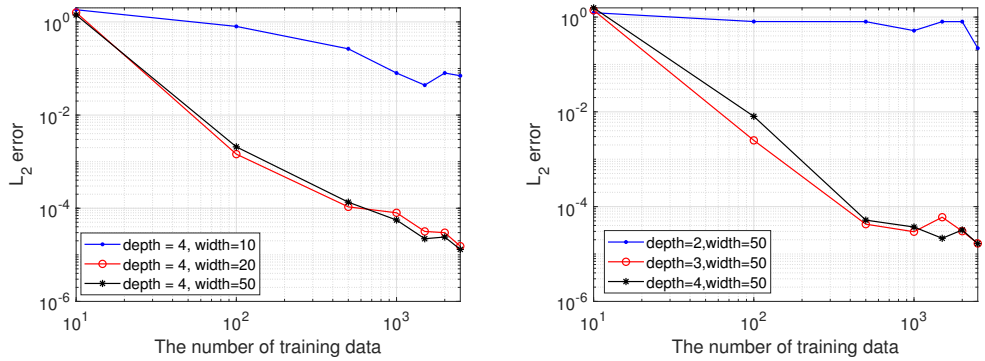
Example 3. *Nonlinear PDEs on surfaces*

In order to confront a more complicated model on different surfaces, the nonlinear model is considered in this example. The governing equation is

$$\left(a\Delta_S - \vec{\mathbf{b}} \cdot \nabla_S + c\right)u(x,y,z) + g(u) = f(x,y,z). \tag{21}$$

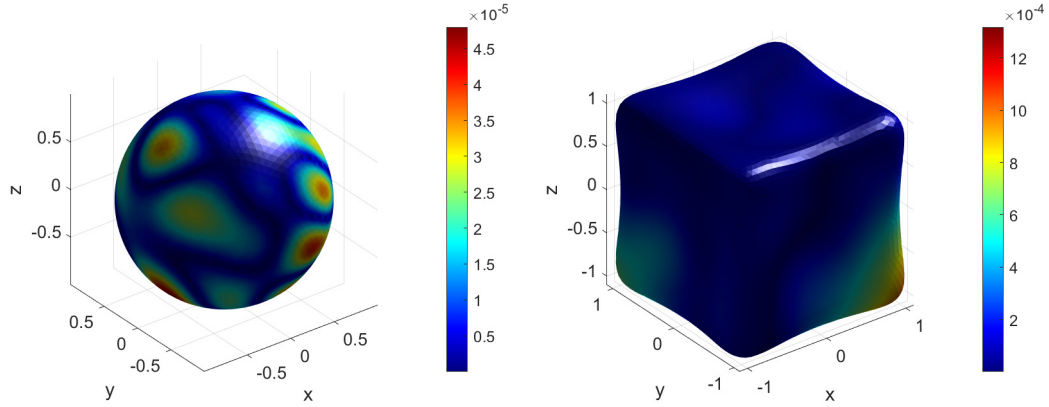
Herein $g(u) = u^2$ is the nonlinear term, the exact solution was set to $u = e^{x+y}\sin(z)$ and the parameters were $a = 1, \vec{\mathbf{b}} = \mathbf{0}, c = 0$. Similarly, the loss function in extrinsic form could be expressed as Equation (10).

We again performed the convergence analysis for this nonlinear model on a unit sphere, as exhibited in Figure 7. Apparently, compared with the results in Example 1, the numerical results of the nonlinear model are not accurate enough when the depth or width is too small. This means when the number of layers or the number of neurons is too small, the complex nonlinear behavior cannot be perfectly captured in spite of good nonlinear mapping capabilities of neural networks. As the width and depth increase, the numerical results show convergence similarly to the linear problems. We also plot the distribution of absolute error on the unit sphere and tooth surface under depth = 4 and width = 50, as shown in Figure 8, which indicates again that the PINNs combined with extrinsic approach perform well not only for linear problems but also for nonlinear problems on surfaces.



(**a**) (**b**)

Figure 7. Example 3: Convergence results by using (**a**) different widths and (**b**) different depths.



(**a**) (**b**)

Figure 8. Example 3: Snapshots of absolute error distribution under width = 50 and depth = 4 for nonlinear problems on (**a**) a unit sphere and (**b**) tooth surface.

