Computers and Mathematics with Applications 121 (2022) 62–73

Contents lists available at ScienceDirect

Computers and Mathematics with Applications

www.elsevier.com/locate/camwa

HomPINNs: Homotopy physics-informed neural networks for learning multiple solutions of nonlinear elliptic differential equations

Yao Huang [a,d], Wenrui Hao [b], Guang Lin [c,*]

[a] School of Mathematics and Statistics, Xi'an Jiaotong University, Xi'an, Shaanxi 710049, China
[b] Department of Mathematics, Pennsylvania State University, University Park, PA 16802, USA
[c] Department of Mathematics, and School of Mechanical Engineering, Purdue University, West Lafayette, IN 47907, USA
[d] Department of Electrical and Computer Engineering, National University of Singapore, Singapore, 117583, Singapore

ARTICLE INFO

Keywords:
Machine learning
Physics-informed neural networks
Nonlinear elliptic differential equation
Multiple solutions
Homotopy method
Data-driven computation

ABSTRACT

Physics-informed neural networks (PINNs) based machine learning is an emerging framework for solving nonlinear differential equations. However, due to the implicit regularity of neural network structure, PINNs can only find the flattest solution in most cases by minimizing the loss functions. In this paper, we combine PINNs with the homotopy continuation method, a classical numerical method to compute isolated roots of polynomial systems, and propose a new deep learning framework, named homotopy physics-informed neural networks (HomPINNs), for solving multiple solutions of nonlinear elliptic differential equations. The implementation of an HomPINN is a homotopy process that is composed of the training of a fully connected neural network, named the starting neural network, and training processes of several PINNs with different tracking parameters. The starting neural network is to approximate a starting function constructed by the trivial solutions, while other PINNs are to minimize the loss functions defined by boundary condition and homotopy functions, varying with different tracking parameters. These training processes are regraded as different steps of a homotopy process, and a PINN is initialized by the well-trained neural network of the previous step, while the first starting neural network is initialized using the default initialization method. Several numerical examples are presented to show the efficiency of our proposed HomPINNs, including reaction-diffusion equations with a heart-shaped domain.

1. Introduction

Nonlinear differential equations are used, in a wide variety of disciplines, from biology [1,2], economics [3,4], physics [5,6], chemistry [7–9], and engineering [10], to quantitatively model the important features of real-world phenomena, such as equilibrium, conservation, motion, diffusion, reaction, pattern. Since most nonlinear differential equations do not have explicit solutions, solving nonlinear differential equations has become a particularly challenging task, especially when interested in multiple solutions which widely exist in combustion theory, astrophysics, differential geometry, general relativity, mathematical biology, general relativity, meteorology, shape optimization, and optimal transport [11–15]. Although that the solution existence and uniqueness can be proved for some time-dependent systems, most steady-state systems bring multiple solutions due to the non-linearity. There are many numerical approaches for computing the multiple solutions of nonlinear differential equations: **Time marching approach** is to introduce the time evolution to the nonlinear differential equation and solve the parabolic system for any given initial condition until the solution does not change much. However, this approach cannot filter out the metastable solutions. Moreover, their performance highly depends on the initial conditions in [16]. **A variational approach** has been developed to study the variational problems [17–19], for instance, the energy functional is solved via characterizing a typically minimax type critical point by the Ljusternik-Schnirelman principle [20,21,19]. However, due to the high nonlinearity in infinite-dimensional spaces, the complexity of obtaining an initial guess sufficiently close to an unknown saddle is commonly viewed as almost the same as that of finding the solution. This approach is for variational problems only and is hard to apply a general system that has no variational form [19]. **Newton's method** has also been used to compute the multiple solutions but depends on an initial guess, which severely reduces its effectiveness to find multiple solutions [22–24]. Some initial guesses have to be constructed based on the understanding of the nonlinear PDE [16], thus

* Corresponding author.
E-mail addresses: mathuangy@gmail.com (Y. Huang), wxh64@psu.edu (W. Hao), guanglin@purdue.edu (G. Lin).

https://doi.org/10.1016/j.camwa.2022.07.002
Received 31 December 2021; Received in revised form 6 May 2022; Accepted 8 July 2022

obtaining an initial guess sufficiently close to an unknown solution is a very challenging job. Recently, a deflation technique that was introduced to solve nonlinear equations [25] has been applied to nonlinear differential equations for computing multiple solutions and bifurcations [26,27]. However, this method introduces artificial singularities and may cause numerical instabilities, and cannot converge to another solution if they are far away from each other [25].

All the above traditional methods highly depend on the initial guess which is often hard to choose if the system is not fully understood. In order to solve this challenge, the homotopy continuation method has been developed for solving systems of nonlinear differential equation. As a classical numerical method, the homotopy continuation method is first applied to compute the isolated roots of polynomial systems [28–30], and then extended to solve multiple solutions of nonlinear differential equations by discretizing the differential equations with proper numerical methods into polynomial systems [31,32]. The basic idea of the homotopy continuation method is to track the solutions of the target system from the simple starting system with known solutions. In addition, the so-called bootstrapping method [33] combined the homotopy continuation method and domain decomposition techniques to compute multiple solutions of nonlinear differential equations.

Neural network-based (NN-based) techniques have also been used to solve differential equations from a few decades ago [34–36]. As a combination of a series of linear functions and nonlinear activation functions, the deep neural network (DNN) has universal approximation property and can approximate any nonlinear functions [37–40].

Most recently, a new framework, so-called physics-informed neural networks (PINNs), is presented in [41] to learn the solutions of differential equations guided by governing equations and initial/boundary data thus has greatly improved the capability of neural networks in solving differential equations [34,35]. Many variants of PINNs have been proposed to solve fractional partial differential equations [42], stochastic differential equations [43], the Euler equations with high-speed aerodynamic flows [44], Boltzmann equation with the Bhatnagar-Gross-Krook collision model [45], and so on, which have shown the effectiveness of PINNs for solving a different kind of differential equations. PINNs also have many other applications, such as material identification [46], water waves inversion with multi-fidelity data [47]. Besides, Parallel PINNs and XPINNs frameworks are proposed in [48] and [49] based on domain decomposition, and they can also be used for conservation laws [50,51]. Recently, Error estimates for PINNs approximating the Navier-Stokes equations are presented in [52]. In addition, there are also some other NN-based approaches to approximate differential equations based on Galerkin methods [53], the Ritz method [54], just to mention a few. However, due to the implicit regularity of neural network structure, all of those NN-based techniques usually can learn the flattest solution, i.e., low-frequency solution, of differential equations. Thus, it is very challenging to learning multiple solutions of nonlinear differential equations with DNNs.

In this work, we propose a new neural network framework to compute multiple solutions of nonlinear differential equations by combining PINNs with the homotopy continuation method, i.e., homotopy physics-informed neural networks (HomPINNs). The proposed framework is designed firstly to approximate the starting function with a full-connected neural network, whose loss function is defined as mean square error between the output of neural network and the corresponding data of starting functions. Then, a homotopy function is defined to track the solutions of the target differential equation from the starting functions with tracking the homotopy parameter t. The homotopy process can be divided into several steps by discretizing the tracking parameter t. After training the starting neural network with the starting function, the other steps are trained as PINNs initialized with neural network parameters of the previous well-trained neural network. The loss functions are constructed by homotopy functions with corresponding tracking parameters and initial conditions. When the parameter t satisfies $t = 1$, the corresponding optimized neural network with a loss under preset threshold is a solution of the nonlinear differential equation. Different from the traditional homotopy continuation method which needs to discretize differential equations into polynomial systems before the homotopy process, the NN-based homotopy process directly handles the homotopy function in differential equation form with the aide of automatic differentiation technique [55] involved in the backward propagation. Because the starting function is endowed with randomness, different solutions can be obtained by repeating the homotopy process with different starting functions.

The proposed framework in this paper is general for computing multiple solutions of different kinds of nonlinear differential equations. To illustrate the idea of this framework, we consider the following nonlinear elliptic differential equations:

$$\begin{cases} \mathcal{L}u(\mathbf{x}) = f(u, \mathbf{x}), & \mathbf{x} \in \Omega \\ \mathcal{B}u(\mathbf{x}) = b(\mathbf{x}), & \mathbf{x} \in \partial\Omega \end{cases} \tag{1}$$

where $\mathcal{L}$ is a linear differential operator, for example Laplace operator Δ, $f(u, \mathbf{x})$ is a nonlinear forcing term depending on both u and $\mathbf{x}$, $\mathcal{B}u(\mathbf{x}) = b(\mathbf{x})$ stands for the general boundary condition, and Ω is a bounded domain in $\mathbb{R}^d$. There are several advantages of our proposed HomPINNs listed below:

1) HomPINNs can learn multiple solutions of nonlinear differential equation, while in most case, PINNs can only find the flattest solution;
2) Compared with existing traditional numerical methods, our proposed HomPINNs are more efficient, e.g. uncover new solutions for the Henon equation in example 4.3;
3) HomPINNs are capable for solving nonlinear differential equation with arbitrary domains since the neural network discretization is a mesh free method, e.g., the heart-shaped domain in example 4.4.
4) HomPINNs are robust on different neural network structures for learning multiple solutions of the nonlinear differential equation, e.g., Table 1, Table 3 and Table 4 in section 4.

The remainder of this paper is organized as follows. The bases of our proposed HomPINNs, i.e., physics-informed neural networks and homotopy continuation method, are firstly introduced in Section 2. In Section 3, the details about the construction of HomPINNs are presented with schematic and algorithm. In addition, the design of starting function with hat function or Gaussian function is also shown in this section. In section 4, we provide several numerical examples to show the efficiency of our proposed HomPINNs, including reaction-diffusion equations with the heart-shaped domain. Finally, the conclusion is presented in Section 5.

2. Background

In this section, we introduce the PINNs which is designed to solve supervised learning tasks with respect to the given differential equations, and the homotopy continuation method which is a typical technique to obtain multiple solutions of different kind of differential equations or polynomial systems.

