Machine Learning for Computational Science and Engineering (2025) 1:6
https://doi.org/10.1007/s44379-024-00007-7

RESEARCH

Augmented neural forms with parametric condition-matching operators for solving ordinary differential equations

Adam D. Kypriadis[1] · Isaac E. Lagaris[1] · Aristidis Likas[1] · Konstantinos E. Parsopoulos[1]

Received: 9 October 2024 / Accepted: 14 November 2024

Abstract
The approximation of solutions to ordinary and partial differential equations is an important task. Complementing the classical numerical analysis methods of solution, neural forms offer a valuable alternative approach. Neural forms, which are closed-form expressions involving neural networks, are specifically designed to exactly satisfy prescribed initial or boundary conditions. Starting from the important class of ordinary differential equations, the present work aims to refine and extend the methodology of neural forms, paving the way for its application to the highly challenging field of partial differential equations. First, a formalism is developed for the systematic construction of proper neural forms with parametric condition-matches, amenable to optimization. Second, a novel technique is described for converting Neumann or Robin conditions into equivalent parametric Dirichlet conditions. Third, a methodology is introduced for determining an upper bound on the absolute deviation from the exact solution. The proposed approach was applied on a set of diverse test problems, including first and second order ordinary differential equations, as well as first order systems. Stiff differential equations have been considered as well. The obtained solutions were evaluated against known exact solutions, solutions derived by the physics-informed neural networks method, and solutions obtained via a contemporary finite difference technique. The reported results demonstrate that the augmented neural forms provide closed-form solutions featuring high-quality interpolation and controllable overall accuracy. These attributes are essential for expanding the approach to treat challenging partial differential equation problems.

Keywords Ordinary differential equations · Neural networks · Neural forms · Condition matching

1 Introduction

Neural networks have been successfully employed for the solution of diverse problems in science and engineering. Their universal approximation capability, when using activation functions of specific type [1–3], renders them particularly suitable for modeling tasks. Within this framework, neural networks have been used to model solutions for ordinary differential equations (ODEs) and partial differential equations (PDEs) as well. Although the first relevant neural methods appeared in the mid-90s [4–9], their application remained relatively limited for a period of 20 years. This stagnation can be attributed to several factors, including the lack of widely accessible specialized software, and the limited computing power available at the time. The recent advancements in deep learning, the emergence of sophisticated software platforms (such as TensorFlow, Keras, and PyTorch) and the advent of high-performance computing technologies (including GPUs and parallel multicore multiprocessor clusters) have led to a resurgence of interest in solving PDEs through neural networks. Today, a number of alternative techniques based on a plethora of neural network types have been developed and are currently available [10–20].

The primary goal of neural methods is the construction of a parametric model that can be tuned to produce a solution with a prescribed degree of accuracy. This is carried out by minimizing a properly designed *error function* (also known as *loss* or *cost function*) within the space of the model parameters. In

✉ Konstantinos E. Parsopoulos
kostasp@uoi.gr

Adam D. Kypriadis
a.kypriadis@uoi.gr

Isaac E. Lagaris
lagaris@uoi.gr

Aristidis Likas
arly@uoi.gr

[1] Department of Computer Science and Engineering, University of Ioannina, GR-45110 Ioannina, Greece

Published online: 12 December 2024

addition, differential equations are accompanied by boundary and/or initial conditions that must be satisfied. From here on we will use the term *conditions* to refer either to boundary or initial conditions, unless otherwise specified. It is important to realize that a trial solution which fails to meet the prescribed conditions cannot be considered as an adequate approximation to the true solution of the problem under consideration.

A common approach treats the conditions as equality constraints when minimizing the error function. In this case, constrained optimization methods (interior point, active set, augmented Lagrangian, etc.) [21, 22] that exactly satisfy the constraints should be used. However, their application appears to be complicated and time-consuming as well. Alternatively, equality constraints may be approximately treated via penalty methods using unconstrained optimization. Penalty terms are added to the original error function in an attempt to inhibit constraint violations. In fact, this is by far the most commonly followed approach, codenamed physics-informed neural networks (PINNs) by Raissi et al. [16], perhaps due to reasons of implementation convenience and simplicity of interpretation. If the parametric trial solution is crafted in such a way so as to satisfy exactly the specified conditions regardless of the parameter values, the constraints are eliminated, and therefore the error function is simplified. The trial solution is then cast as a neural form [17], instead of a single neural network. In the case of differential equations defined inside rectangular domains, this approach is preferable because the exact satisfaction of the conditions allows the method to attain higher accuracy as pointed out in Müller and Zeinhofer [23]. In contrast, for PDEs within domains of complex geometry, the design of proper trial solutions that satisfy prescribed conditions is challenging, and thus the alternative approximate penalty approach has been frequently preferred.

In cases where the conditions are treated as constraints, the trial solution typically consists of a single neural network. On the other hand, the constraint elimination approach requires more elaborate functional constructs. To this end, *neural forms* were indirectly introduced in Lagaris et al. [8] and further developed in Lagari et al. [17]. In both cases, a rigid polynomial expression was used to account for the conditions. An alternative neural form type was introduced in McFall and Mahan [10], involving a function representing the distance of a domain point from the boundary, an idea that was further extended in Berg and Nyström [14] for the case of Dirichlet conditions. Further developments on this subject have recently appeared in [24, 25].

Neural forms that exactly satisfy the ODE conditions are obviously not unique. Therefore, the question concerning the existence of a possibly optimal choice naturally arises. This is one of the issues addressed here by introducing the *augmented neural forms* that are presented in the following paragraphs. The contributions of the present work can be summarized as follows:

1. A flexible, parametric neural form replaces the rigid polynomial boundary or initial match used in [8, 17].
2. A technique is introduced for transforming Neumann or Robin to equivalent parametric Dirichlet conditions.
3. A methodology is introduced for evaluating the quality of the approximate solution by determining a reliable upper bound of its absolute deviation from the exact solution.

The performance of the proposed approach was assessed on first and second order ODEs, as well as on first order systems, subject to Dirichlet, Neumann, mixed Dirichlet-Neumann, Robin, and Cauchy conditions. The obtained solutions were compared to existing exact solutions, to solutions obtained by the PINNs, and to solutions obtained by the `ode113` numerical solver of MATLAB® [26], which is based on a state-of-the-art variable-step variable-order Adams-Bashforth-Moulton method. The conducted analysis verified that the proposed augmented neural forms can provide robust, closed-form solutions with exact satisfaction of the prescribed conditions, along with excellent generalization performance.

The rest of the article is organized as follows: Section 2 briefly reviews neural ODE solvers, and analyzes the neural forms concept. Section 3 elaborates on the augmented neural forms framework for first and second order ODEs. Section 4 presents the transformation of Neumann or Robin conditions to equivalent parametric Dirichlet conditions. Section 5 provides a comprehensive analysis leading to a reliable upper bound for the absolute deviation between the approximate and the exact solution. Section 6 describes in detail the experimental setup. Section 7 presents the obtained solutions and a comparison thereof, as mentioned above. Finally, Section 8 provides an overview of the contributions of this work, along with comments, conclusions, and suggestions for further research.

2 Neural forms for solving ordinary differential equations

Consider the general form of a second order ODE:

$$\mathcal{L}_x \psi(x) = f(x), \quad x \in [a, b], \tag{1}$$

with boundary conditions (BCs):

$$\mathcal{B}_a \psi(x)|_a = \xi_a, \qquad \mathcal{B}_b \psi(x)|_b = \xi_b, \tag{2}$$

or with initial conditions (ICs):

$$\psi(x)|_a = \xi_0, \qquad \psi'(x)|_a = \xi_1, \tag{3}$$

where $\mathcal{L}_x$ is a second order differential operator, $\mathcal{B}_a$ and $\mathcal{B}_b$ are the corresponding boundary condition operators, and ξ's are prescribed constants. An appropriate neural model for a trial solution Ψ_T that approximates the exact solution $\psi(x)$ can be designed in many ways. The simplest and most common approach sets the trial solution equal to a neural network $N(x, \boldsymbol{w})$ with weights $\boldsymbol{w}$, i.e.:

$$\psi(x) \approx \Psi_T(x, \boldsymbol{w}) \triangleq N(x, \boldsymbol{w}), \tag{4}$$

and requires that both the ODE and the accompanying conditions are satisfied. This is accomplished by finding suitable values for the weights $\boldsymbol{w}$ so that:

$$\int_a^b dx\ [\mathcal{L}_x \Psi_T(x, \boldsymbol{w}) - f(x)]^2 = 0, \tag{5}$$

subject to either:

$$\mathcal{B}_a \Psi_T(x, \boldsymbol{w})|_a = \xi_a, \qquad \mathcal{B}_b \Psi_T(x, \boldsymbol{w})|_b = \xi_b, \qquad \text{(case with BCs)} \tag{6}$$

or:

$$\Psi_T(x, \boldsymbol{w})|_a = \xi_0, \qquad \Psi_T'(x, \boldsymbol{w})|_a = \xi_1. \qquad \text{(case with ICs)} \tag{7}$$

This problem is typically addressed by minimizing the left-hand side of Eq. (5) under the conditions in Eqs. (6) or (7), using constrained optimization methods. The *baseline neural method* [20], also known as PINNs [16], treats the constraints by adding penalties to the error function, which then assumes the following form:

$$E(\boldsymbol{w}) \triangleq \int_a^b dx\ [\mathcal{L}_x \Psi_T(x, \boldsymbol{w}) - f(x)]^2 + \zeta\ Pen(\boldsymbol{w}), \tag{8}$$

where $\zeta > 0$ is a positive penalty-regulating coefficient and:

$$Pen(\boldsymbol{w}) = \begin{cases} [(\mathcal{B}_a \Psi_T(x, \boldsymbol{w})|_a - \xi_a)^2 + (\mathcal{B}_b \Psi_T(x, \boldsymbol{w})|_b - \xi_b)^2], & \text{for BCs,} \\ [(\Psi_T(x, \boldsymbol{w})|_a - \xi_0)^2 + (\Psi_T'(x, \boldsymbol{w})|_a - \xi_1)^2], & \text{for ICs,} \end{cases} \tag{9}$$

and employs unconstrained optimization.

