VETOR

Numerical Solutions of Differential Equations using Artificial Neural Networks☆

Soluções Numéricas de Equações Diferenciais com Redes Neurais Artificiais

José Miguel Aroztegui Massera†, Thiago José Machado

Centro de Informática, Universidade Federal da Paraíba, João Pessoa, Brasil
†**Corresponding author:** jose.miguel@ci.ufpb.br

Abstract

In this article, we study a way to numerically solve differential equations using neural networks. Basically, we rewrite the differential equation as an optimization problem, where the parameters related to the neural network are optimized. The proposal of this work constitutes a variation of the formulation introduced by Lagaris et al. [1], differing mainly in the form of the construction of the approximate solution. Although we only deal with first and second order ordinary differential equations, the numerical results show the efficiency of the proposed method. Furthermore, this method has a great potential, due to the amount of differential operators and applications in which it can be used.

Keywords

Neural networks • Differential equations • Optimization

Resumo

Neste artigo, vamos estudar uma forma de resolver numericamente equações diferenciais utilizando redes neurais. Basicamente, reescrevemos a equação diferencial como um problema de otimização, onde os parâmetros associados à rede neural são otimizados. A proposta deste trabalho apresentada aqui constitui uma variação da formulação introduzida por Lagaris et al. [1], diferenciando-se principalmente na forma de construção da solução aproximada. Apesar de lidarmos apenas com equações diferenciais ordinárias de primeira e segunda ordens, os resultados numéricos mostram a eficiência do método proposto. Além disso, ele possui bastante potencial, devido a quantidade de equações diferenciais e aplicações nas quais ele pode ser utilizado.

Palavras-chave

Redes neurais • Equações diferenciais • Otimização

1 Introduction

Neural networks represent an important branch of machine learning. The main idea is that the neural network must emulate human brains. To achieve this, a set of input and output data is used to train the network, which is called dataset. Once trained, the network must be able to recognize patterns, make predictions and/or make classifications of new data. The theoretical foundations of neural networks mainly involve concepts of Optimization and Computational Linear Algebra [2, 3]. Regarding applications, there is a large list of works in various fields

☆This article is an extended version of the work presented at the Joint XXIV ENMC National Meeting on Computational Modelling and XII ECTM Meeting on Science and Technology of Materials, held in webinar mode, from October 13th to 15th, 2021.

of knowledge, among which we list the following: inverse problems, optimal control problems, pricing models, petroleum engineering, topological optimization, chemical reactions, electromagnetism and text processing. For an overview of applications involving neural networks and existing work in the literature, the reader can consult [4].

As shown in Cybenko [5] and Hornik et al. [6], continuous functions on compact subsets of $\mathbb{R}^n$ can be uniformly approximated by linear combinations of sigmoidal functions. Therefore, it is possible to define and train neural networks to approximate continuous functions and, consequently, we can use the derivatives of these neural networks to obtain approximate solutions of differential equations, as long as the highest order derivative of the solution is continuous. For boundary value problems, there is a possibility that the solution is not continuous. However, even for this type of problem it is possible to approximate the solution by neural networks, as can be seen in Lagaris et al. [1], Khemchandani et al. [7] and Avrutskiy [8].

In the original methodology introduced in Lagaris et al. [1], when solving a differential equation with boundary conditions, the approximation of the exact solution, which is called test function, is configured in order to satisfy the conditions imposed on the boundary, so that the neural network contained in it must be trained to fit only to the differential equation. The proposal of this work consists of an adaptation of this methodology, where two alternative formulations are presented. In the first one, just one neural network is taken that must simultaneously satisfy the boundary conditions and fit the differential equation. In the second one, a test function formed by two independent neural networks is used, where one of them must be trained to satisfy the boundary conditions, while the other must fit the differential equation.

In the next section, we present a general formulation of differential equations, with multi-index notation and differential operators. This general formulation allows to justify and extend both the method presented in Lagaris et al. [1] and the formulations proposed in this work. In other words, in order to facilitate the explanation of the methods covered in this work, differential equations, boundary conditions and optimization problems are rewritten in terms of the differential operators.

The paper is organized as follows. In section 2, the general problem of differential equations is formulated and the test functions with neural networks that will approach the exact solution to the differential equations are established. In order to fit neural networks to the relations imposed by the differential equations, general problems of optimization are defined. In Section 3, the optimization problems are defined for the specific cases of first and second order Ordinary Differential Equations (ODEs). Then, in Section 4, details of how to compute ODE solutions are defined from numerical optimization solutions. As expected, it turns out that the defined solution by the test functions correspond to good approximations for solving first and second order ODEs. In the latter case, different types of boundary conditions are considered. Finally, in Section 5 we present the conclusions of the work.