Example 4. *Time-dependent nonlinear PDEs on surfaces*

In this example, a time-dependent nonlinear convective diffusion reaction equation on a unit sphere is considered as

$$\frac{\partial u}{\partial t} = \left(a\Delta_S - \vec{\mathbf{b}} \cdot \nabla_S + c\right)u(x,y,z,t) + g(u) + f \tag{22}$$

in which $g(u) = u^2$ and $a = 1, \vec{\mathbf{b}} = \mathbf{0}, c = 0$. The exact solution is given as $u = e^{x+y+z}\sin(t)$. Differently from the traditional methods combined with some time integration methods, the variable t in this example is considered as an individual variable, just like the spatial variable in the loss function, i.e.,

$$Loss(\boldsymbol{\mu}) = \frac{1}{N}\sum_{k=1}^{N}\left[\partial_t u_{NN}(\mathbf{x}_k, t_k; \boldsymbol{\mu}) - (a\Delta_S - \vec{\mathbf{b}} \cdot \nabla_S + c)u_{NN}(\mathbf{x}_k, t_k; \boldsymbol{\mu}) - g(\tilde{u}) - f(\mathbf{x}_k)\right]^2 \tag{23}$$

We plot the distribution of absolute error on the unit sphere at $t = 0.1$ as illustrated in Figure 9. The L_2 error is 1.65×10^{-3} using 2500 points with time increment $\Delta t = 0.01$. When considering the continuous time models, the original Equation (22) becomes a 4D problem. We found that PINNs has a good ability to approximate high-dimensional problems, which can be well combined with an extrinsic approach.

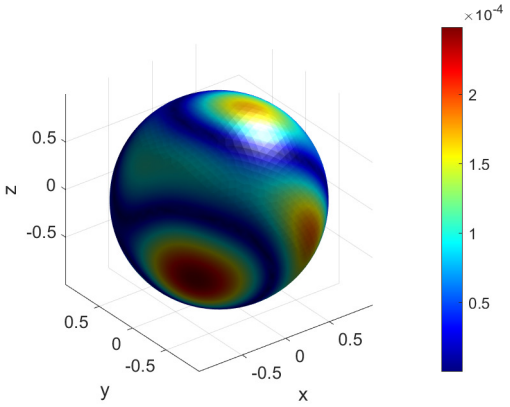


Figure 9. Example 4: Distribution of absolute error under width = 50 and depth = 4 for time-dependent nonlinear problems (22) on the unit sphere.

4. Conclusions and Discussions

In this work, the extrinsic approach based on PINNs is proposed and shows good performance and potential in the solutions of linear or nonlinear partial differential equations (PDEs) on surfaces embedded in high dimensional space. We could conclude from the first example that PINNs converge rapidly at the beginning of the increasing number of training points due to the dominant effect of the discretization error, and the solution will not be obviously improved with the further increase in the number of training points due to the dominant effect of optimization error. The second and third examples show that PINNs, as combinations of machine learning and differential equations, will not lose accuracy as the dimensionality (shape) increases in complexity; and will remain stable regardless of the distribution of training data or the complexity of the problem, as long as the data provided are accurate enough and the depth/width is large enough. This indicates the PINNs have good stability and robustness. In addition, we also compared the embedding approach based on PINNs with the extrinsic approach; the extrinsic approach based on PINNs showed better accuracy and used less computational time.

As a matter of fact, PDEs on curved surfaces or manifolds involve applications in biological pattern formation. In [31] and the references therein, it is proved that the geometry and specifically curvature play vital roles in biological pattern formation on curved surfaces. To deal with those surfaces composed of scatter points in realistic problems, two additional techniques, surface reconstruction [32,33] and the pseudospectral approach [9,16], could be further considered. Additionally, although the continuous time models are fine, they still face a dilemma when dealing with long simulations and large amounts of data, so there is a need

to introduce other techniques [34]. We revealed the potential of an extrinsic approach based on PINNs for surface problems in this work and leave the long simulations on complicated surfaces to our future work.