Alternatively, neural forms cast the trial solution as a sum of two terms:

$$\psi(x) \approx \Psi_T(x, \boldsymbol{w}) \triangleq A(x) + G(x, \boldsymbol{w}), \tag{10}$$

where $A(x)$ is a smooth function matching the boundary or initial conditions, henceforth referred to as a *boundary* or *initial match*, respectively. Moreover, $G(x, \boldsymbol{w})$ is a parametric function typically depending on a neural network with null contribution to the problem's conditions, i.e.:

$$\forall w, \quad \mathcal{B}_a G(x, \boldsymbol{w})|_a = 0 \ \text{ and } \ \mathcal{B}_b G(x, \boldsymbol{w})|_b = 0 \ \text{ for BCs,}$$
$$\text{or}$$
$$\forall w, \quad G(x, \boldsymbol{w})|_a = 0 \ \text{ and } \ G'(x, \boldsymbol{w})|_a = 0 \ \text{ for ICs.}$$

Note that, with the above formulation, the trial solution satisfies the prescribed ODE conditions by construction.

The functions $A(x)$ and $G(x, \boldsymbol{w})$ are closely related, and their design is based on a common matching operator. Let $\mathcal{P}_x$ be a boundary matching operator defined as:

$$\mathcal{P}_x \psi(x) \triangleq A(x).$$

We can easily verify that this operator is not unique. In fact, there is an infinite number of alternative forms. For instance, for the case of Dirichlet boundary conditions, $\mathcal{P}_x$ may be given by a linear relation as follows:

$$\mathcal{P}_x \psi(x) \triangleq \psi(a)\, \frac{x-b}{a-b} + \psi(b)\, \frac{x-a}{b-a}, \tag{11}$$

or as any of the nonlinear alternatives:

$$\mathcal{P}_x \psi(x) \triangleq \psi(a) \left(\frac{x-b}{a-b}\right)^n + \psi(b) \left(\frac{x-a}{b-a}\right)^n, \quad n = 2, 3, \ldots \tag{12}$$

Starting from the identity:

$$\psi(x) = \mathcal{P}_x \psi(x) + (1 - \mathcal{P}_x)\, \psi(x),$$

a plausible trial solution based on a neural network $N(x, \boldsymbol{w})$ may be cast as:

$$\Psi_T(x, \boldsymbol{w}) = \mathcal{P}_x \psi(x) + (1 - \mathcal{P}_x) N(x, \boldsymbol{w})$$

and, hence, the $G(x, \boldsymbol{w})$ function in Eq. (10) can be constructed as:

$$G(x, \boldsymbol{w}) = (1 - \mathcal{P}_x)\, N(x, \boldsymbol{w}).$$

Then, an approximate solution of the ODE is obtained by minimizing the error function:

$$E(\boldsymbol{w}) \triangleq \int_a^b dx\ [\mathcal{L}_x \Psi_T(x, \boldsymbol{w}) - f(x)]^2, \tag{13}$$

with respect to the neural network's weight vector $\boldsymbol{w}$.

3 Augmented neural forms for different types of boundary and initial conditions

As illustrated in Eqs. (11) and (12), the function $A(x)$ is not unique. Therefore, we can take a further step by representing it as a parametric function $A(x, \boldsymbol{\theta})$ that satisfies the problem's conditions for any value of the parameter vector $\boldsymbol{\theta}$. The error function of Eq. (13) can then be written as:

$$E(\boldsymbol{w}, \boldsymbol{\theta}) \triangleq \int dx \left[\mathcal{L}_x \Psi_T(x, \boldsymbol{w}, \boldsymbol{\theta}) - f(x)\right]^2, \tag{14}$$

and minimized with respect to both $\boldsymbol{w}$ and $\boldsymbol{\theta}$, leading to an optimized choice for the match $A(x, \boldsymbol{\theta})$ and, consequently, for the trial neural form solution. In the following paragraphs, we elaborate on this concept for the cases of first order ODEs and systems, as well as for second order ODEs.

3.1 Augmented neural forms for first order ordinary differential equations and systems

Let the first order ODE:

$$\mathcal{L}_x \psi(x) = f(x), \quad x \in [a, b], \tag{15}$$

with the initial condition $\psi(a) = \xi_a$. The functions $A(x, \boldsymbol{\theta})$, $G(x, \boldsymbol{w}, \boldsymbol{\theta})$, $\Psi_T(x, \boldsymbol{w}, \boldsymbol{\theta})$, are then given by:

$$\begin{aligned} A(x, \boldsymbol{\theta}) &= \mathcal{P}_x(\boldsymbol{\theta})\psi(x) \triangleq \psi(a)\left(F(x, \boldsymbol{\theta}) - F(a, \boldsymbol{\theta}) + 1\right), \\ G(x, \boldsymbol{w}, \boldsymbol{\theta}) &= (1 - \mathcal{P}_x(\boldsymbol{\theta}))N(x, \boldsymbol{w}), \\ \Psi_T(x, \boldsymbol{w}, \boldsymbol{\theta}) &= A(x, \boldsymbol{\theta}) + G(x, \boldsymbol{w}, \boldsymbol{\theta}), \end{aligned}$$

where $F(x, \boldsymbol{\theta})$ is a smooth, bounded function for $x \in [a, b]$. Note that $F(x, \boldsymbol{\theta})$ may be a neural network, i.e., $F(x, \boldsymbol{\theta}) = N_1(x, \boldsymbol{\theta})$. The associated error function is then given by:

$$E(\boldsymbol{w}, \boldsymbol{\theta}) \triangleq \int dx \left[\mathcal{L}_x \Psi_T(x, \boldsymbol{w}, \boldsymbol{\theta}) - f(x)\right]^2.$$

Generalizing the concepts above, let a system of first order ODEs:

$$\mathcal{L}\Psi(x) = f(x),$$

with $\Psi(x)$, $f(x) \in \mathbb{R}^n$, where:

$$\mathcal{L}\Psi(x) \triangleq \frac{d}{dx}\Psi(x) + F(x, \Psi),$$

and $F \in \mathbb{R}^n$. Then, the initial matches are defined as:

$$A_i(x, \boldsymbol{\theta_i}) = \mathcal{P}_i(\boldsymbol{\theta_i})\psi_i(x) = \psi_i(a)\left(\tilde{N}_i(x, \boldsymbol{\theta_i}) - \tilde{N}_i(a, \boldsymbol{\theta_i}) + 1\right), \quad i = 1, 2, \ldots, n,$$

and, respectively, we have:

$$G_i(x, \boldsymbol{\theta_i}, \boldsymbol{w_i}) = (1 - \mathcal{P}_i(\boldsymbol{\theta_i}))\, N_i(x, \boldsymbol{w_i}),$$

where $\tilde{N}_i(x, \boldsymbol{\theta_i})$ denotes the corresponding neural network for the i-th initial match, and $N_i(x, \boldsymbol{w_i})$ denotes the network for the G_i function associated with the i-th trial solution. Hence:

$$\Psi_T^{[i]}(x, \boldsymbol{w}, \boldsymbol{\theta}) = A_i(x, \boldsymbol{\theta_i}) + G_i(x, \boldsymbol{w_i}, \boldsymbol{\theta_i}), \quad \forall i = 1, 2, \ldots, n,$$

are the trial solutions for the above system of simultaneous equations.

3.2 Augmented neural forms for second order ordinary differential equations

Second order ODEs may be accompanied by Dirichlet, Neumann, mixed Dirichlet-Neumann, Robin, or Cauchy conditions. Each condition type corresponds to a different boundary matching operator, hence, we examine each case separately.