2 General method to solve differential equations with neural networks

Let $\Omega \subset \mathbb{R}^n$ be an open and bounded set, with boundary denoted by $\Gamma = \partial\Omega$, and $\Psi : \Omega \to \mathbb{R}$ a function having k continuous derivatives. According to Evans [9], a quasilinear partial differential equation of order k can be written as follows:

$$\sum_{|\alpha|=k} D^\alpha \Psi \cdot \varphi_1(D^{k-1}\Psi, \dots, D\Psi, \Psi, x) = \varphi_2(D^{k-1}\Psi, \dots, D\Psi, \Psi, x), \quad x \in \Omega. \tag{1}$$

In (1), we are using the multi-index notation. So we have that $\alpha = (\alpha_1, \dots, \alpha_n)$, with each entry being a non-negative integer and $|\alpha| = \alpha_1 + \alpha_2 + \dots + \alpha_n$. Also, for every $j \in \{1, \dots, k\}$, the j-order derivative of Ψ is defined as:

$$D^j \Psi = \left(\frac{\partial^{\alpha_1}}{\partial x_1^{\alpha_1}} \frac{\partial^{\alpha_2}}{\partial x_2^{\alpha_2}} \cdots \frac{\partial^{\alpha_n}}{\partial x_n^{\alpha_n}} \right) \Psi, \quad \text{with } |\alpha| = j. \tag{2}$$

From (1), we define the differential operators $\mathcal{G}[\Psi(x)]$ and $\mathcal{H}[\Psi(x)]$, where $\mathcal{G}[\Psi(x)]$ and $\mathcal{H}[\Psi(x)]$ define the left and right hand side of (1), respectively. Let $\mathcal{B}[\Psi(x)]$ be a trace operator of order equal to or less than $k-1$ and h a continuous function defined over Γ. With those elements, consider the following boundary value problem:

$$\begin{cases} \mathcal{G}[\Psi(x)] = \mathcal{H}[\Psi(x)] & \text{in} \quad \Omega, \\ \mathcal{B}[\Psi(x)] = h(x) & \text{on} \quad \Gamma. \end{cases} \tag{3}$$

Here we will not deal with issues related to regularity of the domain or its boundary. Let's just assume that the problem (3) has a unique solution.

The approach to obtain a function that approximates the exact solution of (3), proposed in Lagaris et al. [1], is presented below. From this point on, it is assumed in (1) that the function φ_1 is constant and that φ_2 is linear. In this approach the test function Ψ_t is given by:

$$\Psi_t(x, p) = A(x) + F(x, N(x, p)), \tag{4}$$

where the function $A : \Omega \to \mathbb{R}$ is defined to ensure that Ψ_t satisfies the boundary condition in (3). However, this term does not should interfere with the differential equation, which makes A a solution to the following problem:

$$\begin{cases} \mathcal{G}[A(x)] &= \mathcal{H}[A(x)] \quad \text{in} \quad \Omega, \\ \mathcal{B}[A(x)] &= h(x) \qquad \text{on} \quad \Gamma. \end{cases} \tag{5}$$

As a consequence of the boundary condition in (5), the function $F : \Omega \to \mathbb{R}$ must be taken in such a way that $\mathcal{B}[F(x, N(x,p))] = 0$ on Γ. We also have that the function $N : \Omega \times \mathbb{R}^{H(n+2)} \to \mathbb{R}$ is a neural network with n inputs, containing a layer composed of H sigmoids and a linear output, that is,

$$N(x,p) = \sum_{i=1}^{H} v_i \, \sigma\left(\sum_{j=1}^{n} \omega_{ij} x_j + u_i \right). \tag{6}$$

In this case, we have that the parameter p is an array in $\mathbb{R}^{H(n+2)}$ with weights v_i, ω_{ij} and bias u_i of the neural architecture, with $i \in \{1, \dots, H\}$ and $j \in \{1, \dots, n\}$. We also have the sigmoid function $\sigma : \mathbb{R} \to \mathbb{R}$ which is defined as follows:

$$\sigma(t) = \frac{1}{1 + e^{-t}}. \tag{7}$$

Finally, the strategy adopted in [1] consists in solving the following optimization problem:

$$\min_{p} E(p) := \sum_{y_i \in \widehat{\Omega}} (\mathcal{G}[\Psi_t](y_i, p) - \mathcal{H}[\Psi_t](y_i, p))^2, \tag{8}$$

where $\widehat{\Omega}$ is a discrete subset of Ω. In other words, the want the optimal adjustment of the parameters of the neural network contained in Ψ_t, in order to minimize the distance between $\mathcal{G}[\Psi_t]$ and $\mathcal{H}[\Psi_t]$. It is noteworthy that, although the optimization is done only with respect to the parameter p, the amount of points in $\widehat{\Omega}$ also exerts a lot of influence in the numeric results.

Based on the method introduced by Lagaris et al. [1], we propose two alternative formulations, considering different test functions. Let's denote by Ψ_1 and Ψ_2 the respective test functions of each formulation. In both cases, the functions $A = A(x)$ and $F = F(x, N)$, that appear in (4), are not used. We only consider one or a combinations of two neural networks to define the test function.

In the first proposed formulation, we assume that $\Psi_1 : \Omega \times \mathcal{P} \to \mathbb{R}$ is given by

$$\Psi_1(x,p) = N(x,p), \tag{9}$$

where N is defined by (6). In this case, we can define the optimization problem as follows:

$$\begin{cases} \min\limits_{p \in \mathcal{P}} E(p) := \sum\limits_{y_i \in \widehat{\Omega}} (\mathcal{G}[\Psi_1](y_i, p) - \mathcal{H}[\Psi_1](y_i, p))^2 \\ \text{subject to } \mathcal{B}[\Psi_1](y_i, p) = h(y_i) \text{ on } \widehat{\Gamma}, \end{cases} \tag{10}$$

where $\widehat{\Gamma}$ is a discretization of boundary Γ.