2.1. Physics-informed neural networks

The physics-informed neural networks leverage the universal function approximation property of deep neural networks to represent the solutions of general linear and nonlinear partial differential equations, and its schematic is shown in Fig. 1. Compared with standard deep learning approaches, this framework builds the physical laws (governing equations and boundary conditions) into the loss function utilizing automatic differentiation techniques [55] which is available in many machine learning libraries, such as PyTorch [56] and TensorFlow [57]. For the nonlinear PDE (1) that we consider in this paper, the solution

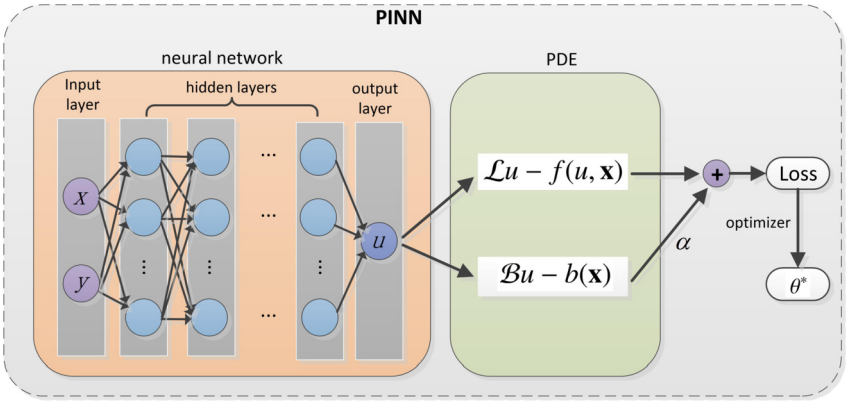


Fig. 1. Schematic of physics-informed neural networks (PINNs): The orange part is a fully connected neural network $u_N(\mathbf{x};\theta)$ ($\mathbf{x} = (x, y)$) which serves as a surrogate model of the PDE solution; the green part is informed physics described as governing equation and boundary condition which is used to define loss function by automatic differentiation [55].

can be approximated by a fully connected feed-forward neural network $u_N(\mathbf{x};\theta)$ in which the neural network parameters are optimized such that the resulting approximate solution will satisfy the governing equation and the boundary conditions, simultaneously.

Generally, there are two ways to approximate the solution $u(\mathbf{x};\theta)$. The first one is to build both the nonlinear differential equation and the boundary condition into the loss function together, namely,

$$L(\theta) = \frac{1}{N_g}\sum_{i=1}^{N_g}\left(\mathcal{L}u_N(\mathbf{x}_i^g) - f(u_N, \mathbf{x}_i^g)\right)^2 + \alpha\frac{1}{N_b}\sum_{j=1}^{N_b}\left(\mathcal{B}u_N(\mathbf{x}_j^b) - b(\mathbf{x}_j^b)\right)^2, \quad (2)$$

where $u_N(\mathbf{x};\theta)$ is an approximation of $u(\mathbf{x})$, $\{\mathbf{x}_i^g\}_{i=1}^{N_g}$, and $\{\mathbf{x}_j^b\}_{j=1}^{N_b}$ are collocation points of the governing equation and the boundary condition, respectively, and α is the balance parameter between the governing equation and the boundary condition. This is the most primitive and universal idea of PINNs presented in [41] as shown in Fig. 1.

The second one is to restrict the boundary condition on the neural network by augmenting the generic neural network $u_N(\mathbf{x};\theta)$ with several special designed functions to obtain the final neural network $\tilde{u}_N(\mathbf{x};\theta)$ which satisfies the boundary condition $\mathcal{B}\tilde{u}_N(\mathbf{x};\theta)|_{\Omega} = \mathbf{b}(\mathbf{x})$, automatically [42]. For example, if the boundary condition is $u(0) = u(1) = 0$ on $[0, 1]$, then the final neural network is defined as $\tilde{u}_N(x;\theta) = x(x-1)u_N(x;\theta)$. Then the loss function is defined based on the governing equation only, namely,

$$L(\theta) = \frac{1}{N_g}\sum_{i=1}^{N_g}\left(\mathcal{L}\tilde{u}_N(\mathbf{x}_i^g) - f(\tilde{u}_N, \mathbf{x}_i^g)\right)^2. \quad (3)$$

2.2. Homotopy continuation method

Denoting the numerical solution as $\mathbf{U}$, we have the following discretized polynomial system

$$\mathbf{F}_h(\mathbf{U}) \equiv \mathcal{L}_h\mathbf{U} - f_h(\mathbf{U}) = 0, \quad (4)$$

where $\mathcal{L}_h$ and f_h are the discretized operators of $\mathcal{L}$ and $f(\cdot, x)$ by finite difference method [58], finite element method [59], or spectral method [60]. If $f(u)$ is a polynomial function of u, then $\mathbf{F}_h$ is a polynomial system of $\mathbf{U}$. In order to compute all solutions of the elliptic equation, we define the following homotopy equation

$$\mathbf{H}(\mathbf{U}, t) := t\mathbf{F}_h(\mathbf{U}) + \gamma(1-t)\mathbf{G}_h(\mathbf{U}) = 0, \quad (5)$$

where $\mathbf{G}_h(\mathbf{U}) = 0$ is called the "starting system" [61], $t \in [0, 1]$ is called the homotopy parameter, and γ is a random complex number [61,62]. When $t = 0$, Eq. (5) is the starting system $\mathbf{G}_h(\mathbf{U}) = 0$. When $t = 1$, Eq. (5) recovers the target system (4), which is then solved by tracking solutions with respect to the homotopy parameter t from 0 to 1.

The starting system is chosen via closely mirroring the structure of $\mathbf{F}_h(\mathbf{U})$, such as the total degree start system ($\mathbf{G}_h(\mathbf{U})$ has the same degree as $\mathbf{F}_h(\mathbf{U})$), the multi-homogeneous start system by dividing the variables $\mathbf{U}$ into several homogeneous groups, the linear product start systems by dividing into several linear systems, and etc. Since both $\mathbf{G}_h(\mathbf{U})$ and $\mathbf{F}_h(\mathbf{U})$) are polynomials, all solutions of $\mathbf{F}_h(\mathbf{U})$ can be theoretically guaranteed by the homotopy continuation due to Bertini's theorem [61,63,64]. Roughly speaking, if two polynomials have no common solutions, a general linear combination of them, namely, the homotopy function, will define a smooth hypersurface. One choice of the starting system is so-called *total degree system*, namely,

$$\left(\mathbf{G}_h(\mathbf{U})\right)_k = (U_k)^{d_k} - 1, \quad k = 1, 2, \ldots, n, \quad (6)$$

where $\mathbf{U} = [U_1, \cdots, U_n]^T$ and d_k is the degree of the k-th equation of $\mathbf{F}_h(\mathbf{U})$. The total degree system is guaranteed by Bézout's Theorem [65] to compute all the solutions of $\mathbf{F}_h(\mathbf{U})$. Comparing to traditional methods that compute one single solution only, such as Newton-like methods and gradient methods, the disadvantage of using (6) is time consuming computations introduced by tracking $\Pi_{k=1}^n d_k$ solution paths when carrying out homotopy continuation, which could be very expensive for solving nonlinear PDEs on the fine grid. In order to speed up the computation, we combine the PINNs with the homotopy method.

3. Homotopy physics-informed neural networks (HomPINNs) for computing multiple solutions

In this section, we will construct neural networks leveraging the homotopy method to compute multiple solutions of the nonlinear PDE (1).

3.1. The architecture of homotopy physics-informed neural networks (HomPINNs)

Let $F(u) = \mathcal{L}u - f(u, \mathbf{x})$ and $G(u) = u^k - u_s^k$, where k is the degree of function f about u and u_s are the given simple starting functions, for example, piecewise linear function for one dimensional situation. We can apply homotopy continuation method to obtain multiple solutions of nonlinear PDE (1). The homotopy function is constructed as

$$H(u, t) = tF(u) + (1-t)G(u), \quad (7)$$

where $t \in [0, 1]$ is homotopy parameter.

By denoting the HomPINNs as $u_N(\mathbf{x};\theta,t)$, the loss function of the HomPINNs is defined as

$$L(\theta, t) = \frac{1}{N_g}\sum_{i=1}^{N_g} H(u_N(\mathbf{x}_i^g;\theta,t),t)^2 + \alpha\frac{1}{N_b}\sum_{j=1}^{N_b}\left(\mathcal{B}u_N(\mathbf{x}_j^b;\theta,t) - b(\mathbf{x}_j^b)\right)^2, \quad (8)$$

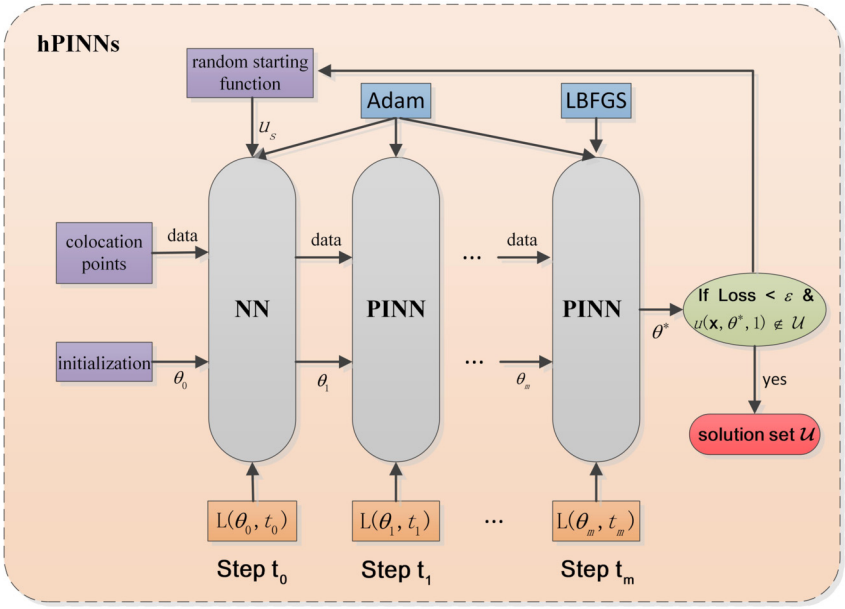


Fig. 2. Schematic of homotopy physics-informed neural networks (HomPINNs): the left neural network NN is a general fully-connected neural network to approximate the starting function u_s; other neural networks are PINNs initialized with parameters of previous well-trained neural network; the homotopy process is from step t_0 to step t_m, and if the final neural network with parameter θ^* satisfies the criteria in the green oval box, then it is a new learned solution of the target system to be added to solution set $\mathcal{U}$.