3.2.1 Case of Dirichlet conditions

Consider the following ODE:

$$\mathcal{L}_x \psi(x) = f(x), \quad x \in [a, b], \tag{16}$$

with Dirichlet boundary conditions:

$$\psi(a) = \xi_a, \qquad \psi(b) = \xi_b,$$

where $\mathcal{L}_x$ is a second order differential operator. Also, consider a function $A_\psi(x)$ defined for all $x \in [a, b]$, satisfying $A_\psi(a) = \psi(a)$ and $A_\psi(b) = \psi(b)$. Then, there are diverse ways to construct $A_\psi(x)$. For example, it may be crafted as:

$$\begin{aligned} A_\psi(x) = \psi(a)&\left(\frac{x-b}{a-b} + F(x) - F(a)\frac{x-b}{a-b} - F(b)\frac{x-a}{b-a}\right) \\ &+\psi(b)\left(\frac{x-a}{b-a} + H(x) - H(a)\frac{x-b}{a-b} - H(b)\frac{x-a}{b-a}\right), \end{aligned} \tag{17}$$

with $F(x)$ and $H(x)$ being smooth, bounded functions in x. Let us employ the form of Eq. (17) with:

$$F(x) = N_1(x, \boldsymbol{\theta_1}), \qquad H(x) = N_2(x, \boldsymbol{\theta_2}), \tag{18}$$

where $N_1(x, \boldsymbol{\theta_1})$ and $N_2(x, \boldsymbol{\theta_2})$ are neural networks with weight vectors $\boldsymbol{\theta_1}$ and $\boldsymbol{\theta_2}$, respectively. Then, the associated Dirichlet boundary matching operator $\mathcal{P}_x$ is given as:

$$\mathcal{P}_x(\boldsymbol{\theta_1}, \boldsymbol{\theta_2})\psi(x) \triangleq \psi(a)\left(\frac{x-b}{a-b} + N_1(x, \boldsymbol{\theta_1}) - N_1(a, \boldsymbol{\theta_1})\frac{x-b}{a-b}\right.$$

$$
\begin{aligned}
&-N_1(b,\boldsymbol{\theta_1})\frac{x-a}{b-a}\Big)\\
&+\psi(b)\left(\frac{x-a}{b-a}+N_2(x,\boldsymbol{\theta_2})-N_2(a,\boldsymbol{\theta_2})\frac{x-b}{a-b}\right.\\
&\left.-N_2(b,\boldsymbol{\theta_2})\frac{x-a}{b-a}\right).
\end{aligned}
$$

Thus, we have:

$$
G(x,\boldsymbol{\theta_1},\boldsymbol{\theta_2},\boldsymbol{w})=[1-\mathcal{P}_x(\boldsymbol{\theta_1},\boldsymbol{\theta_2})]\,N(x,\boldsymbol{w}), \tag{19}
$$

yielding the trial solution:

$$
\Psi_T(x,\boldsymbol{\theta_1},\boldsymbol{\theta_2},\boldsymbol{w})=\mathcal{P}_x(\boldsymbol{\theta_1},\boldsymbol{\theta_2})\,\psi(x)+G(x,\boldsymbol{\theta_1},\boldsymbol{\theta_2},\boldsymbol{w}), \tag{20}
$$

which is a proper neural form for the case of Dirichlet conditions.

3.2.2 Case of mixed Dirichlet-Neumann conditions

Let us now consider the ODE of Eq. (16) with mixed conditions:

$$
\psi(a)=\xi_a \ \text{(Dirichlet)}, \qquad \psi'(b)=\xi_b \ \text{(Neumann)}.
$$

A plausible boundary match is given as:

$$
\begin{aligned}
\mathcal{P}_x(\boldsymbol{\theta_1},\boldsymbol{\theta_2})\psi(x)\triangleq\ &\psi(a)\Big[1+N_1(x,\boldsymbol{\theta_1})-N_1(a,\boldsymbol{\theta_1})\\
&-(x-a)N_1'(b,\boldsymbol{\theta_1})\Big]\\
&+\psi'(b)\Big[(x-a)+N_2(x,\boldsymbol{\theta_2})\\
&-N_2(a,\boldsymbol{\theta_2})-(x-a)N_2'(b,\boldsymbol{\theta_2})\Big].
\end{aligned}
$$

The function $G(x,\boldsymbol{\theta_1},\boldsymbol{\theta_2},\boldsymbol{w})$ and the trial solution $\Psi_T(x,\boldsymbol{\theta_1},\boldsymbol{\theta_2},\boldsymbol{w})$ for this case are given as in Eqs. (19) and (20), respectively.

3.2.3 Case of Neumann conditions

Consider the ODE of Eq. (16) with Neumann conditions:

$$
\psi'(a)=\xi_a, \qquad \psi'(b)=\xi_b.
$$

The boundary matching operator assumes the following form:

$$
\mathcal{P}_x(\boldsymbol{\theta_1},\boldsymbol{\theta_2})\psi(x)\ \triangleq\ \psi'(a)\Lambda_1(x,\boldsymbol{\theta_1})+\psi'(b)\Lambda_2(x,\boldsymbol{\theta_2}),
$$

where:

$$
\begin{aligned}
\Lambda_1(x,\boldsymbol{\theta_1})=\ &\frac{(x-b)^2}{2(a-b)}+N_1(x,\boldsymbol{\theta_1})-N_1'(a,\boldsymbol{\theta_1})\frac{(x-b)^2}{2(a-b)}\\
&-N_1'(b,\boldsymbol{\theta_1})\frac{(x-a)^2}{2(b-a)},
\end{aligned}
$$

and:

$$
\begin{aligned}
\Lambda_2(x,\boldsymbol{\theta_2})=\ &\frac{(x-a)^2}{2(b-a)}+N_2(x,\boldsymbol{\theta_2})-N_2'(a,\boldsymbol{\theta_2})\frac{(x-b)^2}{2(a-b)}\\
&-N_2'(b,\boldsymbol{\theta_2})\frac{(x-a)^2}{2(b-a)}.
\end{aligned}
$$

Again, the functions $G(x,\boldsymbol{\theta_1},\boldsymbol{\theta_2},\boldsymbol{w})$, $\Psi_T(x,\boldsymbol{\theta_1},\boldsymbol{\theta_2},\boldsymbol{w})$, are given by Eqs. (19) and (20), respectively.

3.2.4 Case of Cauchy conditions

Cauchy conditions are initial conditions of the form:

$$
\psi(a)=\xi_0, \qquad \psi'(a)=\xi_1.
$$

The following initial matching operator is admissible:

$$
\begin{aligned}
\mathcal{P}_x(\boldsymbol{\theta_1},\boldsymbol{\theta_2})\psi(x)\triangleq\ &\psi(a)\Big[1+N_1(x,\boldsymbol{\theta_1})-N_1(a,\boldsymbol{\theta_1})\\
&-(x-a)N_1'(a,\boldsymbol{\theta_1})\Big]\\
&+\psi'(a)\Big[(x-a)+N_2(x,\boldsymbol{\theta_2})\\
&-N_2(a,\boldsymbol{\theta_2})-(x-a)N_2'(a,\boldsymbol{\theta_2})\Big].
\end{aligned}
$$

Equations (19) and (20) hold in this case too.

3.2.5 Case of Robin conditions

Robin conditions are defined as linear combinations of the form:

$$
\lambda\,\psi(a)+\mu\,\psi'(a)=\xi_a, \qquad \gamma\,\psi(b)+\delta\,\psi'(b)=\xi_b. \tag{21}
$$

The boundary matching operator in this case is defined as:

$$
\begin{aligned}
\mathcal{P}_x(\boldsymbol{\theta_1},\boldsymbol{\theta_2})\psi(x)\triangleq\ &\big[\lambda\,\psi(a)+\mu\,\psi'(a)\big]\,F(x,\boldsymbol{\theta_1})\\
&+\big[\gamma\,\psi(b)+\delta\,\psi'(b)\big]\,H(x,\boldsymbol{\theta_2}),
\end{aligned}
$$

where:

$$
F(x,\boldsymbol{\theta_1})=N_1(x,\boldsymbol{\theta_1})+F_1(\boldsymbol{\theta_1})(2x-a-b)+F_2(\boldsymbol{\theta_1})(x-a)(x-b),
$$

with:

$$F_1(\boldsymbol{\theta_1}) = \frac{\delta\left[1-\lambda N_1(a,\boldsymbol{\theta_1})-\mu N_1'(a,\boldsymbol{\theta_1})\right]-\mu\left[\gamma N_1(b,\boldsymbol{\theta_1})+\delta N_1'(b,\boldsymbol{\theta_1})\right]}{\delta\left[2\mu-\lambda(b-a)\right]+\mu\left[2\delta+\gamma(b-a)\right]},$$

$$F_2(\boldsymbol{\theta_1}) = \frac{-\gamma N_1(b,\boldsymbol{\theta_1})-\delta N_1'(b,\boldsymbol{\theta_1})-\left[2\delta+\gamma(b-a)\right]F_1(\boldsymbol{\theta_1})}{\delta(b-a)},$$

and:

$$H(x,\boldsymbol{\theta_2}) = N_2(x,\boldsymbol{\theta_2}) + H_1(\boldsymbol{\theta_2})(2x-a-b) + H_2(\boldsymbol{\theta_2})(x-a)(x-b),$$

with:

$$H_1(\boldsymbol{\theta_2}) = \frac{\delta\left[-\lambda N_2(a,\boldsymbol{\theta_2})-\mu N_2'(a,\boldsymbol{\theta_2})\right]+\mu\left[1-\gamma N_2(b,\boldsymbol{\theta_2})-\delta N_2'(b,\boldsymbol{\theta_2})\right]}{\delta\left[2\mu-\lambda(b-a)\right]+\mu\left[2\delta+\gamma(b-a)\right]},$$

$$H_2(\boldsymbol{\theta_2}) = \frac{1-\gamma N_2(b,\boldsymbol{\theta_2})-\delta N_2'(b,\boldsymbol{\theta_2})-\left[2\delta+\gamma(b-a)\right]H_1(\boldsymbol{\theta_2})}{\delta(b-a)},$$

while Eqs. (19) and (20) provide the trial solution.