Regarding the second proposed formulation, let's consider the test function $\Psi_2 : \Omega \times \mathcal{P}_1 \times \mathcal{P}_2 \to \mathbb{R}$ defined by:

$$\Psi_2(x, p_1, p_2) = N_1(x, p_1) + N_2(x, p_2), \tag{11}$$

where $N_1 : \Omega \times \mathcal{P}_1 \to \mathbb{R}$ and $N_2 : \Omega \times \mathcal{P}_2 \to \mathbb{R}$ are neural networks with n inputs and linear outputs, each consisting of a combination of sigmoids, that is,

$$N_1(x, p_1) = \sum_{i=1}^{H_1} \alpha_i \, \sigma\left(\sum_{j=1}^{n} \beta_{ij} x_j + \gamma_i \right) \quad \text{e} \quad N_2(x, p_2) = \sum_{i=1}^{H_2} a_i \, \sigma\left(\sum_{j=1}^{n} b_{ij} x_j + c_i \right). \tag{12}$$

In this case, we have that network N_1 must minimize the distance between the operators $\mathcal{G}$ and $\mathcal{H}$ inside the domain Ω and the network N_2 serves to approximate boundary conditions on Γ. Then, taking $\widehat{\Omega} \subset \Omega$ and $\widehat{\Gamma} \subset \Gamma$ as the respective domain and its boundary discretizations, the optimization problem with constraint is defined as follows:

$$\begin{cases} \min\limits_{(p_1, p_2) \in \mathcal{P}_1 \times \mathcal{P}_2} E(p_1, p_2) := \sum\limits_{y_i \in \widehat{\Omega}} (\mathcal{G}[\Psi_2](y_i, p_1) - \mathcal{H}[\Psi_2](y_i, p_1))^2 \\ \text{subject to } \mathcal{B}[N_1](y_i, p_1) = 0 \text{ and } \mathcal{B}[N_2](y_i, p_2) = h(y_i) \text{ on } \widehat{\Gamma} \end{cases} \tag{13}$$

In order to explain the proposed methods, in the next section will be presented some applications in ordinary differential equations of first and second orders.

3 Applications of the method in Ordinary Differential Equations

In this section, we will apply the formulations presented in the previous section to first and second order ODEs. Since we are dealing with only one-dimensional problems, the neural networks introduced above are adapted to this scenario. In other words, the networks $N = N(x, p)$, $N_1 = N_1(x, p_1)$ and $N_2 = N_2(x, p_2)$, defined in (6) and (12) assume, respectively, the following definitions:

$$\begin{cases} N(x,p) = \sum_{i=1}^{H} v_i \sigma(\omega_i x + u_i) \\ N_1(x,p_1) = \sum_{i=1}^{H_1} \alpha_i \, \sigma(\beta_i x + \gamma_i) \\ N_2(x,p_2) = \sum_{i=1}^{H_2} a_i \, \sigma(b_i x + c_i) \end{cases} \tag{14}$$

3.1 First Order Ordinary Differential Equations

Let's consider the following initial value problem:

$$\begin{cases} \dfrac{d\Psi}{dx}(x) &= f(x, \Psi(x)); \quad x \in (x_0, x_1), \\ \Psi(x_0) &= A_0, \end{cases} \tag{15}$$

where $A_0 \in \mathbb{R}$ and f are taken so that the problem (15) has a unique solution and that this solution has continuous derivative.

From the formulation introduced in [1], it follows that the test function has the following form:

$$\Psi_0(x, p) = A_0 + (x - x_0) N(x, p), \tag{16}$$

where N is the neural network given by (14). Then, adapting the formulation of the optimization problem (8) to solve (15), we obtain:

$$\min_{p \in \mathcal{P}} \sum_{y_i \in \widehat{\Omega}} \left(\frac{\partial \Psi_0}{\partial x}(y_i, p) - f(y_i, \Psi_0(y_i, p)) \right)^2 \tag{17}$$

Now we introduce the first proposed formulation to solve the problem (15). In this case, the test function is defined as

$$\Psi_1(x, p) = N(x, p), \tag{18}$$

where the neural network N is given by (14). The adaptation of the constrained optimization problem (10) takes the form below:

$$\begin{cases} \min\limits_{p \in \mathcal{P}} \sum\limits_{y_i \in \widehat{\Omega}} \left(\dfrac{\partial \Psi_1}{\partial x}(y_i, p) - f(y_i, \Psi_1(y_i, p)) \right)^2 \\ \text{subject to } \Psi_1(x_0, p) = A_0 \end{cases} \tag{19}$$

Finally, by applying the second proposed formulation to solve the problem (15), we have that the test function is given by

$$\Psi_2(x, p_1, p_2) = N_1(x, p_1) + N_2(x, p_2), \tag{20}$$

where the neural networks N_1 and N_2 are defined in (14). For this case, the optimization problem (13) becomes:

$$\begin{cases} \min\limits_{(p_1,p_2) \in \mathcal{P}_1 \times \mathcal{P}_2} \sum\limits_{y_i \in \widehat{\Omega}} \left(\dfrac{\partial \Psi_2}{\partial x}(y_i, p_1) - f(y_i, \Psi_2(y_i, p_1)) \right)^2 \\ \text{subject to } \begin{cases} N_1(x_0, p_1) &= 0 \\ N_2(x_0, p_2) &= A_0 \end{cases} \end{cases} \tag{21}$$

3.2 Second order ordinary differential equations

Here we are going to deal with two problems involving second-order equations: one with both prescribed boundary conditions and the other with mixed boundary conditions.