where $t \in (0,1]$, $\{\mathbf{x}_i^g\}_{i=1}^{N_g}$ and $\{\mathbf{x}_j^b\}_{j=1}^{N_b}$ are collocation points of homotopy equation and boundary condition, respectively. In particular, if $t = t_0 = 0$, then the loss function is defined as

$$L(\theta, 0) = \frac{1}{N_g} \sum_{i=1}^{N_g} \left(u_N(\mathbf{x}_i^g; \theta, 0) - u_s(\mathbf{x}_i^g) \right)^2. \tag{9}$$

Thus the starting neural network of HomPINNs is trained by minimizing the loss function (9) for different starting functions $\{u_{s,i}\}_{i=1}^{m_s}$. Then we track the solution path $u_N(\mathbf{x}; \theta, t)$ by optimizing the loss function (8) with respect to t and recover (1) at $t = 1$. Then the HomPINNs are trained for $t = t_1, t_2, \cdots, t_m$ where $0 < t_1 < t_2 < \cdots < t_m = 1$. In Step t_i, the network parameters θ are updated by

$$\theta \leftarrow \theta - \tau_n \nabla_\theta L(\theta, t_i),$$

where τ_n is the learning rate in the nth iteration, $n = 1, 2, \cdots, N_{I,i}$ and $i = 0, 1, \cdots, m$. The schematic of the HomPINNs is illustrated in Fig. 2 and the training process of HomPINNs is summarized in Algorithm 1.

Algorithm 1 The training process of HomPINNs.

Input: collocation points $\{\mathbf{x}_i^h\}_{i=1}^{N_h}$, $\{\mathbf{x}_i^b\}_{i=1}^{N_b}$, homotopy step $\{t_i\}_{i=0}^{m}$, starting functions $\{u_{s,i}\}_{i=1}^{m_s}$, the iteration number of every step $\{N_{I,i}\}_{i=0}^{m}$, the tolerance ε of finial loss
Output: the learned solution set $\mathcal{U}$
let $u_s = u_{s,1}$, $p = 0$ and $\mathcal{U} = \{\}$ (empty set)
While $p < m_s$ **do**
  for $k = 1, 2, \cdots, N_{I,0}$ **do**
    optimize the loss function $L(\theta, 0)$ (9) with *Adam*
  end
  for $i = 1, 2, \cdots, m$ **do**
    for $j = 1, 2, \cdots, N_{I,i}$ **do**
      optimize the loss function $L(\theta, t_i)$ (8) with *Adam*
    end
  end
  optimize $L(\theta, 1)$ with optimizer *LBFGS* to obtain $u_N(\mathbf{x}; \theta^*, 1)$
  If $L(\theta^*, 1) < \varepsilon$ & $u_N(\mathbf{x}; \theta^*, 1) \notin \mathcal{U}$ **do**
    put $u_N(\mathbf{x}; \theta^*, 1)$ into solution set $\mathcal{U}$
  end
  $p = p + 1$
end

3.2. Starting functions

Since $\mathcal{L}$ in Eq. (7) is a linear operator and does not contribute to the non-linearity, we may choose $G(u) = f(u)$. Thus the homotopy function becomes

$$H(u, t) \equiv t\mathcal{L}u + f(u) = 0. \tag{10}$$

In this case, we construct a starting function $u_s(\mathbf{x})$ with $f(u_s(\mathbf{x})) = \mathbf{0}$ for any given coarse point $\mathbf{x}$. In numerical computation, we construct the starting functions by randomly choosing the trivial solutions $f(u_s(\mathbf{x}_i)) = 0$ on the coarse gird points $\mathbf{x}_i$, $i = 1, 2, \cdots, m_c$, where m_c is the number of coarse grid points.

3.2.1. 1D starting function

Suppose the 1D domain is $[a, b]$, and $c_0, c_1, \cdots, c_{m_c-1}$ are uniformly coarse points satisfying $a = c_0 < c_1 < \cdots < c_{m_c-1} = b$, and $h = c_{i+1} - c_i = (b-a)/(m_c - 1)$, $i = 0, 1, \cdots, m_c - 2$. The 1D hat basis functions on the interval $[a, b]$ are defined as follows [66]:

$$\phi_0(x) = \begin{cases} (c_1 - x)/h, & a \le x \le c_1, \\ 0, & \text{otherwise}; \end{cases} \tag{11}$$

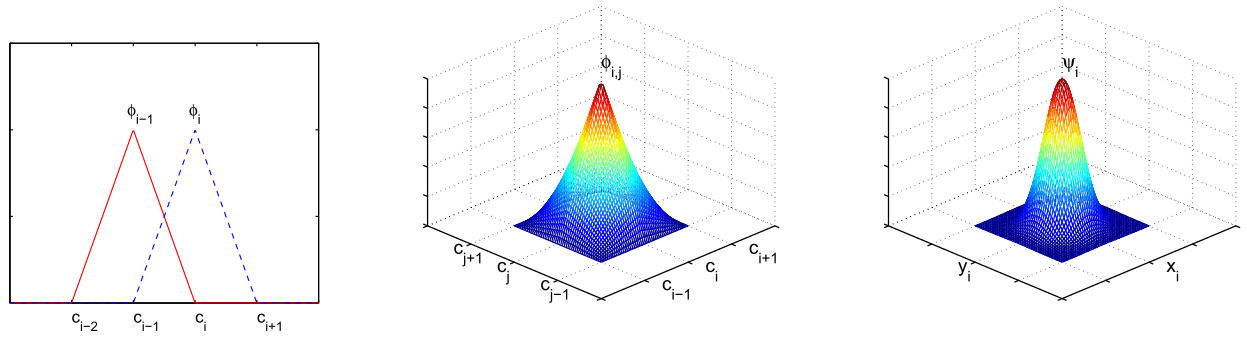
$$\phi_{m_c-1}(x) = \begin{cases} (x - c_{m_c-1})/h, & c_{m_c-2} \le x \le b, \\ 0, & \text{otherwise}, \end{cases} \tag{12}$$

and

$$\phi_i(x) = \begin{cases} (x - c_{i-1})/h, & c_{i-1} \le x \le c_i, \\ (c_{i+1} - x)/h, & c_i \le x \le c_{i+1}, \\ 0, & \text{otherwise}, \end{cases} \quad (1 \le i \le m_c - 2). \tag{13}$$

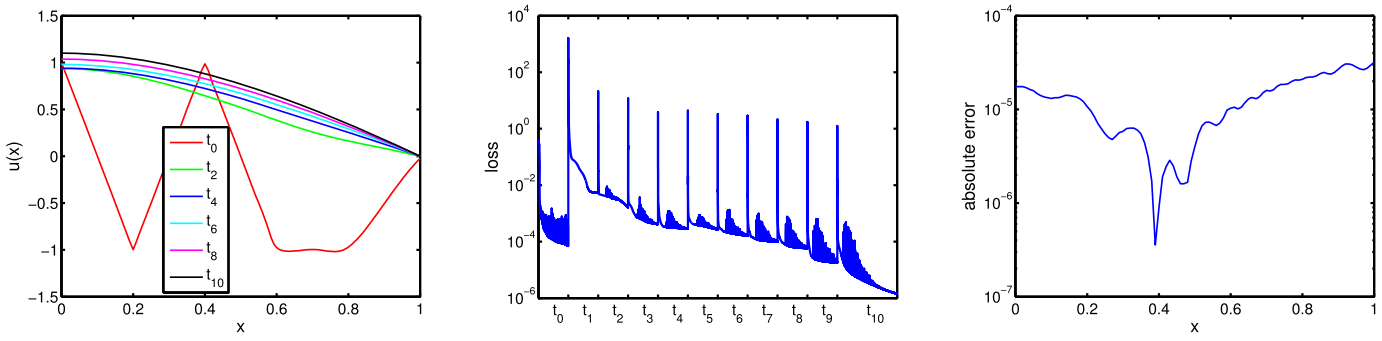
A example of 1D hat functions ϕ_{i-1} and ϕ_i on points c_{i-1} and c_i is illustrated on Fig. 3a. Then, the starting function u_s is defined as a linear combination of 1D hat basis functions $\{\phi_i\}_{i=0}^{m_c-1}$, i.e.,

$$u_s(x) = \sum_{i=0}^{m_c-1} v_i \phi_i, \tag{14}$$



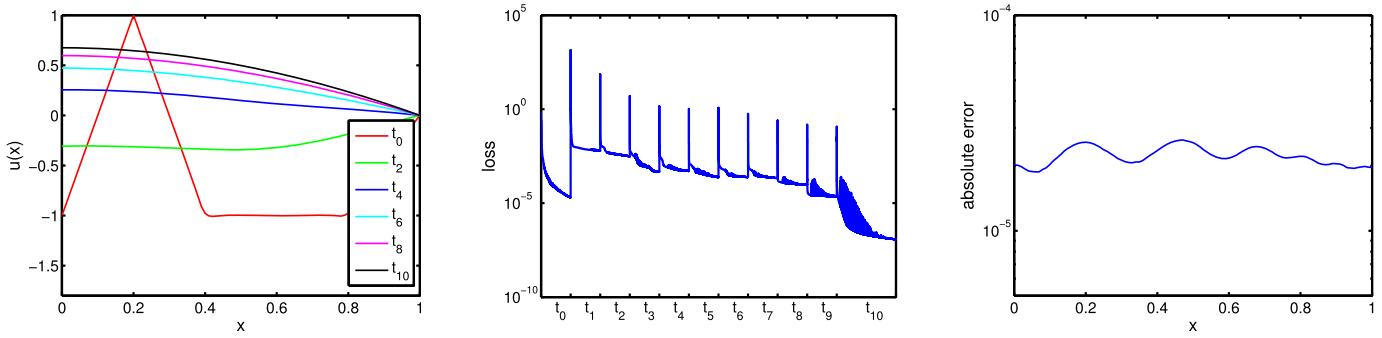
(a) 1D hat functions (b) 2D hat function (c) 2D gaussian function

Fig. 3. Different starting functions: (a) 1D hat functions ϕ_{i-1} and ϕ_i on points c_{i-1} and c_i, respectively; (b) 2D hat function $\phi_{i,j}$ on point (c_i, c_j); (c) 2D gaussian function ψ_i on point (x_i, y_i) and $\sigma = 0.15$.