4 Reductive transformation of Neumann conditions to equivalent Dirichlet boundary conditions

Instead of maintaining different boundary matching operators for all condition types, neural forms for Dirichlet conditions of the form $\psi(a) = \xi_a$, $\psi(b) = \xi_b$, can be solely considered. In this case, a mixed problem with conditions $\psi'(a) = \xi_a$ and $\psi(b) = \xi_b$ can be tackled by replacing $\psi(a)$ with a suitable expression, such that the Dirichlet-type neural form satisfies the mixed conditions. Let us demonstrate this idea with a simple example, using the following non-parametric (rigid) boundary matching operator:

$$\mathcal{P}_x\psi(x) = \psi(a)\left(\frac{x-b}{b-a}\right)^2 + \psi(b)\left(\frac{x-a}{b-a}\right)^2. \tag{22}$$

The trial solution is given as:

$$\Psi_T(x,\boldsymbol{\theta}) = N(x,\boldsymbol{\theta}) + [\psi(a)-N(a,\boldsymbol{\theta})]\left(\frac{x-b}{b-a}\right)^2 + [\psi(b)-N(b,\boldsymbol{\theta})]\left(\frac{x-a}{b-a}\right)^2, \tag{23}$$

and its first derivative with respect to x is:

$$\Psi_T'(x,\boldsymbol{\theta}) = N'(x,\boldsymbol{\theta}) + 2\,[\psi(a)-N(a,\boldsymbol{\theta})]\,\frac{x-b}{(b-a)^2} + 2\,[\psi(b)-N(b,\boldsymbol{\theta})]\,\frac{x-a}{(b-a)^2}.$$

Demanding that $\Psi_T'(a,\boldsymbol{\theta}) = \psi'(a)$ and solving for $\psi(a)$ yields:

$$\psi(a) = N(a,\boldsymbol{\theta}) + \frac{b-a}{2}\left[N'(a,\boldsymbol{\theta}) - \psi'(a)\right].$$

Substituting $\psi(a)$ in Eq. (23) results in the neural form that satisfies the mixed boundary conditions. We provide the following result:

$$\psi(a) = \frac{(b-a)\left[\xi_a - \mu\,N'(a,\boldsymbol{\theta})\right] - 2\,\mu\,N(a,\boldsymbol{\theta})}{(b-a)\,\lambda - 2\,\mu},$$

$$\psi(b) = \frac{(b-a)\left[\xi_b - \delta\,N'(b,\boldsymbol{\theta})\right] + 2\,\gamma N(b,\boldsymbol{\theta})}{(b-a)\,\gamma + 2\,\delta}.$$

for the case where the Robin conditions of Eq. (21) are prescribed.

5 Goodness of the approximate solution

Assessing the quality of an approximate solution is of fundamental importance. Relevant research outcomes have recently been reported in Liu et al. [27]. In the following paragraphs, we present a perturbation approach that can yield, under certain weak assumptions, a reliable upper bound for the absolute deviation $|\Psi_T(x)-\psi(x)|$ of the obtained solution from the exact one.

More specifically, let:

$$\Psi_T(x) = \mathcal{P}_x(\boldsymbol{\theta}^*)\psi(x) + (1-\mathcal{P}_x(\boldsymbol{\theta}^*))\,N(x,\boldsymbol{w}^*),$$

be the trial solution obtained by an augmented neural form, where $\boldsymbol{\theta}^*$ and $\boldsymbol{w}^*$ are the final values of $\boldsymbol{\theta}$ and $\boldsymbol{w}$ after training. The corresponding error is positive and may be written as $E(\Psi_T) = s^2$, where s is a scalar. We now construct a new trial solution by adding a perturbation $\eta(x,\boldsymbol{\gamma})$ to the obtained trial solution $\Psi_T(x)$, where $\boldsymbol{\gamma}$ is the perturbation's parameter vector. In order to preserve the satisfaction of the prescribed conditions, the perturbation should be of the form:

$$\eta(x,\boldsymbol{\gamma}) = (1-\mathcal{P}_x(\boldsymbol{\theta}^*))\,N_s(x,\boldsymbol{\gamma}),$$

with $N_s(x,\boldsymbol{\gamma})$ being a neural network. The perturbed trial solution, $\Psi_T(x)+\eta(x,\boldsymbol{\gamma})$, produces the error function $E(\Psi_T+\eta)$ that is minimized with respect to the perturbation parameters $\boldsymbol{\gamma}$, starting from an initial vector $\boldsymbol{\gamma}^{(0)}$ that satisfies $\eta\left(x,\boldsymbol{\gamma}^{(0)}\right) = 0$. The minimized error value, corresponding to a final vector $\boldsymbol{\gamma}^*$, is $E(\Psi_T+\eta) \triangleq \delta^2 < s^2$.

Consider a second order ODE of the form $\psi''(x) - f(x,\psi,\psi') = 0$. The perturbed error value is then given

as:

$$E(\Psi_T + \eta) = \int \Big[\Psi_T''(x) + \eta''(x, \boldsymbol{\gamma}) - f(x, \Psi_T + \eta, \Psi_T' + \eta')\Big]^2 dx. \tag{24}$$

Assuming that $\eta(x, \boldsymbol{\gamma})$ is a small perturbation, we may consider the first order approximation:

$$f(x, \Psi_T + \eta, \Psi_T' + \eta') \approx f(x, \Psi_T, \Psi_T') + \eta \frac{\partial f}{\partial \Psi_T} + \eta' \frac{\partial f}{\partial \Psi_T'}. \tag{25}$$

Substituting in Eq. (24), we derive that:

$$\delta^2 = s^2 + 2 \int (\Psi_T'' - f) \left(\eta'' - \eta \frac{\partial f}{\partial \Psi_T} - \eta' \frac{\partial f}{\partial \Psi_T'}\right) dx. \tag{26}$$

If there exists an η_{ex} such that $E(\Psi_T + \eta_{ex}) = 0$, then:

$$s^2 + 2 \int (\Psi_T'' - f) \left[\eta_{ex}'' - \eta_{ex} \frac{\partial f}{\partial \Psi_T} - \eta_{ex}' \frac{\partial f}{\partial \Psi_T'}\right] dx = 0. \tag{27}$$

Combining Eqs. (26) and (27) yields:

$$|\eta_{ex}(x)| = \frac{|\eta(x, \boldsymbol{\gamma}^*)|}{1 - \frac{\delta^2}{s^2}}, \tag{28}$$

which provides a reliable estimate for an upper bound of the deviation $|\Psi_T(x) - \psi(x)|$ as long as the assumed linear approximation in Eq. (25) is valid. In turn, this implies that $\eta(x, \boldsymbol{\gamma})$ must be a small perturbative correction.

6 Experimental setting

The following section provides a comprehensive presentation of the experimental assessment procedure of the proposed augmented neural forms. More specifically, it includes detailed information on the selected test problems, dataset generation schemes, neural architecture, and performance measures.

6.1 Test problems

The test problems included a number of first order and second order ODEs, as well as first order ODE systems, accompanied by boundary and initial conditions of various types that are frequently encountered in real world problems.