3.2.1 Problem with Dirichlet Boundary Conditions

Consider the following second order boundary value problem containing only Dirichlet boundary conditions:

$$\begin{cases} \dfrac{d^2\Psi}{dx^2}(x) = f\left(x, \Psi(x), \dfrac{d\Psi}{dx}(x)\right); \quad x \in (x_0, x_1), \\ \Psi(x_0) = A_0, \quad \Psi(x_1) = A_1. \end{cases} \tag{22}$$

It is assumed that A_0 and A_1 are real numbers and the function f is defined in such a way that the problem (22) has a unique solution and that this solution has two continuous derivatives.

From the formulation proposed by Lagaris et al. [1], it follows that the test function for the problem (22) is given by:

$$\Psi_0(x, p) = (A_1 - A_0)\frac{x - x_0}{x_1 - x_0} + A_0 + (x - x_0)(x_1 - x)N(x, p), \tag{23}$$

where the neural network N is given by (14). In this case, the problem (8) takes the following form:

$$\min_{p \in \mathcal{P}} \sum_{y_i \in \widehat{\Omega}} \left[\frac{\partial^2 \Psi_0}{\partial x^2}(y_i, p) - f\left(y_i, \Psi_0(y_i, p), \frac{\partial \Psi_0}{\partial x}(y_i, p)\right)\right]^2 \tag{24}$$

By applying the formulations proposed in this work to solve the problem (22), we take the test functions Ψ_1 and Ψ_2, respectively, by (18) and (20). The optimization problems (10) and (13) are now given, respectively, by:

$$\begin{cases} \min\limits_{p \in \mathcal{P}} \sum\limits_{y_i \in \widehat{\Omega}} \left(\dfrac{\partial^2 \Psi_1}{\partial x^2}(y_i, p) - f\left(y_i, \Psi_1(y_i, p), \dfrac{\partial \Psi_1}{\partial x}(y_i, p)\right)\right)^2 \\ \text{subject to} \begin{cases} \Psi_1(x_0, p) &= A_0 \\ \Psi_1(x_1, p) &= A_1 \end{cases} \end{cases} \tag{25}$$

and

$$\begin{cases} \min\limits_{(p_1, p_2) \in \mathcal{P}_1 \times \mathcal{P}_2} \sum\limits_{y_i \in \widehat{\Omega}} \left(\dfrac{\partial^2 \Psi_2}{\partial x}(y_i, p_1, p_2) - f\left(y_i, \Psi_2(y_i, p_1, p_2), \dfrac{\partial \Psi_2}{\partial x}(y_i, p_1, p_2)\right)\right)^2 \\ \text{subject to} \begin{cases} N_1(x_0, p_1) &= 0 \\ N_1(x_1, p_1) &= 0 \\ N_2(x_0, p_2) &= A_0 \\ N_2(x_1, p_2) &= A_1 \end{cases} \end{cases} \tag{26}$$

3.2.2 Problem with mixed boundary conditions

Next, we will consider the following boundary value problem of second order containing mixed boundary conditions:

$$\begin{cases} \dfrac{d^2\Psi}{dx^2}(x) = f\left(x, \Psi(x), \dfrac{d\Psi}{dx}(x)\right); \quad x \in (x_0, x_1), \\ \Psi(x_0) = A_0, \quad \dfrac{d\Psi}{dx}(x_0) = A_1. \end{cases} \tag{27}$$

According to the formulation proposed by Lagaris et al. [1], the test function related to the problem (27) is defined as:

$$\Psi_0(x, p) = A_0 + A_1(x - x_0) + (x - x(0))^2 N(x, p), \tag{28}$$

where the neural network N is given by (14). In this case, the problem (8) has its formulation given by (24).

Following the same structure as the previous subsection, now we adapt the formulations proposed in this work to the problem (27). Again, we have that the test functions Ψ_1 and Ψ_2 are given, respectively, by (18) and (20). Finally, the optimization problems (10) and (13) are reduced, respectively, to:

$$\begin{cases} \min\limits_{p \in \mathcal{P}} \sum\limits_{y_i \in \widehat{\Omega}} \left(\dfrac{\partial^2 \Psi_1}{\partial x^2}(y_i, p) - f\left(y_i, \Psi_1(y_i, p), \dfrac{\partial \Psi_1}{\partial x}(y_i, p)\right)\right)^2 \\ \text{such that} \begin{cases} \Psi_1(x_0, p) &= A_0 \\ \dfrac{\partial \Psi_1}{\partial x}(x_0, p) &= A_1 \end{cases} \end{cases} \tag{29}$$

and

$$\begin{cases} \min\limits_{(p_1,p_2)\in\mathcal{P}_1\times\mathcal{P}_2} \sum\limits_{y_i\in\widehat{\Omega}} \left(\dfrac{\partial^2\Psi_2}{\partial x}(y_i,p_1,p_2) - f\left(y_i, \Psi_2(y_i,p_1,p_2), \dfrac{\partial\Psi_2}{\partial x}(y_i,p_1,p_2)\right)\right)^2 \\ \text{such that} \begin{cases} N_1(x_0,p_1) &= 0 \\ \dfrac{\partial N_1}{\partial x}(x_0,p_1) &= 0 \\ N_2(x_0,p_2) &= A_0 \\ \dfrac{\partial N_2}{\partial x}(x_0,p_2) &= A_1 \end{cases} \end{cases} \tag{30}$$

4 Numerical results

This section compares analytical solutions of some ordinary differential equations with numerical solutions of optimization problems defined in the previous section. The section is divided into three subsections. The first subsection details general aspects for all numerical examples. The second subsection presents examples of first-order ODEs and their solutions to the problems (17), (19) and (21). Finally, the third subsection includes second-order ODEs and their solutions to the problems (24), (25), (26), (29) and (30).