(a) Solution 1 (b) Loss (c) Absolute error

Fig. 4. The first solution with different homotopy parameter $t = t_i$ $(i = 0, 2, 4, 6, 8, 10)$ (Left), the corresponding training loss with different t_i (Middle), and absolute error between learned solution and real solution (Right) for example 4.1.



(a) Solution 2 (b) Loss (c) Absolute error

Fig. 5. The second solution with different homotopy parameter $t = t_i$ $(i = 0, 2, 4, 6, 8, 10)$ (Left), the corresponding training loss with different t_i (Middle), and absolute error between learned solution and real solution (Right) for example 4.1.

where v_i is randomly chosen from trivial solution set $\mathcal{V}_0$ defined by the nonlinear forcing term. For example, $\mathcal{V}_0 = \{1, -1\}$ if the nonlinear forcing term is taken as $f(u) = u^2 - 1$.

3.2.2. 2D starting function

Similar to 1D starting function, the starting function on the 2D domain $[a, b] \times [a, b]$ is based on 2D hat basis functions which are defined as

$$\phi_{i,j}(x, y) = \phi_i(x)\phi_j(y), \quad 0 \le i, j \le m_c - 1, \tag{15}$$

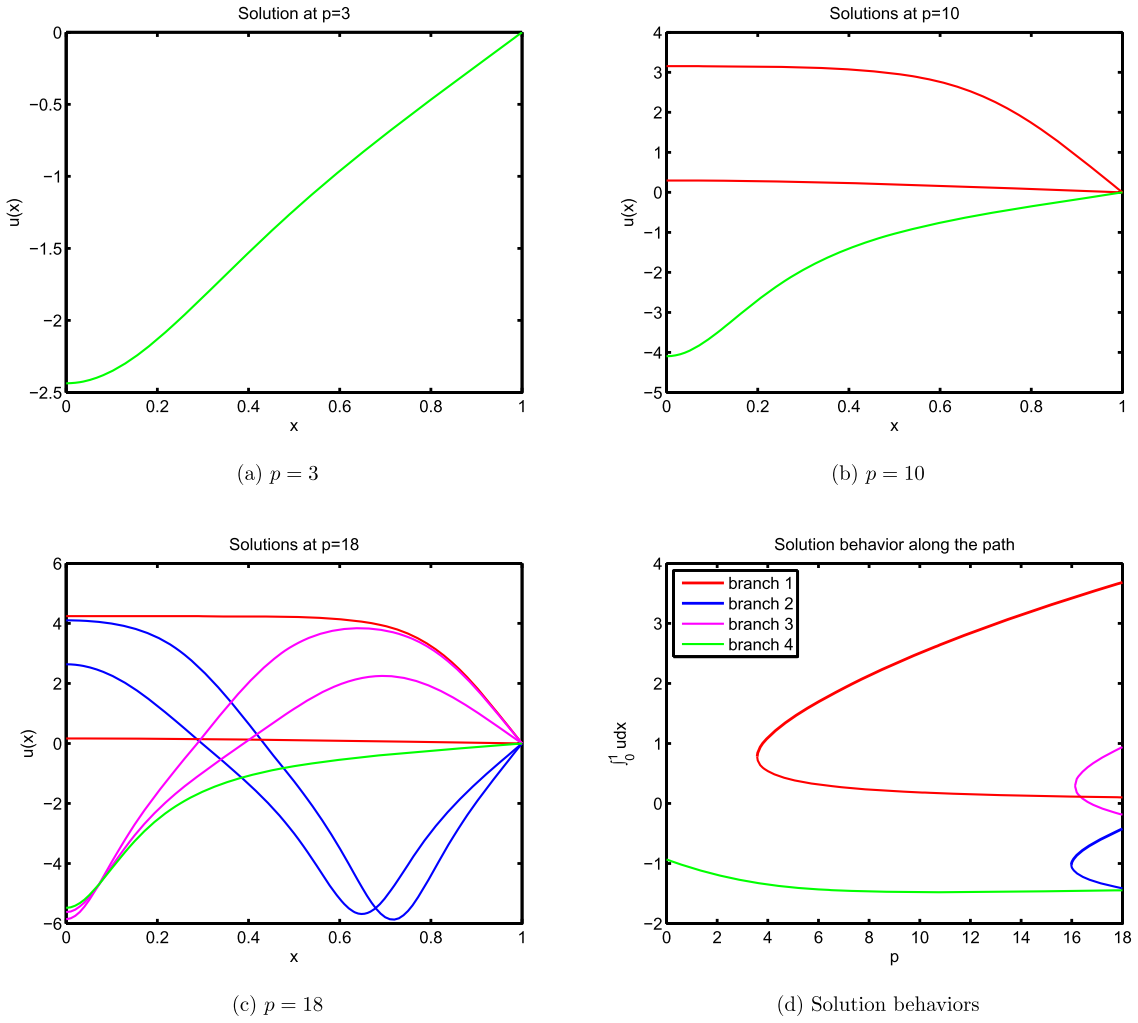
where $\phi_i(x)$ and $\phi_j(y)$ are 1D hat basis functions. A 2D hat function $\phi_{i,j}$ on point (c_i, c_j) is shown in Fig. 3b. Then, the 2D starting function becomes

Table 1
The loss of two solutions of example 4.1 with different network structures.

	width (# neurons)	10	30	50
2 layers	solution one	1.18e−5	2.86e−6	4.08e−6
	solution two	5.11e−5	1.50e−5	9.19e−6
3 layers	solution one	6.61e−6	4.31e−7	8.27e−8
	solution two	1.79e−5	2.37e−7	1.12e−7

$$u_s(x, y) = \sum_{i=0}^{m_c-1} \sum_{j=0}^{m_c-1} v_{i,j}\phi_{i,j}(x, y), \tag{16}$$

coefficient $v_{i,j}$ is randomly chosen from trivial solution set $\mathcal{V}_0$.



(a) $p = 3$ (b) $p = 10$

(c) $p = 18$ (d) Solution behaviors

Fig. 6. Multiple solutions of example 4.2 with different parameters p (a-c) and the solution behavior with respect to the parameter p (d): (a) only one solution for $p = 3$; (b) three solutions for $p = 10$; (c) seven solutions for $p = 18$; (d) $\int_0^1 u(x, p)dx$ v.s. p.

Table 2
The numerical errors in L_2 norm of example 4.1 with different network structures.

	width (# neurons)	10	30	50
2 layers	solution one	4.80e−5	2.54e−5	1.54e−5
	solution two	2.96e−5	2.29e−5	1.30e−5
3 layers	solution one	3.90e−5	1.56e−5	0.41e−5
	solution two	2.93e−5	2.08e−5	1.19e−5

For arbitrary 2D domains, we choose 2D Gaussian functions as the basis function:

$$\psi_i(x, y; \sigma) = \exp\Big(- \frac{(x - x_i)^2 + (y - y_i)^2}{2\sigma^2} \Big), \quad i = 1, 2, \cdots, m_c, \tag{17}$$

where $\{(x_i, y_i)\}_{i=1}^{m_c}$ are uniform or nearly uniform coarse points, σ is the Gaussian parameter which depends on the density of coarse points, and m_c is the number of all coarse points. An illustration of a 2D gauss function ψ_i on point (x_i, y_i) is presented in Fig. 3c with $\sigma = 0.15$. Then, the corresponding starting functions are constructed as

$$u_s(x, y) = \sum_{i=1}^{m_c} v_i \psi_i(x, y; \sigma), \tag{18}$$

with coefficient v_i randomly chosen from trivial solution set $\mathcal{U}_0$.

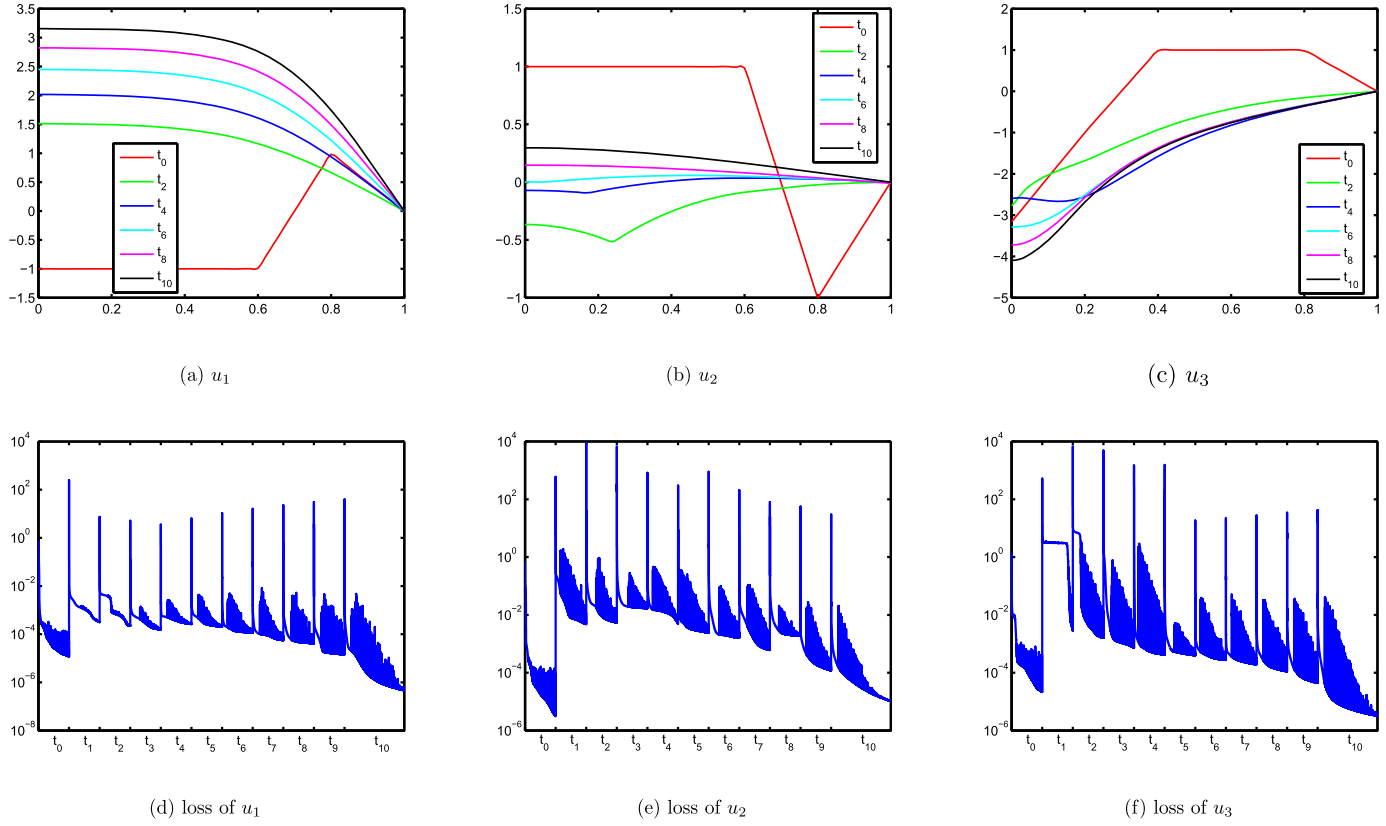
Remark 1. *The hat and Gaussian functions are widely used basis functions. Besides, the random selection of coefficients in the starting function makes HomPINNs learn different solutions of nonlinear differential equations.*