1. *Test Problem 1* (first order ODE)
This is a stiff ODE [28] defined as:

$$\begin{aligned} \psi'(x) &= -50\,(\psi(x) - \cos(x)), \quad x \in [0, 1.5], \\ \psi(0) &= 0.15, \end{aligned}$$

with exact solution:

$$\psi(x) = \left(0.15 - \frac{2500}{2501}\right) \exp(-50\,x) + \frac{50}{2501} \sin(x) + \frac{2500}{2501} \cos(x).$$

2. *Test Problem 2* (second order nonlinear ODE)
This problem is defined as:

$$\psi''(x) - \psi(x)\,\psi'(x) - \psi^3(x) = 0, \quad x \in [0, 9.5],$$

with exact solution:

$$\psi(x) = (10 - x)^{-1}.$$

In our experiments, it has been considered as a boundary value problem with Dirichlet, Neumann, mixed, and Robin conditions, and also as an initial value problem with Cauchy conditions.
3. *Test Problem 3* (second order ODE)
This is Bessel's equation subject to Cauchy initial conditions:

$$x^2\,\psi''(x) + x\,\psi'(x) + (x^2 - \nu^2)\,\psi(x) = 0, \quad x \in [0, 10], \tag{29}$$

$$\psi(0) = 1, \qquad \psi'(0) = 0,$$

The exact solution is Bessel's function of first kind, $J_\nu(x)$, which is non-singular at the origin. Here, the case $\nu = 0$ was considered.
4. *Test Problem 4* (first order system)
This is a nonlinear system of two equations defined as Lagaris et al. [8]:

$$\begin{aligned} \psi_1'(x) &= \cos(x) + \psi_1^2(x) + \psi_2(x) - \Big[1 + x^2 + \sin^2(x)\Big], \\ \psi_2'(x) &= 2x - \left(1 + x^2\right) \sin(x) + \psi_1(x)\psi_2(x), \quad x \in [0,3], \\ \psi_1(0) &= 0, \quad \psi_2(0) = 1, \end{aligned} \tag{30}$$

with exact solution:

$$\psi_1(x) = \sin(x), \quad \psi_2(x) = 1 + x^2.$$

The trial solution is adapted for the fist order system as:

$$\begin{aligned}\Psi_{T_1}(x, \boldsymbol{w_1}, \boldsymbol{\theta_1}) &= N_1(x, \boldsymbol{w_1}) + \left[1 + \tilde{N}_1(x, \boldsymbol{\theta_1}) - \tilde{N}_1(0, \boldsymbol{\theta_1})\right] \\ &\quad [\psi_1(0) - N_1(0, \boldsymbol{w_1})], \\ \Psi_{T_2}(x, \boldsymbol{w_2}, \boldsymbol{\theta_2}) &= N_2(x, \boldsymbol{w_2}) + \left[1 + \tilde{N}_2(x, \boldsymbol{\theta_2}) - \tilde{N}_2(0, \boldsymbol{\theta_2})\right] \\ &\quad [\psi_2(0) - N_2(0, \boldsymbol{w_2})],\end{aligned}$$

where N_1, $\tilde{N}_1$, and N_2, $\tilde{N}_2$, are the neural networks involved in the trial solutions Ψ_{T_1} and Ψ_{T_2}, respectively.

5. *Test Problem 5* (first order system)
This is a stiff system defined as Hairer and Wanner [28]:

$$\begin{aligned}\psi_1'(x) &= -10\,\psi_1(x) + 6\,\psi_2(x), \\ \psi_2'(x) &= 13.5\,\psi_1(x) - 10\,\psi_2(x),\end{aligned} \qquad x \in [0, 5], \quad (31)$$

$$\psi_1(0) = \frac{4}{3}\exp(1.0), \qquad \psi_2(0) = 0,$$

with exact solution:

$$\begin{aligned}\psi_1(x) &= \frac{2}{3}\exp(1.0)\,(\exp(-x) + \exp(-19x))\,, \\ \psi_2(x) &= \exp(1.0)\,(\exp(-x) - \exp(-19x))\,.\end{aligned}$$

6. *Test Problem 6* (reduction of second order ODE to first order system)
This is a nonlinear system derived by reducing the second order ODE of Test Problem 2 as follows:

$$\begin{aligned}\psi_1'(x) &= \psi_2(x), \\ \psi_2'(x) &= \psi_1(x)\,\psi_2(x) + \psi_1^3(x),\end{aligned} \qquad x \in [0, 9.5], \quad (32)$$

This problem was considered under various types of boundary conditions, namely Dirichlet, Neumann, mixed Dirichlet-Neumann, as well as with Cauchy initial conditions.

The exact solutions for all test problems are depicted in Fig. 1.

6.2 Dataset generation

For each test problem, two sets of data points were generated, namely a *training dataset* of size M_{tr} that was used to train the neural forms, and a *test dataset* of size M_{ts} that was used to assess the generalization quality. For the training dataset generation, two grid types were considered with data points $x_i \in [a, b]$:

(1) *Equidistant points*:

$$x_i = a + (i-1)\,\frac{b-a}{M_{\text{tr}} - 1}, \qquad i = 1, 2, \ldots, M_{\text{tr}}.$$

(2) *Chebyshev points*:

$$x_i = \frac{a+b}{2} + \frac{b-a}{2}\,\cos\left(\frac{(2\,i-1)\,\pi}{2\,M_{tr}}\right), \quad i = 1, 2, \ldots, M_{tr}.$$

The two grid types offer different point distributions inside the interval of interest. Chebyshev grids get denser towards the end points and are known to suppress the Runge phenomenon. Gauss-Legendre grids were tested as well, offering almost identical results to those of the Chebyshev grid. Hence, they were omitted from our study.

The size of the training set is always problem dependent and, thus, regularly determined through preliminary trial-and-error experimentation. As expected, more complex problems require higher number of training points. Table 1 reports the corresponding numbers in our experiments. For each test problem, three levels (low, medium, and high) were considered for M_{tr}. On the other hand, the test set consisted of $M_{\text{ts}} = 1000$ equidistant points in all cases.

6.3 Neural network architecture

In all test problems, a feedforward neural network with one hidden layer and sigmoid nodes was employed. Such a network admits a scalar $x \in \mathbb{R}$ as input, and produces a scalar output as follows:

$$N(x, \boldsymbol{\theta}) = \sum_{i=1}^{K} a_i\,\sigma(w_i x + b_i), \quad (33)$$

where K is the number of neurons, $\sigma(z) = [1 + \exp(-z)]^{-1}$ is the sigmoid activation function, and $\boldsymbol{\theta}$ is the network's parameter vector of size $|\boldsymbol{\theta}|$. For a network of K neurons, we have $|\boldsymbol{\theta}| = 3K$ parameters:

$$\boldsymbol{\theta} = \Big(\underbrace{a_1, w_1, b_1}_{\text{neuron 1}}, \underbrace{a_2, w_2, b_2}_{\text{neuron 2}}, \ldots, \underbrace{a_K, w_K, b_K}_{\text{neuron K}}\Big)$$

Our implementation was coded in the `Fortran` programming language, while the training was performed by the `Merlin` optimization platform [29, 30], which offers a variety of robust and efficient minimization routines. The minimization algorithms that were used in our experiments were (*i*) a pattern search method based on alternating variables, (*ii*) the irregular simplex method of Nelder and Mead [31], (*iii*) Powell's version of BFGS (TOLMIN) [32] with Goldfarb-Idnani factors and weak Wolfe conditions for the line search, and (*iv*) a trust-region version of BFGS based on Powell's dogleg technique [33]. All experiments were performed on the `PRECIOUS` high performance computing infrastructure [34], which consists of 12 computing nodes,