4.1 General comments on the numerical results

In order to obtain numerical solutions to the optimization problems, we used the Matlab routine *fmincon*, which is a general solver for nonlinear programming problems with equality and inequality constraints. Extensive user documentation for this solver can be found with the "doc fmincon" command in Matlab.

The *fmincon* algorithm generates a sequence of parameters $p^{(0)}, p^{(1)}, \ldots, p^{(m)}, \ldots$ convergent to a local minimum p^* of the optimization problem. The user can select the algorithm that *fmincon* will employ to generate this sequence of points. In this work, the "interior-point" and "sqp" algorithms were used. A wide bibliography can be found for these algorithms, for example in Byrd et al. [10] and Herskovits [11]. There was no significant difference in using one or the other algorithm to solve the optimization problems. The results obtained in this work employ the "sqp" algorithm and we will consider the starting point as the null vector: $p^{(0)} = 0$.

In practice the solver *fmincon* ends its execution by returning a point of the sequence $p^{(m)}$ that verifies a stop condition. When the stop condition happens, the solver terminates and returns a parameter called $exitflag$. We remark the following conditions that stops the execution of *fmincon*:

- $exitflag = 0$: this condition indicates that *fmincon* terminates prematurely because the number of iterations or the number of function evaluations exceeds a predefined maximum number. In this case the point $p^{(m)}$ is generally not a good solution to the optimization problem.
- $exitflag = 1$: in this case the solver ends because $p^{(m)}$ is considered a local minimum.
- $exitflag = 2$ or 3: indicates that $p^{(m)}$ is a possible local minimum. In this case the solver ends because, in two successive iterations, the value of the objective function value vary less than a tolerance or the optimization variables vary less than a tolerance.

In all examples, their domains has the form $\Omega = [x_0, x_1]$ and their discretization $\widehat{\Omega}$ are defined with n_x equidistant points in Ω. It is denoted by Ψ_{ana} the analytical solution, and by Ψ_{num} the numerical solution of the problems (17), (19), (21), (24), (25), (29) and (30), respectively. The numerical solution is defined as $\Psi_{num}(x) = \Psi_i(x, p^{(m)})$ with $i \in \{0, 1, 2\}$. The functions N, N_1 and N_2 are one-input neural networks, as defined in (14). The network N is the sum of H sigmoids, in this case it has $3H$ parameters gathered in the vector $p \in \mathbb{R}^{3H}$. Likewise, the networks N_1 and N_2 are defined with H_1 and H_2 sigmoid, respectively. Thus, $p_1 \in \mathbb{R}^{3H_1}$ and $p_2 \in \mathbb{R}^{3H_2}$.

The number of variables of the optimization problem (number of entries in the vector p) is denoted by n_p. For problems with ones neural network (17), (19), (24) and (25) we have $n_p = 3H$. For problems with two neural networks (21), (26) and (30), we have $n_p = 3H_1 + 3H_2$.

We say that a numerical solution Ψ_{num} is acceptable for $\varepsilon > 0$ if:

$$error = \max_{x\in\widehat{\Omega}} |\Psi_{ana}(x) - \Psi_{num}(x)| \leqslant \varepsilon \tag{31}$$

The numerical experiments show that a local minimum of the optimization problems, detected by *fmincon*, do not guarantee that the numerical solution Ψ_{num} is acceptable for $\varepsilon \approx 0$. In all examples, it was observed that the number of iterations m is greater than n_p. This usually leads to premature termination of *fmincon* with $exitflag = 0$. In this case the numerical solution obtained is such that $|\Psi_{ana}(x) - \Psi_{num}(x)|$ is not close to zero for some points $x \in \widehat{\Omega}$. In some other cases the premature termination of *fmincon* with $exitflag = 0$ happens because the maximum number of functions evaluations is exceeded.

4.2 Examples with first-order ODEs

4.2.1 Example 1

Consider the following initial value problem given by:

$$\begin{cases} \dfrac{d\Psi}{dx}(x) &= \alpha\Psi(x) - \beta e^{\alpha x} sin(\beta x); \quad x \in (0,1), \\ \Psi(0) &= 1 \end{cases} \tag{32}$$

where $\alpha = -2$ and $\beta = 10$. The exact solution of (32) is

$$\Psi_{ana}(x) = e^{\alpha x} cos(\beta x).$$

Fig. 1 shows the graph of the analytic solution Ψ_{ana} and the numerical solutions of problems (17), (19) and (21) . The solutions obtained with the problems (17) and (21) are acceptable, however the solution of (19) is not satisfactory.