4. Numerical examples

In this section, several examples are presented to show the robustness and the efficiency of HomPINNs for learning multiple solutions of nonlinear PDEs. The first two examples are 1D nonlinear PDEs with mixed boundary conditions, and they are implemented on a personal computer with a 5G Quadro P2000 GPU. The third example is the 2D Henon equation with Dirichlet boundary conditions and nine nontrivial solutions. The last example is the stationary spatial patterns of the well-known Gray-Scott model with Neumann boundary condition on a heart-shaped domain. Both of the two 2D examples are implemented on a scientific workstation with a 24G Titan RTX GPU. All the examples are performed in Python 3.8 utilizing Pytorch library.

In addition, we state some basic setting for all the examples as follows:

1) **neural network setting**
We use 3 hidden layer neural networks with 30 neurons per layer for 1D examples, 80 neurons per layer for 2D examples, and the corresponding activation function is chosen as a hyperbolic tangent function. All the neural networks are initialized using He initialization [67], a default initialization method in the Pytorch library.



(a) u_1 (b) u_2 (c) u_3

(d) loss of u_1 (e) loss of u_2 (f) loss of u_3

Fig. 7. Three solutions with $p = 10$ and the corresponding loss during the homotopy processes in example 4.2.

Table 3
The average loss of multiple solutions example 4.2 for different parameter p with different network structures.

	width(# neurons)	10	30	50
$p = 18$	2 layer	$3.7487e-4$	$3.9978e-4$	$4.9444e-4$
	3 layers	$2.9744e-4$	$1.4039e-4$	$3.8864e-5$
$p = 10$	2 layer	$3.2129e-4$	$8.8537e-5$	$2.4777e-5$
	3 layers	$1.6417e-4$	$8.0367e-6$	$8.4730e-6$
$p = 3$	2 layer	$8.5049e-6$	$2.2326e-6$	$3.3666e-5$
	3 layers	$8.0059e-6$	$1.3030e-6$	$1.2209e-6$

2) **hyperparameters of homotopy process**
In all the examples, we set $\alpha = 10$ and $m = 10$, i.e.,there are 11 steps in a homotopy process, namely, $t_i = 0.1 \times i$, $i = 0, 1, \cdots, 10$. The numbers of collocation points are set as $N_g = 100$ and $N_b = 2$ for 1D examples, $N_g = 5000$ and $N_b = 500$ for example 3 in Subsection 4.3, and $N_g = 9768$ and $N_b = 630$ for example 4 in Subsection 4.4. The number of starting functions are set as $m_s = 20$ in the initial iteration and $m_s = 10$ for later iteration until the solution set keeps unchanged in Algorithm 1. In each homotopy process, the numbers of iterations are set as $N_{I,i} = 20000$ for $i = 0, 1, \cdots, 9$ and $N_{I,10} = 40000$.

3) **optimizer setting**
We use the *Adam* training algorithm [68] in the Pytorch library with the following parameters: $betas = (0.9, 0.999), eps = 10^{-8}, weight\_decay = 0, amsgrad = False, maximize = False$, to solve the optimization problem $\{L(\theta, t_i)\}_{i=0}^{m}$ defined in (8) and (9). The learning rate is set as $\tau_n = \tau_{initial} \times \gamma^{\lceil \frac{n}{\Delta} \rceil}$, which means that we decay the learning rate by γ every Δ iterations. In all examples and for every homotopy Step t_i, we set $\tau_{initial} = 0.002$ and $\Delta = 1000$, while decaying rates $\gamma = 0.85$ for 1D examples and $\gamma = 0.95$ for 2D examples. Besides, after optimizing by *Adam*, the solutions are refined by *LBFGS*, a quasi-Newton method available in Pytorch library with parameters $max\_iter = 10000$ for 1D examples, $max\_iter = 50000$ for 2D examples, $tolerance\_grad = 1.0 \times np.finfo(float).eps$, to obtain a better accuracy.

Table 4
The average loss of multiple solutions in example 4.3 with different network structures.

width (# neurons)	50	80	100
2 layers	$2.58e{-3}$	$1.32e{-3}$	$8.11e{-4}$
3 layers	$9.98e{-4}$	$6.34e{-4}$	$4.81e{-4}$

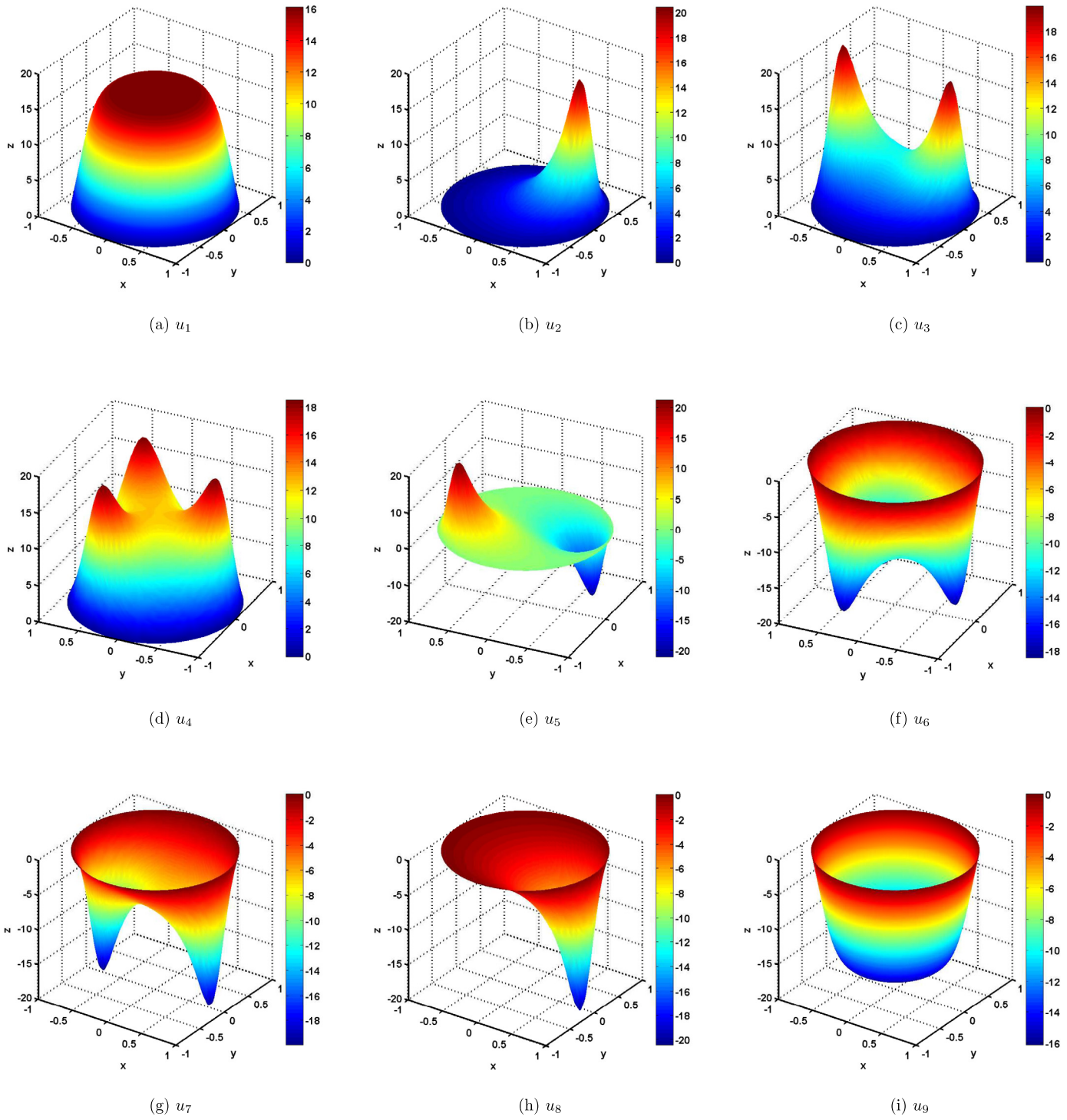
4.1. A 1D example with two solutions

We first consider a 1D nonlinear differential equation on $[0, 1]$ with mixed boundary condition as follows

$$\begin{cases} u_{xx} = -1.2(1 + u^4), \quad x \in [0, 1] \\ u'(0) = u(1) = 0. \end{cases} \tag{19}$$

It is shown in [33] that this example has two analytical solutions. Since Eq. (19) does not have real trivial solutions, we define $\mathcal{U}_0 = \{1, -1, \sqrt{2}, -\sqrt{2}\}$ by solving $f(u) = (u^2 - 1)(u^2 - 2) = 0$. The learning processes of these two solutions and the corresponding loss at each homotopy step are illustrated in Fig. 4 and Fig. 5. The loss and the numerical errors in L_2 norm are shown in Tables 1 and 2, respectively and demonstrate the good accuracy of HomPINNs on different network structures.