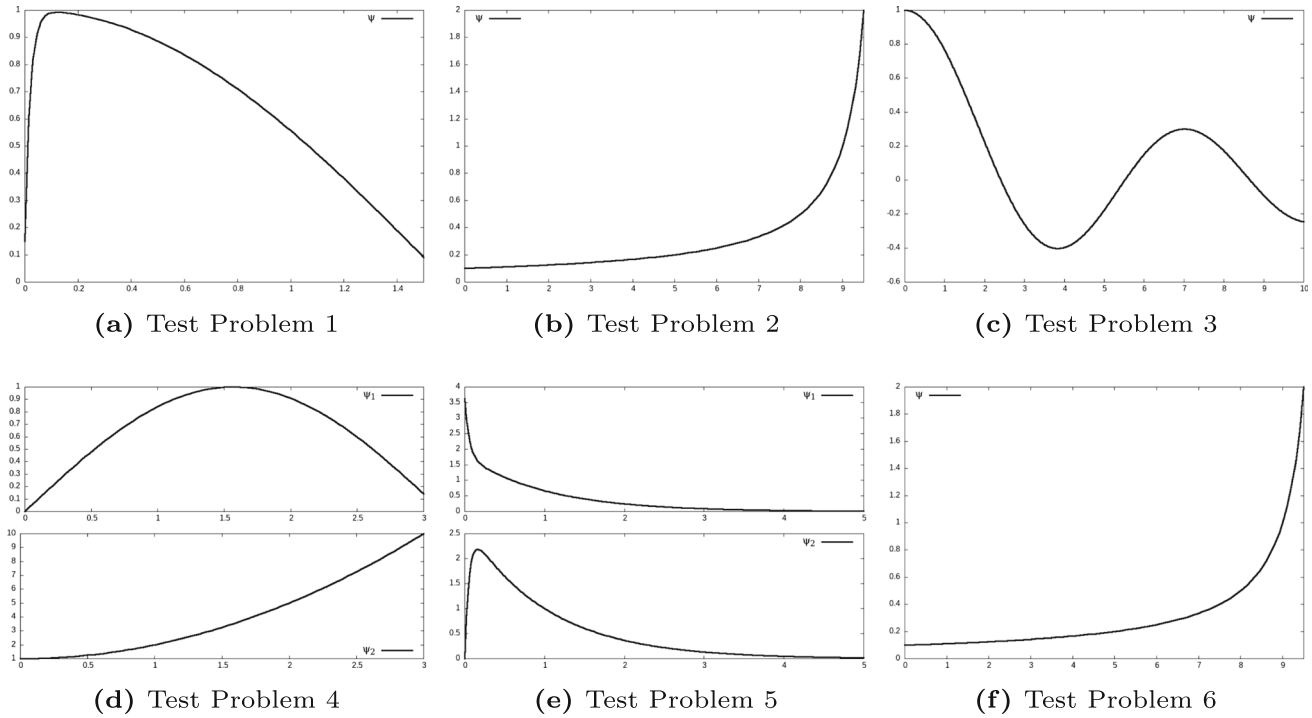

Fig. 1 Plots of the exact solutions for all test problems

each of which utilizes two Intel(R) Xeon(R) Gold 5220R processors with 48 computing cores. The number of neural form parameters used in our experiments are reported in Table 1.

6.4 Quality criteria for approximate solutions

An ideal criterion for the quality of an approximate solution is its difference from the exact solution. Given that, in previously unmet problems, the exact solution is not available, an alternative quality criterion can be the size of the ODE residuals. However, this measure may not exhibit the same behavior as the solutions' difference. In our experiments, all the selected test problems had known exact solutions, hence allowing accurate comparisons and evaluation of our approximate solutions.

Table 1 Number of training points (M_{tr}), test points (M_{ts}), and neural form parameters ($|\boldsymbol{\theta}|$) per test problem

Problem	low	medium	high	(M_{ts})	low	medium	high
1	40	80	160	1000	36	72	144
2	90	180	270	1000	90	180	270
3	90	180	270	1000	90	180	270
4	70	130	250	1000	60	120	240
5	70	130	250	1000	60	120	240
6	180	270	360	1000	180	270	360

Let M denote the number of data points over which the performance is evaluated (i.e., $M = M_{\text{tr}}$ for training, and $M = M_{\text{ts}}$ for testing), $\psi(x_i)$ be the exact solution at x_i, and $\Psi_T(x_i, \boldsymbol{\theta})$ be the trial solution at x_i, where $\boldsymbol{\theta}$ refers collectively to the neural form parameters. Then, the employed performance measures were as follows:

(1) *Measures based on the residuals*

$$\text{MSE} = \frac{1}{M}\sum_{i=1}^{M}[\mathcal{L}_x\Psi_T(x_i,\boldsymbol{\theta}) - f(x_i)]^2 \quad \text{(Mean squared error)} \tag{34}$$

$$\text{MXE} = \max_i \left\{[\mathcal{L}_x\Psi_T(x_i,\boldsymbol{\theta}) - f(x_i)]^2\right\} \quad \text{(Maximum squared error)} \tag{35}$$

(2) *Measures based on estimated deviation*

$$\text{MSED} = \frac{1}{M}\sum_{i=1}^{M}[\eta_{ex}(x_i)]^2 \quad \text{(Mean squared estimated deviation)} \tag{36}$$

$$\text{MXED} = \max_i \left\{[\eta_{ex}(x_i)]^2\right\} \quad \text{(Maximum squared estimated deviation)} \tag{37}$$

(3) *Measures based on exact deviation*

$$\text{MSD} = \frac{1}{M}\sum_{i=1}^{M}[\Psi_T(x_i,\boldsymbol{\theta}) - \psi(x_i)]^2 \quad \text{(Mean squared deviation)} \tag{38}$$

$$\text{MXD} = \max_i \left\{[\Psi_T(x_i,\boldsymbol{\theta}) - \psi(x_i)]^2\right\} \quad \text{(Maximum squared deviation)} \tag{39}$$

Note that measures based on ODE residuals indicate precision over the ODE's satisfaction, while measures based on the estimated and exact deviation indicate the distance from the exact solution $\psi(x)$.

7 Experimental results

Our experimental setting was designed to assess the performance of the proposed methodology across a diverse range of problems and computational scenarios that are commonly encountered in practice. More specifically, both equidistant grids and Chebyshev grids were used to generate training points. Three levels (low, medium, and high) were considered for the grid-point density as well as for the network size (total number of parameters) as reported in Table 1. In order to prevent overtraining and its subsequent degradation in generalization quality, scenarios where the number of network parameters exceeded the number of grid points were excluded. Thus, a total number of 12 distinct scenarios were ultimately considered.

For each training scenario and test problem, 100 independent experiments were conducted and the obtained solutions were assessed according to the quality measures presented in Section 6.4. A computational budget of 220000 function evaluations proved to be sufficient for training in all test problems. Note that Test Problem 2 was solved also using the reductive transformation of Section 4, without any significant difference. The results of the proposed neural forms were compared with those of the following methods:

1. The PINNs method where the trial solution is a standalone neural network and penalty terms are added to the error function, in order to account for the prescribed conditions according to Eqs. (8) and (9) with $\zeta = 1$.
2. The `ode113` numerical solver of MATLAB® [26], which is a variable-step variable-order Adams-Bashforth-Moulton PECE solver of orders 1 to 13. In this case, the boundary value problems were handled via shooting, while the performance over the test datasets was computed using spline interpolation.

Due to the large volume of results, Tables 2 and 3 present a condensed selection that includes the mean and maximum squared deviations from the exact solutions for the best neural form architecture, the `ode113` solver, and the top-performing PINN. The detailed results for each individual test problem are reported in the Supplementary material of the present article (see Sections 1 and 2 of the Supplementary material file).

The reported results clearly demonstrate that the proposed augmented neural forms offered high-quality solutions for all test problems. As expected, the `ode113` solver achieved superior precision at the points of its adaptive grid. The grids used by `ode113` are different from the training grids of the neural forms. Also, the testing grid, which is quite denser, contains points that are not contained in the training grids. Obtaining the solution at the testing grid points for the `ode113` case requires the use of an interpolation scheme. While, for the neural forms one has only to evaluate the obtained closed-form solution at these points.

For the `ode113` case, we considered three alternative interpolation schemes, namely cubic, quintic and septic splines, which use polynomials of 3-rd, 5-th, and 7-th degree, respectively. In our experiments the cubic and septic schemes yielded inferior approximation compared to neural forms, while the quintic yielded marginally superior results as can be verified by inspecting Tables 2 and 3. However, it should be underlined that one does not *a priori* know which interpolation scheme would be optimal on a given problem. On the other hand, neural forms offer a reliable choice without uncertainty, as they are interpolants themselves and hence they can be differentiated, integrated, or otherwise processed in a straightforward manner. This is an important feature that all traditional numerical solvers lack.

A comparison of the augmented neural forms against PINNs does not yield a conclusive indication of superiority for either, as can be readily deduced by inspecting Tables 2 and 3. More specifically, the reported MSD values over the test data suggest that each method achieves better performance for approximately half of the test problems. Further evidence is reported in Table 4, which reports the basic descriptive statistics, namely the minimum, maximum, mean, median, and standard deviation of the MSD values over 100 independent experiments for each test problem, for both the neural forms and the PINNs method. The last column reports p-values of Wilcoxon rank-sum tests performed between the MSD samples of the compared approaches for each test problem. The small p-values obtained in all but one case indicate statistically significant differences between the compared methods for significance levels 0.05 and 0.01. This suggests that, although a dominant method may exist for each distinct test problem, the count over all test problems is almost equally distributed between the two methods.