Author Contributions: Conceptualization, Z.F.; methodology, Z.F. and Z.T.; software, Z.T.; validation, Z.F., Z.T. and S.R.; writing—original draft preparation, Z.T.; writing—review and editing, Z.F. and S.R. All authors have read and agreed to the published version of the manuscript.

Funding: The work described in this paper was supported by the National Science Funds of China (grant number 12122205), Fundamental Research Funds for the Central Universities (grant number B220203018) and the Six Talent Peaks Project in Jiangsu Province of China (grant number 2019-KTHY-009).

Data Availability Statement: The data that support the findings of this study are available from the corresponding author (Z.F.) upon reasonable request.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Witkin, A.; Kass, M. Reaction-Diffusion Textures. *ACM Siggraph Comput. Graph.* **1995**, *25*, 299–308. [CrossRef]
2. Diewald, U.; Preusser, T. Anisotropic diffusion in vector field visualization on Euclidean domains and surfaces. *IEEE Trans. Vis. Comput. Graph* **2000**, *6*, 139–149. [CrossRef]
3. Myers, T.G.; Charpin, J.P.F. A mathematical model for atmospheric ice accretion and water flow on a cold surface. *Int. J. Heat Mass Transf.* **2004**, *47*, 5483–5500. [CrossRef]
4. Xu, J.J.; Zhao, H.K. An Eulerian Formulation for Solving Partial Differential Equations Along a Moving Interface. *J. Sci. Comput.* **2003**, *19*, 573–594. [CrossRef]
5. Ruuth, S.J.; Merriman, B. A simple embedding method for solving partial differential equations on surfaces. *J. Comput. Phys.* **2008**, *227*, 1943–1961. [CrossRef]
6. Piret, C. The orthogonal gradients method: A radial basis functions method for solving partial differential equations on arbitrary surfaces. *J. Comput. Phys.* **2012**, *231*, 4662–4675. [CrossRef]
7. Hansbo, P.; Larson, M.G.; Zahedi, S. A cut finite element method for coupled bulk-surface problems on time-dependent domains. *Comput. Methods Appl. Mech. Eng.* **2016**, *307*, 96–116. [CrossRef]
8. Cheung, K.C.; Ling, L. A Kernel-Based Embedding Method and Convergence Analysis for Surfaces PDEs. *SIAM J. Sci. Comput.* **2018**, *40*, A266–A287. [CrossRef]
9. Chen, M.; Ling, L. Kernel-based collocation methods for heat transport on evolving surfaces. *J. Comput. Phys.* **2019**, *405*, 109166. [CrossRef]
10. Chen, M.; Ling, L. Kernel-Based Meshless Collocation Methods for Solving Coupled Bulk–Surface Partial Differential Equations. *J. Sci. Comput.* **2019**, *81*, 375–391. [CrossRef]
11. Floater, M.S.; Hormann, K. *Surface Parameterization: A Tutorial and Survey*; Springer: Berlin/Heidelberg, Germany, 2005.
12. Macdonald, C.B.; Ruuth, S.J. The implicit closest point method for the numerical solution of partial differential equations on surfaces. *SIAM J. Sci. Comput.* **2010**, *31*, 4330–4350. [CrossRef]
13. Marcelo, B.; Li-Tien, C.; Stanley, O.; Guillermo, S. Variational Problems and Partial Differential Equations on Implicit Surfaces. *J. Comput. Phys.* **2001**, 174, 759–780.
14. Tang, Z.; Fu, Z.; Chen, M.; Ling, L. A localized extrinsic collocation method for Turing pattern formations on surfaces. *Appl. Math. Lett.* **2021**, *122*, 107534. [CrossRef]
15. Tang, Z.; Fu, Z.; Sun, H.; Liu, X. An efficient localized collocation solver for anomalous diffusion on surfaces. *Fract. Calc. Appl. Anal.* **2021**, *24*, 865–894. [CrossRef]
16. Tang, Z.; Fu, Z.; Chen, M.; Huang, J. An efficient collocation method for long-time simulation of heat and mass transport on evolving surfaces. *J. Comput. Phys.* **2022**, *463*, 111310. [CrossRef]
17. Fu, Z.; Tang, Z.; Xi, Q.; Liu, Q.; Gu, Y.; Wang, F. Localized Collocation Schemes and Their Applications. *Acta. Mech. Sin.* **2022**, *38*, 422167.
18. Lecun, Y.; Bengio, Y.; Hinton, G. Deep learning. *Nature* **2015**, *521*, 436. [CrossRef]
19. Tarkhov, D.A.; Vasilyev, A.N. New Neural Network Technique to the Numerical Solution of Mathematical Physics Problems. I: Simple Problems. *Opt. Mem. Neural Netw.* **2005**, *14*, 59–72.
20. Tarkhov, D.A.; Vasilyev, A.N. New Neural Network Technique to the Numerical Solution of Mathematical Physics Problems II: Complicated and Nonstandard Problems. *Opt. Mem. Neural Netw.* **2005**, *14*, 97–122.
21. Tarkhov, D.; Vasilyev, A.N. *Semi-Empirical Neural Network Modeling and Digital Twins Development*; Academic Press: Cambridge, MA, USA, 2019.
22. Antonov, V.; Tarkhov, D.; Vasilyev, A. Unified approach to constructing the neural network models of real objects Part 1. *Math. Methods Appl. Sci.* **2018**, *41*, 9244–9251. [CrossRef]