Recently, a network-based structure probing deflation method was proposed to identify multiple solutions of differential equations in [69]. The idea is to deflate from one solution to find other solutions. In this example, the first solution u_1 can be found by PINN but the neural



(a) u_1 (b) u_2 (c) u_3

(d) u_4 (e) u_5 (f) u_6

(g) u_7 (h) u_8 (i) u_9

Fig. 8. Nine different solutions of Henon equation (22) in example 4.3: every pair of solutions $\{u_i,\ u_{10-i}\}$ are symmetric about the x-y plane, $i = 1, 2, 3, 4$; Compared with traditional numerical method [72], one more solution u_5 is learned by the proposed HomPINNs.

network deflation is hard to find the other solution because these two solutions are too close to each other [69].

4.2. A 1D example with multiple solutions

Consider the following parametric nonlinear differential equation with multiple solutions,

$$\begin{cases} u_{xx} = u^2(u^2 - p), \quad x \in (0, 1) \\ u'(0) = 0,\ u(1) = 0 \end{cases} \tag{20}$$

with parameter $p > 0$ [70,71]. For any given parameter p, there exist multiple solutions u, moreover, the number of solutions increases as p goes large. In order to compute the multiple solutions with different parameters of equation (20), we take the trivial solution set as $\mathcal{U}_0 = \{1, -1, \sqrt{p}, -\sqrt{p}\}$ by solving a modified nonlinear term $(u^2 - 1)(u^2 - p) = 0$.

The proposed HomPINNs learn one, three and seven solutions for $p = 3$, $p = 10$, and $p = 18$. These results consist with the existing theoretical analysis in [71]. The solutions learned by HomPINNs with different parameters are shown in Fig. 6, and the learning processes of three so-

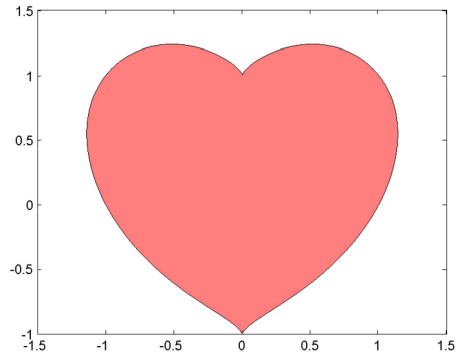


Fig. 9. The computational heart-shaped domain Ω for Gray-Scott model (24) in example 4.4.

lutions at $p = 10$ is presented in Fig. 7. We also summarize the average loss of different solutions for different neural network structures in Table 3 which confirms the robustness of HomPINNs on neural network structures. In addition, to analyze the solution behavior with respect to the parameter p, we define a new loss function with parameter p below:

$$\begin{aligned}\tilde{L}(\theta,p) = &\frac{1}{N_g}\sum_{i=1}^{N_g}\left(\mathcal{L}u_N(\mathbf{x}_i^g;\theta,p) - f(u_N(\mathbf{x}_i^g;\theta,p),p)\right)^2 \\ &+ \alpha\frac{1}{N_b}\sum_{j=1}^{N_b}\left(\mathcal{B}u_N(\mathbf{x}_j^b;\theta,p) - b(\mathbf{x}_j^b)\right)^2,\end{aligned} \tag{21}$$

where $u_N(\mathbf{x}_i^g;\theta,p)$ is the solution of Eq. (21) with any given parameter p. By tracking the solutions from $p = 18$ down to $p = 0$, we discretize the parameter p as $p = p_0, p_1, \cdots, p_{m_p}$, where $p_0 = 18$, $p_{m_p} = 0$ and $m_p + 1$ is the number of discrete points of parameter p. Then we have initial seven solutions $\{u_i(\mathbf{x};\theta_i,p_0)\}_{i=1}^7$ learned by HomPINNs and compute $u(\mathbf{x};\theta_j,p_j)$ $(j > 0)$ by taking $u(\mathbf{x};\theta_{j-1},p_{j-1})$ as the initial guess for the optimization solver until $j = m_p$. Thus we obtain the solution behavior of parametric differential equation (20) shown in Fig. 6d.

4.3. The 2D Henon equation with nine solutions

Next, we consider the following Henon equation [72]

$$\begin{cases}\Delta u + |\mathbf{x}|^7 u^3 = 0, & \mathbf{x}\in\Omega\\ u|_{\partial\Omega} = 0,\end{cases} \tag{22}$$

where $\Omega = \{\mathbf{x} = (x,y)\in\mathbb{R}^2 : |\mathbf{x}| \le 1\}$ and $|\mathbf{x}| = \sqrt{(x^2+y^2)}$. The trivial solution set is given as $\mathcal{U}_0 = \{20, -20, 0\}$ by solving $u(u-20)(u+20) = 0$ which has the same degree as u^3.

Nine different solutions have been learned by the proposed HomPINNs and are presented in Fig. 8. Interestingly, solution u_i is symmetric to solution u_{10-i} about the x-y plane, $i = 1,2,3,4$. The traditional method proposed in [72] only discovers eight solutions $\{u_i, u_{10-i}\}_{i=1}^4$ appearing in pairs which may be caused by some transformations before computation. We do not have any transformations in HomPINNs and define the loss function by using the nonlinear differential equation (22) directly. This is one advantage of the proposed HomPINNs to compensate traditional methods for computing multiple solutions of nonlinear systems. Moreover, we also test the robustness of HomPINNs on different neural network structures and summarize the average loss of different solutions in Table 4.

4.4. Stationary spatial patterns of the Gray-Scott model

The Gray-Scott model, proposed by Gray and Scott [7–9] to describe autocatalytic reactions, takes the following form:

$$\begin{cases}\dfrac{\partial A}{\partial t} = D_A\Delta A + SA^2 - (\mu+\rho)A,\\ \dfrac{\partial S}{\partial t} = D_S\Delta S - SA^2 + \rho(1-S).\end{cases} \tag{23}$$

The steady-state systems with no-flux boundary conditions are written as

$$\begin{cases}D_A\Delta A + SA^2 - (\mu+\rho)A = 0,\\ D_S\Delta S - SA^2 + \rho(1-S) = 0,\\ \dfrac{\partial A}{\partial x}|_{\partial\Omega} = \dfrac{\partial S}{\partial x}S|_{\partial\Omega} = 0,\end{cases} \tag{24}$$

where the domain Ω is set as a heart-shaped domain in $\mathbb{R}^2$ (shown in Fig. 9) formulated by

$$\Omega = \{(x,y)\in\mathbb{R}^2 : (x^2+y^2-1)^3 - x^2y^3 = 0\}.$$

Following [73], the parameters values are set as

$$D_A = 2.5\times10^{-4},\ D_S = 5\times10^{-4},\ \rho = 0.04,\ \text{and}\ \mu = 0.065.$$

Since the diffusion coefficients are small, we normalize the equations to avoid the loss of (24) be too small by dividing the diffusion coefficients. Moreover, to avoid converging to the trivial solution $(A,S) = (0,1)$. we introduce re-scaled variables, $\tilde{\mathbf{u}} = (\tilde{A},\tilde{S})$, such that $\tilde{A} = 2A$ and $\tilde{S} = 2S$. Thus the system of nonlinear differential equations becomes

$$\begin{cases}\mathcal{F}_1(\tilde{A},\tilde{S}) = \Delta\tilde{A} + \dfrac{1}{4D_A}\tilde{S}\tilde{A}^2 - \dfrac{\mu+\rho}{D_A}\tilde{A} = 0,\\ \mathcal{F}_2(\tilde{A},\tilde{S}) = \Delta\tilde{S} - \dfrac{1}{4D_S}\tilde{S}\tilde{A}^2 + \dfrac{\rho}{D_S}(2-\tilde{S}) = 0,\\ \dfrac{\partial\tilde{A}}{\partial x}|_{\partial\Omega} = \dfrac{\partial\tilde{S}}{\partial x}|_{\partial\Omega} = 0,\end{cases} \tag{25}$$

The proposed HomPINNs are firstly used to solve the system (25) by constructing neural networks $\tilde{\mathbf{u}}(\mathbf{x};\theta)$ with a two width output layer to represent two components $\tilde{A}$ and $\tilde{S}$, respectively. The trivial solution set of $\mathcal{F}_1(\tilde{A},\tilde{S})$ is $\mathcal{A}_0 = \{-1,0,2\}$ by solving $(\tilde{A}+1)\tilde{A}(\tilde{A}-2) = 0$. The starting function A_s of A is constructed with Gaussian basis functions by randomly choosing coefficients from $\mathcal{A}_0$, and the starting function of S is defined as $S_s = 2 - A_s$. The HomPINNs learn 25 different solutions which are shown in Figs. 10 and 11 for both $A(x,y)$ and $S(x,y)$ by choosing 60 starting functions. The average loss of the 25 learned neural networks is 6.40×10^{-4}, and the average residual of the governing equations is 3.83×10^{-4}.