Further expanding our analysis, we conducted paired (signed-rank) Wilcoxon tests for each one of reported MSD descriptive statistics in Table 4. More specifically, each MSD column of Table 4 provides the paired samples of the two neural methods regarding the corresponding statistic. For example, the "min" column offers a sample of 15 values for the neural form, and 15 paired values for the PINN. For these samples, the basic statistical measures were calculated and they are reported in Table 5. For example, the maximum among the minimum MSD values achieved by the neural form for all test problems was 0.25e-18, while for the basic neural method it was 0.60e-08. The corresponding

Table 2 Mean (MSD) and maximum (MXD) squared deviation for the best neural form (NF), the best `ode113` results, and the best physics-informed neural network (PINN) for Test Problems 1-4. The corresponding grid type, number of neural form parameters $|\boldsymbol{\theta}|$, and number of training points M_{tr} are also reported

Problem	Conditions	Method	$\lvert\boldsymbol{\theta}\rvert$	M_{tr}	Sd	Grid		Training dataset MSD	Training dataset MXD	Test dataset MSD	Test dataset MXD
1	D	NF	144	160		Ch		0.22e-17	0.36e-16	0.25e-18	0.36e-16
		`ode113`		300	3	Ad		0.58e-28	0.18e-26	0.12e-16	0.76e-14
					5					0.12e-23	0.99e-21
					7					0.15e-13	0.15e-10
		PINN	36	80		Ch		0.16e-17	0.35e-16	0.25e-18	0.36e-16
2	D-D	NF	90	270		Ch		0.83e-23	0.98e-22	0.12e-22	0.98e-22
		PINN	90	180		Ch		0.90e-23	0.68e-22	0.13e-22	0.69e-22
	D-N	NF	90	90		Ch		0.17e-21	0.14e-20	0.24e-21	0.14e-20
		PINN	90	180		Ch		0.37e-22	0.33e-21	0.49e-22	0.33e-21
	N-N	NF	90	270		Ch		0.61e-23	0.25e-22	0.65e-23	0.26e-22
		PINN	180	180		Ch		0.19e-22	0.15e-21	0.27e-22	0.15e-21
	C	NF	90	180		Ch		0.43e-22	0.58e-21	0.13e-22	0.58e-21
		PINN	90	270		Ch		0.75e-20	0.10e-18	0.19e-20	0.10e-18
	R	NF	180	270		Ch		0.40e-23	0.37-22	0.59e-23	0.37e-22
		PINN	180	270		Ch		0.20e-22	0.17e-21	0.29e-22	0.17e-21
3	C	NF	180	180		Eq		0.45e-18	0.21e-17	0.45e-18	0.21e-17
		`ode113`		215	3	Ad		0.31e-29	0.28e-28	0.14e-15	0.18e-14
					5					0.12e-16	0.73e-14
					7					0.70e+18	0.33e+21
		PINN	180	270		Eq		0.38e-18	0.18e-17	0.38e-18	0.18e-17
4	D	NF	240	250		Ch	ψ_1	0.15e-18	0.12e-17	0.61e-19	0.12e-17
							ψ_2	0.12e-17	0.95e-17	0.43e-18	0.95e-17
		`ode113`		109	3	Ad	ψ_1	0.20e-24	0.51e-23	0.56e-16	0.27e-15
							ψ_2	0.14e-23	0.41e-22	0.19e-23	0.41e-22
					5		ψ_1			0.91e-24	0.63e-22
							ψ_2			0.20e-23	0.75e-22
					7		ψ_1			0.10e-08	0.54e-06
							ψ_2			0.80e-06	0.42e-03
		PINN	240	250		Ch	ψ_1	0.91e-07	0.69e-06	0.36e-07	0.70e-06
							ψ_2	0.68e-06	0.57e-05	0.26e-06	0.57e-05

Abbreviations: D=Dirichlet, N=Neumann, D-N=Mixed Dirichlet-Neumann, R=Robin
C=Cauchy, Ch=Chebyshev, Eq=Equidistant, Ad=Adaptive, Sd=Splines polynomial degree

paired samples of the two neural methods for each statistic were compared using the Wilcoxon paired (signed-rank) test, and the obtained p-values are reported in the last column of Table 5. For example, consider the paired samples of the minimum MSD, where each pair consists of the minimum MSD obtained from neural forms and the corresponding minimum MSD of the PINN. Since the calculated p-value is 0.96e-1, there is no statistically significant difference in significance level 0.05 or 0.01 between the two methods.

The p-values reported in Table 5 for all metrics suggest the lack of significant differences. Therefore, although it appears that for each specific problem it is highly probable to have a winner method, there is no statistically verified dominance of one method over the other across all the test problems. Thus, the proposed neural forms provide solutions of comparable quality to the PINNs method, while also maintaining the critical advantage of exact matching the initial/boundary conditions, not only approximately as PINNs. Note that Chebyshev training grids were shown to successfully treat the Runge phenomenon in many cases, thereby enhancing the performance of neural forms with respect to the solution deviation metrics.

Table 3 Mean (MSD) and maximum (MXD) squared deviation for the best neural form (NF), the best `ode113` results, and the best physics-informed neural network (PINN) for Test Problems 5 and 6. The corresponding grid type, number of neural form parameters $|\boldsymbol{\theta}|$, and number of training points M_{tr} are also reported

Problem	Conditions	Method	$\lvert\boldsymbol{\theta}\rvert$	M_{tr}	Sd	Grid		Training dataset MSD	Training dataset MXD	Test dataset MSD	Test dataset MXD
5	D	NF	240	250		Ch	ψ_1	0.30e-21	0.17e-20	0.42e-21	0.17e-20
							ψ_2	0.29e-21	0.18e-20	0.36e-21	0.18e-20
		`ode113`		321	3	Ad	ψ_1	0.52e-29	0.86e-28	0.97e-17	0.14e-14
							ψ_2	0.16e-28	0.31e-27	0.22e-16	0.31e-14
					5		ψ_1			0.72e-24	0.37e-21
							ψ_2			0.16e-23	0.84e-21
					7		ψ_1			0.75e-09	0.59e-06
							ψ_2			0.16e-11	0.13e-08
		PINN	240	250		Ch	ψ_1	0.13e-21	0.95e-21	0.17e-21	0.95e-21
							ψ_2	0.48e-21	0.30e-20	0.67e-21	0.30e-20
6	D-D	NF	270	360		Ch		0.20e-22	0.10e-21	0.27e-22	0.10e-21
		`ode113`		203	3	Ad		0.13e-26	0.56e-26	0.11e-16	0.34e-14
					5					0.63e-22	0.52e-19
					7					0.35e-16	0.27e-13
		PINN	180	180		Ch		0.30e-22	0.21e-21	0.41e-22	0.21e-21
	D-N	NF	360	360		Ch		0.13e-20	0.95e-20	0.18e-20	0.96e-20
		`ode113`		203	3	Ad		0.11e-26	0.46e-26	0.11e-16	0.34e-14
					5					0.63e-22	0.52e-19
					7					0.35e-16	0.26e-13
		Neural network	360	360		Ch		0.20e-21	0.13e-20	0.29e-21	0.13e-20
	N-N	NF	360	360		Ch		0.63e-23	0.56e-22	0.82e-23	0.57e-22
		`ode113`		203	3	Ad		0.42e-27	0.14e-26	0.11e-16	0.34e-14
					5					0.63e-22	0.52e-19
					7					0.52e-16	0.40e-13
		PINN	270	270		Ch		0.13e-21	0.94e-21	0.17e-21	0.94e-21
	C	NF	270	360		Ch		0.24e-19	0.34e-18	0.62e-20	0.34e-18
		`ode113`		203	3	Ad		0.27e-24	0.42e-23	0.11e-16	0.34e-14
					5					0.64e-22	0.53e-19
					7					0.41e-16	0.31e-13
		PINN	360	360		Ch		0.72e-19	0.99e-18	0.18e-19	0.99e-18

Abbreviations: D=Dirichlet, N=Neumann, D-N=Mixed Dirichlet-Neumann, R=Robin
C=Cauchy, Ch=Chebyshev, Eq=Equidistant, Ad=Adaptive, Sd=Splines polynomial degree

Finally, it is worth noting that widely used solvers such as the stochastic gradient descent, Adam [35], and variations [36], which have recently gained increasing popularity for neural network training in big data applications, may not achieve good performance. This can be verified in Section 3 (see Table 27) of the Supplementary material of the present article, which reports the mean and maximum squared deviation from the exact solutions for the best neural form trained using `Merlin` as described in Section 6.3, and the same neural form trained with Adam. Comparing the reported deviations, it becomes evident that for all test problems Adam either converged to solutions of remarkably lower quality or completely failed to obtain a valid solution.