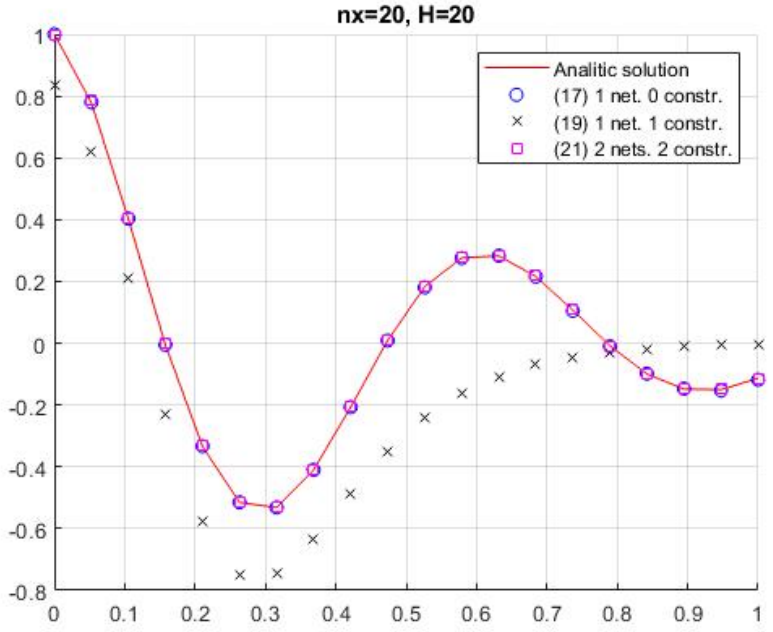

Figure 1: Numerical solutions for Example 1 with 20 points and 20 sigmoids.

Table 1 shows that *fmincon* terminates prematurely ($exitflag = 0$) by reaching the maximum number of function evaluations (6000 evaluations). The number of iterations and errors obtained with the formulation (17) and (21) is similar. It is also observed that *fmincon* cannot find an acceptable solution to the problem (19).

Table 1: Results for Example 1 with $n_x = 20$ and $H = 20$.

Problem	$exitflag$	iterations	$error$
(17)	0	95	4.4×10^{-3}
(19)	0	72	4.3×10^{-1}
(21)	0	93	8.6×10^{-3}

4.2.2 Example 2

In this example we want to solve the following ODE:

$$\begin{cases} \frac{d\Psi}{dx}(x) &= \Psi(x)\left[\beta\cos(\beta x) - \frac{1}{x-\gamma}\right] + \frac{\alpha}{x-\gamma}e^{sin(\beta x)}; \quad x \in (0,1), \\ \Psi(0) &= 0 \end{cases} \tag{33}$$

where $\alpha = -\frac{1}{3}$, $\beta = 4$ and $\gamma = \frac{3}{2}$. The exact solution of (33) is

$$\Psi_{ana}(x) = \frac{\alpha x}{x-\gamma}e^{sin(\beta x)}.$$

Fig. 2 illustrates that the test functions of problems (19) and (21), obtained with *fmincon*, are close to the analytic solution. Table 2 indicates that *fmincon* has reached a local minimum of the problem (17), however the obtained solution is not acceptable for $\varepsilon = 0.1$.

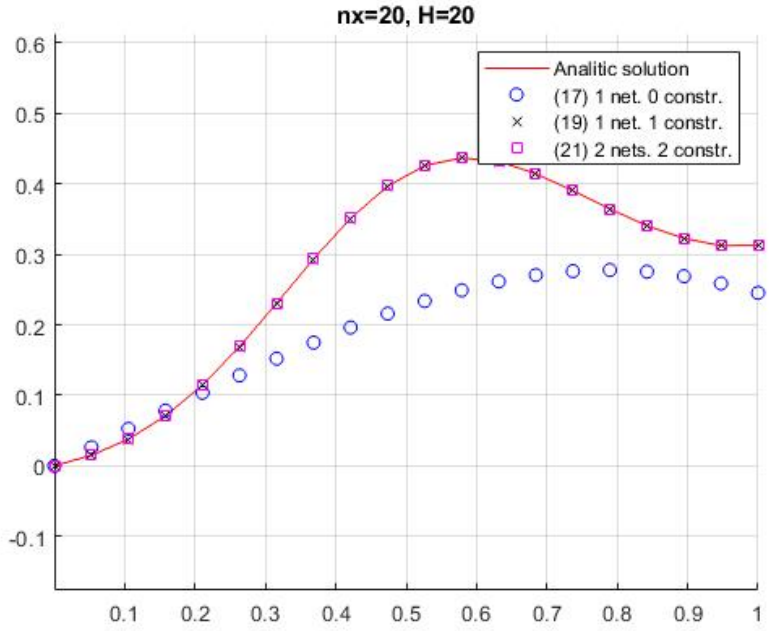

Figure 2: Numerical solutions for Example 2 with 20 points and 20 sigmoids.

Table 2: Results for Example 2 with $n_x = 20$ and $H = 20$.

Problem	$exitflag$	Iterations	$error$
(17)	1	30	1.9×10^{-1}
(19)	0	671	5.9×10^{-5}
(21)	0	768	1.3×10^{-4}

4.3 Examples with second-order ODEs

4.3.1 Example 3

In this case we want to solve the following ODE:

$$\begin{cases} \frac{d^2\Psi}{dx^2}(x) &= -\alpha\frac{d\Psi}{dx}(x) - \beta; \quad x \in (0,1), \\ \Psi(0) &= 0 \\ \Psi(1) &= 0 \end{cases} \tag{34}$$

where $\alpha = 2$ and $\beta = 10$. The analytical solution of (34) is

$$\Psi_{ana}(x) = \gamma e^{-\alpha x} - \frac{\beta}{\alpha}x + \delta, \text{ where } \gamma = \frac{\beta}{\alpha e^{-\alpha} - 1} \text{ and } \delta = -\gamma.$$

The differential equation (34) corresponds to the equation of motion for a body thrown upwards near the earth's surface, $\Psi(x)$ is the position of the body at the instant of time x. The left hand side of the differential equation in (34) is the (unitary) mass times the acceleration. The right hand side of (34) is the sum of applied external forces to the mass: a force opposite to the velocity $-\alpha\frac{d\Psi}{dx}(x)$ and the gravitational force $-\beta$. The mass is thrown from position 0, at initial time $x = 0$, and returned to the same position at final time $x = 1$. The graph of the analytic solution shown in Fig. 3 is not a parabola (the maximum height is before 0.5). The mass goes up with greater speed than when it goes down. This behavior is due to the coefficient of friction $\alpha > 0$.