5. Conclusion

The homotopy continuation method has shown advantages for computing multiple solutions of nonlinear differential equations. The NN-based techniques, including physics-informed neural networks, have demonstrated the effectiveness of learning the solution of differential equations. By combining the homotopy continuation method and physics-informed neural networks, we developed a new deep learning framework, HomPINNs, in this manuscript for solving multiple solutions of nonlinear differential equations. In particular, the proposed HomPINN generally contains several sub-networks: the first one is called starting neural network which is to approximate the starting functions, and the other networks gradually the target system which is the nonlinear differential equation that we want to solve. For the training process, we use the previous well-trained sub-network as the initial guess to train the current sub-network and eventually recover the multiple solutions of nonlinear differential equations. We have tested the HomPINNs on various numerical benchmark problems to show the capability of HomPINNs for learning multiple solutions and the robustness of HomPINNs on different network structures. Moreover, it also shows the efficiency of solving nonlinear differential equations on arbitrary

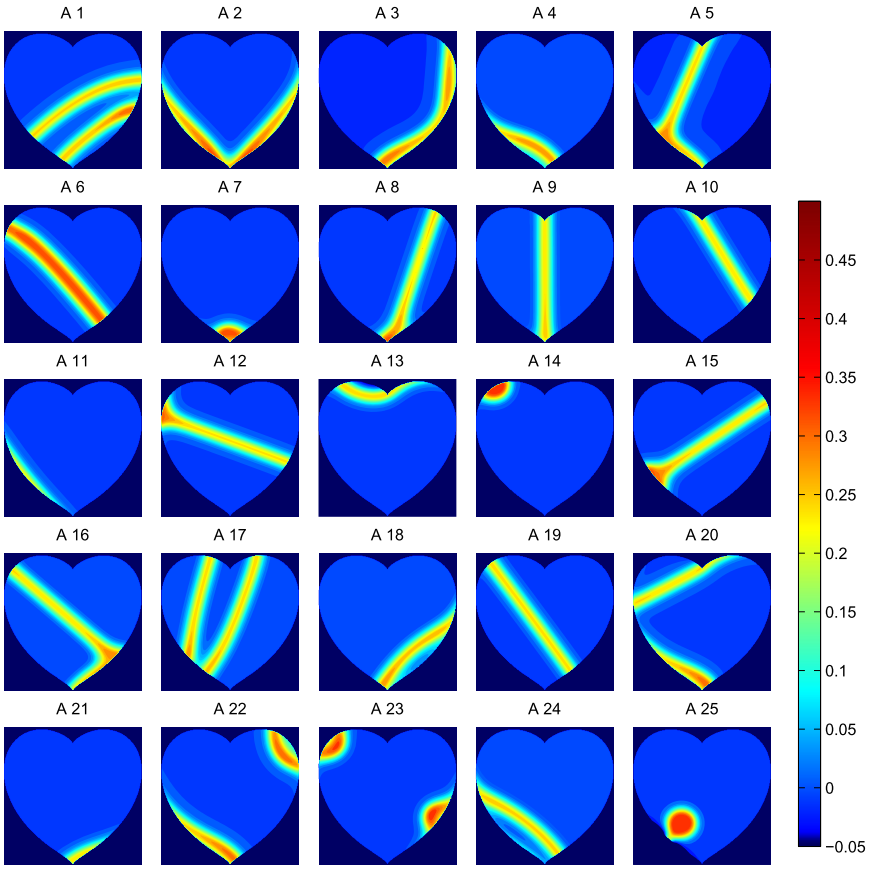


Fig. 10. The contour plots of $A(x, y)$ for example 4.4.

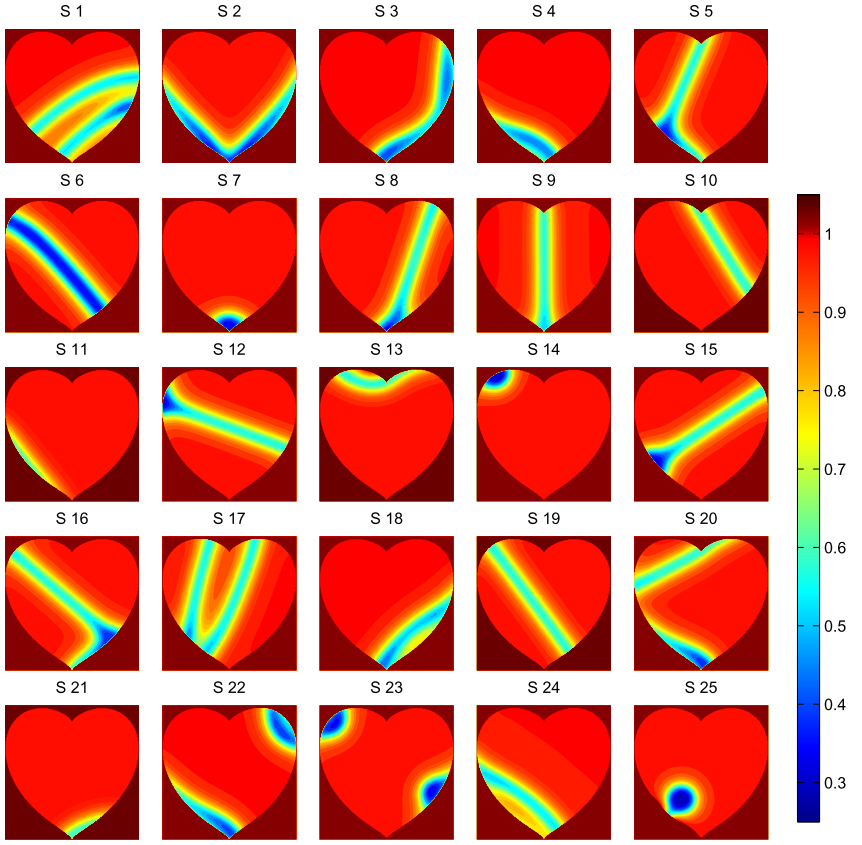


Fig. 11. The contour plots of $S(x, y)$ for example 4.4.

domains which is difficult for traditional methods. In the future, we will apply the HomPINNs to other differential operators and other problems involving multiple solutions such as eigenvalue problems of differential equations. Another future direction is to explore the effect of different activation functions on HomPINNs, for instance, the adaptive activation functions introduced in [74,75,47]. More, the choice of starting functions needs to be further studied to better capture solution structures of nonlinear PDEs.

Acknowledgements

Y. H. is supported by China Scholarship Council (CSC) under award 202006280431. W. H. is supported by National Science Foundation (NSF) grant DMS-1818769 and DMS-2052685. G. L. gratefully acknowledges the support of the National Science Foundation (DMS-1555072, DMS-1736364, DMS-2053746, and DMS-2134209), and Brookhaven National Laboratory Subcontract 382247, and U.S. Department of Energy (DOE) Office of Science Advanced Scientific Computing Research program DE-SC0021142).