23. Raissi, M.; Perdikaris, P.; Karniadakis, G.E. Physics-Informed Neural Networks: A Deep Learning Framework for Solving Forward and Inverse Problems Involving Nonlinear Partial Differential Equations. *J. Comput. Phys.* **2018**, *378*, 686–707. [CrossRef]
24. Raissi, M.; Yazdani, A.; Karniadakis, G.E. Hidden fluid mechanics: Learning velocity and pressure fields from flow visualizations. *Science* **2020**, *367*, 1026–1030. [CrossRef]
25. Jagtap, A.D.; Kharazmi, E.; Karniadakis, G.E. Conservative physics-informed neural networks on discrete domains for conservation laws: Applications to forward and inverse problems. *Comput. Methods Appl. Mech. Eng.* **2020**, *365*, 113028. [CrossRef]
26. Pang, G.; Lu, L.; Karniadakis, G.E. fPINNs: Fractional Physics-Informed Neural Networks. *SIAM J. Sci. Comput.* **2019**, *41*, A2603–A2626. [CrossRef]
27. Mao, Z.; Jagtap, A.D.; Karniadakis, G.E. Physics-informed neural networks for high-speed flows. *Comput. Methods Appl. Mech. Eng.* **2020**, *360*, 112789. [CrossRef]
28. Bihlo, A.; Popovych, R.O. Physics-informed neural networks for the shallow-water equations on the sphere. *J. Comput. Phys.* **2022**, *456*, 111024. [CrossRef]
29. Fang, Z.; Zhan, J. A physics-informed neural network framework for PDEs on 3D surfaces: Time independent problems. *IEEE Access* **2019**, *8*, 26328–26335. [CrossRef]
30. Hesse, K.; Sloan, I.H.; Womersley, R.S. Numerical integration on the sphere. In *Handbook of Geomathematics*; Springer: Berlin/Heidelberg, Germany, 2010.
31. Krause, A.L.; Ellis, M.A.; Van Gorder, R.A. Influence of curvature, growth, and anisotropy on the evolution of Turing patterns on growing manifolds. *Bull. Math. Biol.* **2019**, *81*, 759–799. [CrossRef]
32. Zhao, H.K.; Osher, S.; Fedkiw, R. Fast surface reconstruction using the level set method. In Proceedings of the IEEE Workshop on Variational and Level Set Methods in Computer Vision, Vancouver, BC, Canada, 13 July 2001; pp. 194–201.
33. Liu, S.; Wang, C.C. Quasi-interpolation for surface reconstruction from scattered data with radial basis function. *Comput. Aided Geom. Des.* **2012**, *29*, 435–447. [CrossRef]
34. Gorbachenko, V.I.; Lazovskaya, T.V.; Tarkhov, D.A.; Vasilyev, A.N.; Zhukov, M.V. Neural network technique in some inverse problems of mathematical physics. In Proceedings of the International Symposium on Neural Networks, St. Petersburg, Russia, 6–8 July 2016; Springer: Berlin/Heidelberg, Germany, 2016; pp. 310–316.