Fig. 3 reveals that the numerical solutions to the optimization problems at (24), (25) and (26) are close to the exact solution. The number of iterations for each problem is shown in Table 3. In these example the solver converges to local minimum of problems (24) and (26). For problem (25) the solver stopped with $exitflag = 0$ because it exceeded the iteration limit. In this result we see that the error is acceptable but the optimality condition (first order KKT conditions) was not accomplished yet.

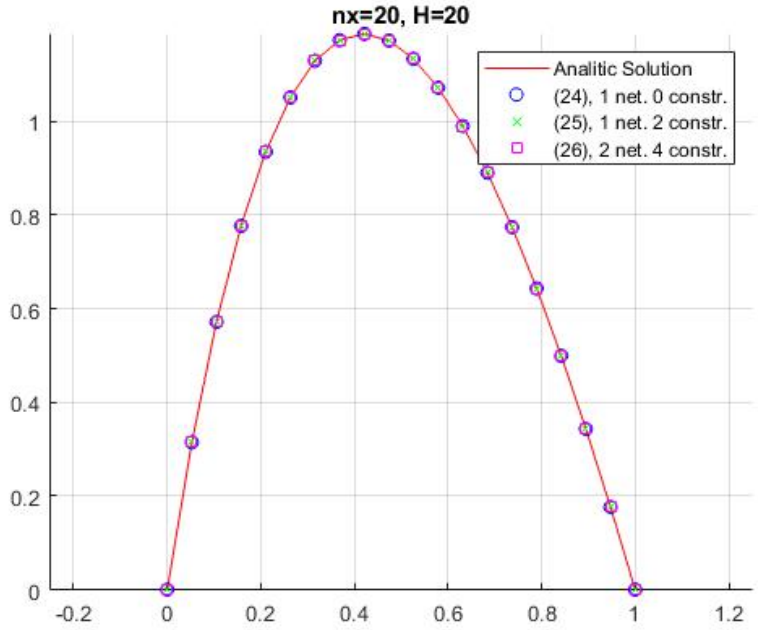

Figure 3: Numerical solutions for Example 3 with 20 points and 20 sigmoids.

Table 3: Results for Example 3 with $n_x = 20$ and $H = 20$.

Problem	$exitflag$	Iterations	$error$
(24)	1	246	2.4×10^{-7}
(25)	0	600	1.2×10^{-6}
(26)	1	259	6.2×10^{-7}

4.3.2 Example 4

In this example, we want to solve:

$$\begin{cases} \dfrac{d^2\Psi}{dx^2}(x) &= -\alpha\dfrac{d\Psi}{dx}(x) - \beta\Psi(x); \quad x \in (0,1), \\ \Psi(0) &= \dfrac{1}{2} \\ \dfrac{d\Psi}{dx}(0) &= 3 \end{cases} \tag{35}$$

where $\alpha = 6$ and $\beta = 8$. The analytical solution is

$$\Psi_{ana}(x) = \gamma e^{-\frac{1}{2}\left(\sqrt{\xi}+\alpha\right)x} + \delta e^{\frac{1}{2}\left(\sqrt{\xi}-\alpha\right)x}, \text{ where } \gamma = -\frac{\alpha-\sqrt{\xi}+8}{4\sqrt{\xi}}, \delta = \frac{\alpha+\sqrt{\xi}+8}{4\sqrt{\xi}} \text{ and } \xi = \alpha^2 - 4\beta.$$

The differential equation in (35) models the motion for a mass-spring-damper system, $\Psi(x)$ is the position of the mass at the instant of time x. In the left hand side of (35), $\frac{d^2\Psi}{dx^2}(x)$ is the (unitary) mass times the acceleration and the right hand side is the sum of the external forces: the friction, opposite the velocity $-\alpha\frac{d\Psi}{dx}(x)$, and the spring force $-\beta\Psi(x)$. At the initial instant $x = 0$, the mass starts from position $\frac{1}{2}$ and has a velocity of 3. The initial velocity will cause the mass to increase its position, but the spring force will cause the position to return asymptotically to the equilibrium position 0. This is the behavior of the analytic solution shown in Fig. 4.

Fig. 4 graphs the test functions found for this example and shows that the solutions of (24) and (29) are good approximations of the analytic solution. Table 4 reveals that the local minimum of their respective optimization problems are found in a similar number of iterations.

Table 4: Results for Example 4 with $n_x = 14$ and $H = 14$.