References

[1] V. Cristini, X. Li, J.S. Lowengrub, S.M. Wise, Nonlinear simulations of solid tumor growth using a mixture model: invasion and branching, J. Math. Biol. 58 (2009) 723–763.
[2] W. Hao, J.D. Hauenstein, B. Hu, A.J. Sommese, A three-dimensional steady-state tumor system, Appl. Math. Comput. 218 (2011) 2661–2669.
[3] T. Puu, Nonlinear economic dynamics, in: Nonlinear Economic Dynamics, Springer, 1991, pp. 1–7.
[4] L.F. Petrov, Nonlinear effects in economic dynamic models, Nonlinear Anal., Theory Methods Appl. 71 (2009) e2366–e2371.
[5] W. Hao, R.I. Nepomechie, A.J. Sommese, Completeness of solutions of Bethe's equations, Phys. Rev. E 88 (2013) 052113.
[6] K. Parand, M. Shahini, M. Dehghan, Rational Legendre pseudospectral approach for solving nonlinear differential equations of Lane–Emden type, J. Comput. Phys. 228 (2009) 8830–8840.
[7] P. Gray, S. Scott, Autocatalytic reactions in the isothermal, continuous stirred tank reactor: isolas and other forms of multistability, Chem. Eng. Sci. 38 (1983) 29–43.
[8] P. Gray, S. Scott, Autocatalytic reactions in the isothermal, continuous stirred tank reactor: oscillations and instabilities in the system a + 2b → 3b; b → c, Chem. Eng. Sci. 39 (1984) 1087–1097.
[9] P. Gray, S. Scott, Sustained oscillations and other exotic patterns of behavior in isothermal reactions, J. Phys. Chem. 89 (1985) 22–32.
[10] T.Y. Hou, J.S. Lowengrub, M.J. Shelley, Boundary integral methods for multicomponent fluids and multiphase materials, J. Comput. Phys. 169 (2001) 302–362.
[11] J.E. Pearson, Complex patterns in a simple system, Science 261 (1993) 189–192.
[12] U. Frisch, S. Matarrese, R. Mohayaee, A. Sobolevski, A reconstruction of the initial conditions of the universe by optimal mass transportation, Nature 417 (2002) 260–262.
[13] L.A. Caffarelli, M. Milman, et al., Monge Ampere equation: applications to geometry and optimization: applications to geometry and optimization: NSF-CBMS conference on the Monge Ampère equation, applications to geometry and optimization, July 9-13, 1997, Florida Atlantic University, vol. 226, American Mathematical Soc., 1999.
[14] E. Tadmor, A review of numerical methods for nonlinear partial differential equations, Bull. Am. Math. Soc. 49 (2012) 507–554.
[15] X. Feng, R. Glowinski, M. Neilan, Recent developments in numerical methods for fully nonlinear second order partial differential equations, SIAM Rev. 55 (2013) 205–267.
[16] W. Lo, L. Chen, M. Wang, Q. Nie, A robust and efficient method for steady state patterns in reaction–diffusion systems, J. Comput. Phys. 231 (2012) 5062–5077.
[17] K.-C. Chang, Variational methods for non-differentiable functionals and their applications to partial differential equations, J. Math. Anal. Appl. 80 (1981) 102–129.
[18] P. Rabinowitz, Minimax Methods in Critical Point Theory with Applications to Differential Equations, Vol. 65, American Mathematical Soc., 1986.
[19] J. Zhou, Solving multiple solution problems: computational methods and theory revisited, Commun. Appl. Math. Comput. 31 (2017) 1–31.
[20] Z. Li, Z. Wang, J. Zhou, A new augmented singular transform and its partial Newton-correction method for finding more solutions, J. Sci. Comput. 71 (2017) 634–659.
[21] L. Lin, X. Cheng, E. Weinan, A. Shi, P. Zhang, A numerical method for the study of nucleation of ordered phases, J. Comput. Phys. 229 (2010) 1797–1809.
[22] E. Allgower, K. Georg, Introduction to Numerical Continuation Methods, Vol. 45, SIAM, 2003.
[23] X. Cai, W. Gropp, D. Keyes, R. Melvin, D. Young, Parallel Newton–Krylov–Schwarz algorithms for the transonic full potential equation, SIAM J. Sci. Comput. 19 (1998) 246–265.
[24] X. Cai, D. Keyes, Nonlinearly preconditioned inexact Newton algorithms, SIAM J. Sci. Comput. 24 (2002) 183–200.
[25] J. Wilkinson, Rounding Errors in Algebraic Processes, Courier Corporation, 1994.
[26] M. Robinson, C. Luo, P. Farrell, R. Erban, A. Majumdar, From molecular to continuum modelling of bistable liquid crystal devices, Liq. Cryst. 44 (2017) 2267–2284.
[27] P. Farrell, A. Birkisson, S. Funke, Deflation techniques for finding distinct solutions of nonlinear partial differential equations, SIAM J. Sci. Comput. 37 (2015) A2026–A2045.
[28] A.J. Sommese, C.W. Wampler, et al., The Numerical Solution of Systems of Polynomials Arising in Engineering and Science, World Scientific, 2005.
[29] C.W. Wampler, A.J. Sommese, Numerical algebraic geometry and algebraic kinematics, Acta Numer. 20 (2011) 469–567.
[30] W. Hao, B. Hu, A.J. Sommese, Numerical algebraic geometry and differential equations, in: Future Vision and Trends on Shapes, Geometry and Algebra, Springer, 2014, pp. 39–53.
[31] Y. Wang, W. Hao, G. Lin, Two-level spectral methods for nonlinear elliptic equations with multiple solutions, SIAM J. Sci. Comput. 40 (2018) B1180–B1205.
[32] W. Hao, J. Hesthaven, G. Lin, B. Zheng, A homotopy method with adaptive basis selection for computing multiple solutions of differential equations, J. Sci. Comput. 82 (2020) 1–17.
[33] W. Hao, J.D. Hauenstein, B. Hu, A.J. Sommese, A bootstrapping approach for computing multiple solutions of differential equations, J. Comput. Appl. Math. 258 (2014) 181–190.
[34] A. Owens, D. Filkin, Efficient training of the back propagation network by solving a system of stiff ordinary differential equations, in: Proceedings IEEE/INNS International Joint Conference of Neural Networks, 1989, pp. 381–386.
[35] M. Dissanayake, N. Phan-Thien, Neural-network-based approximations for solving partial differential equations, Commun. Numer. Methods Eng. 10 (1994) 195–201.
[36] I.E. Lagaris, A. Likas, D.I. Fotiadis, Artificial neural networks for solving ordinary and partial differential equations, IEEE Trans. Neural Netw. 9 (1998) 987–1000.
[37] K. Hornik, M. Stinchcombe, H. White, Multilayer feedforward networks are universal approximators, Neural Netw. 2 (1989) 359–366.
[38] T. Chen, H. Chen, Universal approximation to nonlinear operators by neural networks with arbitrary activation functions and its application to dynamical systems, IEEE Trans. Neural Netw. 6 (1995) 911–917.
[39] J.W. Siegel, J. Xu, Approximation rates for neural networks with general activation functions, Neural Netw. 128 (2020) 313–321.
[40] L. Lu, P. Jin, G. Pang, Z. Zhang, G.E. Karniadakis, Learning nonlinear operators via deeponet based on the universal approximation theorem of operators, Nat. Mach. Intell. 3 (2021) 218–229.
[41] M. Raissi, P. Perdikaris, G.E. Karniadakis, Physics-informed neural networks: a deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations, J. Comput. Phys. 378 (2019) 686–707.
[42] G. Pang, L. Lu, G.E. Karniadakis, fPINNs: fractional physics-informed neural networks, SIAM J. Sci. Comput. 41 (2019) A2603–A2626.
[43] D. Zhang, L. Guo, G.E. Karniadakis, Learning in modal space: solving time-dependent stochastic PDEs using physics-informed neural networks, SIAM J. Sci. Comput. 42 (2020) A639–A665.
[44] Z. Mao, A.D. Jagtap, G.E. Karniadakis, Physics-informed neural networks for high-speed flows, Comput. Methods Appl. Mech. Eng. 360 (2020) 112789.
[45] Q. Lou, X. Meng, G.E. Karniadakis, Physics-informed neural networks for solving forward and inverse flow problems via the Boltzmann-BGK formulation, J. Comput. Phys. (2021) 110676.
[46] K. Shukla, A.D. Jagtap, J.L. Blackshire, D. Sparkman, G.E. Karniadakis, A physics-informed neural network for quantifying the microstructural properties of polycrystalline nickel using ultrasound data: a promising approach for solving inverse problems, IEEE Signal Process. Mag. 39 (2021) 68–77.
[47] A.D. Jagtap, Y. Shin, K. Kawaguchi, G.E. Karniadakis, Deep Kronecker neural networks: a general framework for neural networks with adaptive activation functions, Neurocomputing 468 (2022) 165–180.
[48] K. Shukla, A.D. Jagtap, G.E. Karniadakis, Parallel physics-informed neural networks via domain decomposition, J. Comput. Phys. 447 (2021) 110683.
[49] A.D. Jagtap, G.E. Karniadakis, Extended physics-informed neural networks (XPINNs): a generalized space-time domain decomposition based deep learning framework for nonlinear partial differential equations, Commun. Comput. Phys. 28 (2020) 2002–2041.
[50] A.D. Jagtap, E. Kharazmi, G.E. Karniadakis, Conservative physics-informed neural networks on discrete domains for conservation laws: applications to forward and inverse problems, Comput. Methods Appl. Mech. Eng. 365 (2020) 113028.
[51] A.D. Jagtap, Z. Mao, N. Adams, G.E. Karniadakis, Physics-informed neural networks for inverse problems in supersonic flows, arXiv preprint, arXiv:2202.11821, 2022.
[52] T. De Ryck, A.D. Jagtap, S. Mishra, Error estimates for physics informed neural networks approximating the Navier-Stokes equations, arXiv preprint, arXiv:2203.09346, 2022.
[53] J. Sirignano, K. Spiliopoulos, DGM: a deep learning algorithm for solving partial differential equations, J. Comput. Phys. 375 (2018) 1339–1364.
[54] E. Weinan, B. Yu, The deep Ritz method: a deep learning-based numerical algorithm for solving variational problems, Commun. Math. Stat. 1 (2018) 1–12.
[55] A.G. Baydin, B.A. Pearlmutter, A.A. Radul, J.M. Siskind, Automatic differentiation in machine learning: a survey, J. Mach. Learn. Res. 18 (2018).

[56] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, A. Lerer, Automatic Differentiation in Pytorch, 2017.
[57] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, et al., Tensorflow: a system for large-scale machine learning, in: 12th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 16), 2016, pp. 265–283.
[58] G. Forsythe, W. Wasow, Finite Difference Methods for Partial Differential Equations, John Wiley and Sons, 1960.
[59] P. Ciarlet, The Finite Element Method for Elliptic Problems, SIAM, 2002.
[60] J. Shen, T. Tang, L. Wang, Spectral Methods: Algorithms, Analysis and Applications, Vol. 41, Springer Science & Business Media, 2011.
[61] A. Sommese, C. Wampler, The Numerical Solution of Systems of Polynomials Arising in Engineering and Science, Vol. 99, World Scientific, 2005.
[62] W. Hao, B. Hu, A. Sommese, Numerical algebraic geometry and differential equations, in: Future Vision and Trends on Shapes, Geometry and Algebra, Springer, 2014, pp. 39–53.
[63] D. Bates, J. Hauenstein, A. Sommese, C. Wampler, Numerically Solving Polynomial Systems with Bertini, Vol. 25, SIAM, 2013.
[64] A. Leykin, Numerical algebraic geometry, J. Softw. Algebra Geom. 3 (2011) 5–10.
[65] A. Morgan, A. Sommese, C. Wampler, A product-decomposition bound for Bezout numbers, SIAM J. Numer. Anal. 32 (1995) 1308–1325.
[66] E. Babolian, Z. Masouri, S. Hatamzadeh-Varmazyar, Numerical solution of nonlinear Volterra–Fredholm integro-differential equations via direct method using triangular functions, Comput. Math. Appl. 58 (2009) 239–247.
[67] K. He, X. Zhang, S. Ren, J. Sun, Delving deep into rectifiers: surpassing human-level performance on ImageNet classification, in: 2015 IEEE International Conference on Computer Vision (ICCV), 2015, pp. 1026–1034.
[68] D.P. Kingma, J. Ba Adam, A method for stochastic optimization, arXiv preprint, arXiv:1412.6980, 2014.
[69] Y. Gu, C. Wang, H. Yang, Structure probing neural network deflation, J. Comput. Phys. 434 (2021) 110231.
[70] C. Chen, Z. Xie, Structure of multiple solutions for nonlinear differential equations, Sci. China Ser. A, Math. 47 (2004) 172–180.
[71] W. Hao, C. Zheng, An adaptive homotopy method for computing bifurcations of nonlinear parametric systems, J. Sci. Comput. 82 (2020) 1–19.
[72] Z.-X. Li, Z.-H. Yang, H.-L. Zhu, A bifurcation method for solving multiple positive solutions to the boundary value problem of the Henon equation on a unit disk, Comput. Math. Appl. 62 (2011) 3775–3784.
[73] W. Hao, C. Xue, Spatial pattern formation in reaction–diffusion models: a computational approach, J. Math. Biol. 80 (2020) 521–543.
[74] A.D. Jagtap, K. Kawaguchi, G.E. Karniadakis, Adaptive activation functions accelerate convergence in deep and physics-informed neural networks, J. Comput. Phys. 404 (2020) 109136.
[75] A.D. Jagtap, K. Kawaguchi, G. Em Karniadakis, Locally adaptive activation functions with slope recovery for deep and physics-informed neural networks, Proc. R. Soc. A 476 (2020) 20200334.