Problem	$exitflag$	Iterations	$error$
(24)	1	380	9.0×10^{-6}
(29)	1	383	2.0×10^{-5}
(30)	0	420	2.6×10^{-1}

5 Conclusions

In this work we introduce two alternative methodologies to solve initial and boundary value problems with neural networks. Optimization problems with constraints are defined to fit the parameters of one or two single layer neural networks. Numerical experiments were carried out with first and second order ODEs and the results show that they are comparable to those obtained in Lagaris et al. [1]. The number of iterations performed by *fmincon* to solve (17), (19) and (21) (as well as (24), (25) and (26)) are similar. Numerical results show that some local minimum of the formulated optimization problems are also solution of the initial and boundary values problems. However, as the

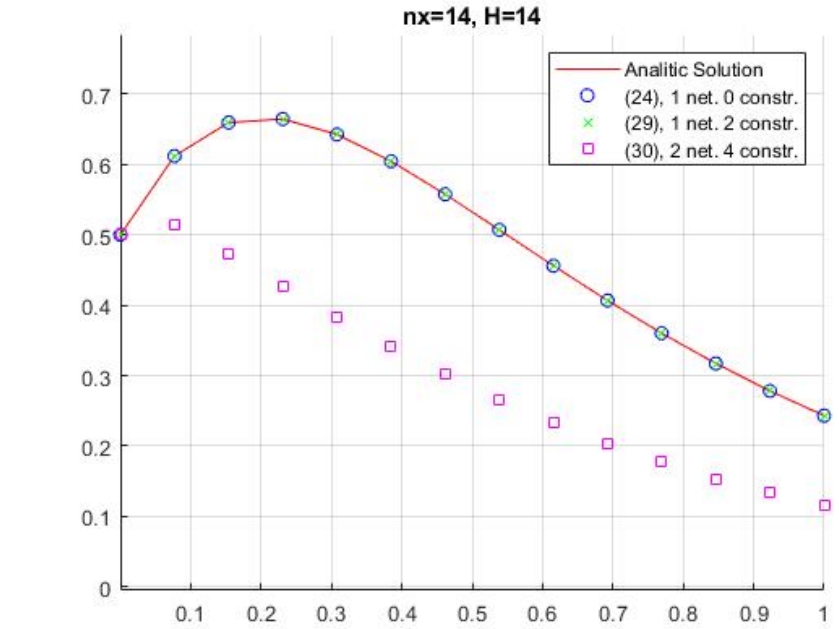

Figure 4: Numerical solutions for Example 4 with 14 points and 14 sigmoids.

numerical example 3 indicates, local minimum may not be a good solution for ODE. This leads to some questions. How to define optimization problems or techniques to avoid those local minima? Is it possible to formulate convex optimization problems to solve ODEs? If there exists convexity, the problem of finding local minimum is equivalent to find satisfactory ODE solutions.

An interesting question that can be addressed in future works concerns the domain discretization. Both the optimization problem associated with the methodology of Lagaris (Equation (8)) and the associated optimization problems to the proposed methodology (Equations (19) and (21)) are defined using a previous discretization of the domain Ω. How this discretization affects the numerical solutions? An alternative that can be implemented in future work is the use of the L^2 norm to measure the distance between the exact and the approximate solution. As this norm is given by an integral, this eliminates the discretization of Ω and numerical errors could be avoided.

In Lagaris et al. [1], unconstrained optimization problems were formulated to solve Partial Differential Equations (PDEs). We believe it is also possible to define equality constraints optimization problems and test functions to solve PDEs.

References

[1] I. Lagaris, A. Likas, and D. Fotiadis, "Artificial neural networks for solving ordinary and partial differential equations," *IEEE Transactions on Neural Networks*, vol. 9, no. 5, pp. 987–1000, 1998. Available at: https://doi.org/10.1109/72.712178

[2] C. Aggarwal, *Linear Algebra and Optimization for Machine Learning*, 1st ed. New York, USA: Springer, 2020.

[3] G. Strang, *Linear Algebra and Learning from Data*, 1st ed. Massachussets, USA: Cambridge Press, 2019.

[4] O. I. Abiodun, A. Jantan, A. E. Omolara, K. V. Dada, N. AbdElatif, and H. Arshad, "State-of-the-art in artificial neural network applications: A survey," *Heliyon*, vol. 4, no. 11, p. e00938, 2018. Available at: https://doi.org/10.1016/j.heliyon.2018.e00938

[5] G. Cybenko, "Approximations by superpositions of a sigmoidal function," *Mathematics of Control, Signals and Systems*, vol. 2, no. 4, pp. 303–314, 1989. Available at: https://doi.org/10.1007/BF02551274

[6] K. Hornik, M. Stinchcombe, and H. White, "Multi-layer feedforward networks are universal approximators," *Neural Networks*, vol. 2, no. 5, pp. 359–366, 1989. Available at: https://doi.org/10.1016/0893-6080(89)90020-8

[7] R. Khemchandani, A. Karpatne, and S. Chandra, "Twin support vector regression for the simultaneous learning of a function and its derivatives," *International Journal of Machine Learning and Cybernetics*, vol. 4, pp. 51–63, 2013. Available at: https://doi.org/10.1007/s13042-012-0072-1

[8] V. Avrutskiy, "Enhancing function approximation abilities of neural networks by training derivatives," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 32, no. 2, pp. 916–924, 2021. Available at: https://doi.org/10.1109/TNNLS.2020.2979706

[9] L. Evans, *Partial Differential Equations*, 2nd ed. Providence, USA: American Mathematical Society, 2010.

[10] R. Byrd, J. C. Gilbert, and J. Nocedal, "A trust region method based on interior point techniques for nonlinear programming," *Mathematical Programming*, vol. 89, no. 1, pp. 149–185, 2000. Available at: https://doi.org/10.1007/PL00011391

[11] J. Herskovits, "A view on nonlinear optimization," in *Advances in Structural Optimization: Solid Mechanics and Its Applications*, 1st ed. Dordrecht, Netherlands: Springer, 1995, vol. 25, pp. 71–116.