7.1 Deviation bounding

To demonstrate the proposed deviation bounding technique presented in Section 5, we applied it on Test Problems 1 and 2 for various condition types. Table 6 reports the obtained deviation statistics. In Test Problem 1, the neural form had 36 network parameters, augmented by 18 additional parameters for $\eta(x, \gamma)$, while the training dataset consisted of 80

Table 4 Descriptive statistics of the mean squared deviations over the test datasets for the best neural form (NF) and the best PINN over 100 experiments per test problem and condition type. The corresponding grid type, number of neural form parameters $|\boldsymbol{\theta}|$, and number of training points M_{tr} are also reported. The provided p-values refer to Wilcoxon rank-sum tests between the two approaches

Test Problem	Condition Type	Method	Training Parameters $\lvert\theta\rvert$	M_{tr}	Grid		Mean Squared Deviation (MSD) min	max	mean	median	std	Wilcoxon p-value
1	D	NF	144	160	Ch		0.25e-18	0.30e-15	0.53e-17	0.49e-18	0.31e-16	0.30e-05
		PINN	36	80	Ch		0.25e-18	0.38e-10	0.47e-12	0.23e-17	0.39e-11	
2	D-D	NF	90	270	Ch		0.12e-22	0.30e-13	0.46e-15	0.13e-17	0.31e-14	0.23e-05
		PINN	90	180	Ch		0.13e-22	0.12e-15	0.37e-17	0.38e-19	0.16e-16	
	D-N	NF	90	90	Ch		0.24e-21	0.16e-11	0.28e-13	0.18e-17	0.17e-12	0.19e-13
		PINN	90	180	Ch		0.49e-22	0.37e-16	0.10e-17	0.60e-19	0.50e-17	
	N-N	NF	90	270	Ch		0.65e-23	0.10e-11	0.18e-13	0.45e-19	0.13e-12	0.20e-03
		PINN	180	180	Ch		0.27e-22	0.72e-17	0.16e-18	0.10e-19	0.75e-18	
	C	NF	90	180	Ch		0.13e-22	0.70e-09	0.79e-11	0.58e-16	0.70e-10	0.76e-09
		PINN	90	270	Ch		0.29e-22	0.22e-12	0.46e-14	0.12e-17	0.30e-13	
	R	NF	180	270	Ch		0.45e-23	0.41e-19	0.13e-20	0.24e-21	0.47e-20	0.31e-17
		PINN	180	270	Ch		0.29e-22	0.53e-17	0.11e-18	0.28e-20	0.70e-18	
3	C	NF	180	180	Eq		0.21e-19	0.17e-06	0.45e-08	0.28e-15	0.24e-07	0.12e-09
		PINN	180	270	Eq		0.24e-19	0.24e-09	0.27e-11	0.19e-16	0.24e-10	
4	D	NF	240	250	Ch	ψ_1	0.16e-20	0.30e-13	0.65e-14	0.69e-17	0.40e-13	0.25e-33
		PINN	240	250	Ch	ψ_1	0.85e-09	0.34e-06	0.86e-07	0.64e-07	0.68e-07	
		NF	240	250	Ch	ψ_2	0.11e-19	0.21e-11	0.46e-13	0.48e-16	0.28e-12	0.25e-33
		PINN	240	250	Ch	ψ_2	0.60e-08	0.24e-05	0.60e-06	0.45e-06	0.47e-06	
5	D	NF	240	250	Ch	ψ_1	0.42e-21	0.39e-17	0.13e-18	0.13e-19	0.46e-18	0.39e-12
		PINN	240	250	Ch	ψ_1	0.17e-21	0.26e-17	0.58e-19	0.19e-20	0.35e-18	
		NF	240	250	Ch	ψ_2	0.36e-21	0.57e-17	0.24e-19	0.28e-19	0.76e-18	0.91e-15
		PINN	240	250	Ch	ψ_2	0.67e-21	0.29e-12	0.19e-14	0.24e-20	0.29e-13	
6	D-D	NF	270	360	Ch		0.27e-22	0.19e-13	0.31e-15	0.38e-19	0.21e-14	0.34e+00
		PINN	180	180	Ch		0.41e-22	0.27e-05	0.27e-07	0.25e-19	0.27e-06	
	D-N	NF	360	360	Ch		0.18e-20	0.34e-15	0.71e-17	0.66e-18	0.35e-16	0.44e-23
		PINN	360	360	Ch		0.21e-21	0.62e-17	0.13e-18	0.67e-20	0.74e-18	
	N-N	NF	360	360	Ch		0.82e-23	0.58e-06	0.58e-08	0.16e-18	0.58e-07	0.92e-04
		PINN	270	270	Ch		0.17e-21	0.73e-17	0.23e-18	0.31e-19	0.93e-18	
	C	NF	270	360	Ch		0.62e-20	0.88e-10	0.21e-11	0.34e-14	0.11e-10	0.26e-25
		PINN	360	360	Ch		0.71e-20	0.95e-16	0.64e-17	0.14e-17	0.14e-16	

Abbreviations: D=Dirichlet, N=Neumann, D-N=Mixed Dirichlet-Neumann, R=Robin, C=Cauchy, Ch=Chebyshev, Eq=Equidistant

equidistant points in [0, 1.5]. In Test Problem 2, the neural form had 180 network parameters, augmented by 60 additional parameters for $\eta(x, \gamma)$, while the training dataset consisted of 270 equidistant points in [0, 9.5]. In both cases, the reported deviation metrics were calculated over 1000 equidistant points taken in [0, 1.5] for Test Problem 1, and in [0, 9.5] for Test Problem 2. For the latter case, Fig. 2 illustrates the absolute deviation and the corresponding estimated upper bound across the ODE's domain.

In terms of CPU time, both the augmented neural forms and the PINNs method were demanding. However, the PINNs method required slightly more processing time due to the extra effort needed for adjusting the penalty coefficient. The `ode113` solver proved to be more efficient, as expected for methods based on finite differences. However, since finite difference methods do not scale well with increasing dimensions, augmented neural forms offer promising alternatives for solving PDEs.

Table 5 Statistical measures calculated over the MSD columns of Table 4 for neural forms (NF) and PINNs. The reported p-values refer to Wilcoxon paired (signed-rank) tests for the two approaches

MSD metric	Method	min	max	mean	median	std	p-value
Min	NF	0.45e-23	0.25e-18	0.20e-19	0.36e-21	0.62e-19	0.96e-01
	PINN	0.13e-22	0.60e-08	0.46e-09	0.17e-21	0.15e-08	
Max	NF	0.39e-17	0.58e-06	0.50e-07	0.10e-11	0.15e-06	0.68e+00
	PINN	0.26e-17	0.27e-05	0.36e-06	0.12e-15	0.86e-06	
Mean	NF	0.13e-21	0.58e-08	0.69e-09	0.65e-14	0.18e-08	0.89e+00
	PINN	0.58e-19	0.46e-06	0.38e-07	0.64e-17	0.11e-06	
Median	NF	0.24-21	0.34e-14	0.25e-15	0.66e-18	0.84e-15	0.28e+00
	PINN	0.19e-20	0.45e-06	0.34e-07	0.38e-19	0.11e-06	
St.D.	NF	0.47e-20	0.58e-07	0.55e-08	0.40e-13	0.15e-07	0.89e+00
	PINN	0.35e-18	0.47e-06	0.54e-07	0.16e-16	0.13e-06	

8 Conclusions

Augmented neural forms offer a reliable approach for solving ODEs under a variety of boundary and initial conditions. The present work contributes to the existing literature by introducing a systematic procedure for designing appropriate trial solutions with neural forms that exactly satisfy the prescribed problem conditions. Furthermore, a novel technique is presented to transform a problem with Neumann or Robin boundary conditions into one with parametric Dirichlet conditions. This transformation provides an alternative methodology for formulating proper trial solutions using neural forms. Moreover, an upper bound for the absolute deviation of the obtained solution from the exact one was introduced, providing a new evaluation metric for the quality of approximation. For initial value problems, higher accuracy may be achieved by employing domain decomposition techniques.

The proposed augmented neural forms were evaluated on test problems consisting of first and second order ODEs, as well as first order ODE systems, under a variety of boundary and initial conditions. The obtained solutions were compared to the exact ones, as well as to solutions obtained by PINNs, and to solutions provided by state-of-the-art numerical solvers such as `ode113` of MATLAB®. The comparison clearly demonstrated that the augmented neural forms can provide accurate solutions with superior generalization performance. Taking also into consideration the provided closed form solutions and the exact satisfaction of the boundary/initial conditions, we can infer that the proposed augmented neural forms constitute a robust and reliable alternative for solving ODE's, while also laying the ground for the treatment of PDEs.

Moreover, our analysis indicates that careful selection of the training optimizer may offer significant advantages. Commonly utilized methods crafted for big data applications, such as stochastic gradient descent, Adam, and variants, may not perform well. Instead, pattern search and quasi-Newton methods were found to be very effective. In addition, we have noticed that Chebyshev grids performed better than equidistant grids for the majority of the considered test problems. This advantage stems from their capability to mitigate the Runge phenomenon which is related to overtraining of the neural networks.

Overall, the proposed approach is easy to implement and offers a differentiable closed-form solution that can be directly employed in subsequent calculations. Deep or shallow neural networks with any activation function meeting the universal approximation requirements may be used. The proposed ideas have the potential to provide a solid

Table 6 Mean (MSD) and maximum (MXD) squared deviation from the exact solution, as well as mean (MSED) and maximum (MXED) estimated deviation for Test Problems 1 and 2 with diverse condition types

Test Problem	Condition type	Deviation metrics			
		MSD	MXD	MSED	MXED
1	Dirichlet	0.19e-16	0.30e-16	0.34e-13	0.63e-13
2	Dirichlet	0.11e-20	0.24e-19	0.62e-15	0.26e-14
	Dirichlet-Neumann	0.68e-20	0.29e-18	0.20e-14	0.36e-14
	Neumann	0.86e-19	0.36e-17	0.14e-11	0.81e-11
	Cauchy	0.32e-19	0.18e-17	0.16e-16	0.43e-16
	Robin	0.55e-23	0.28e-22	0.49e-18	0.15e-17

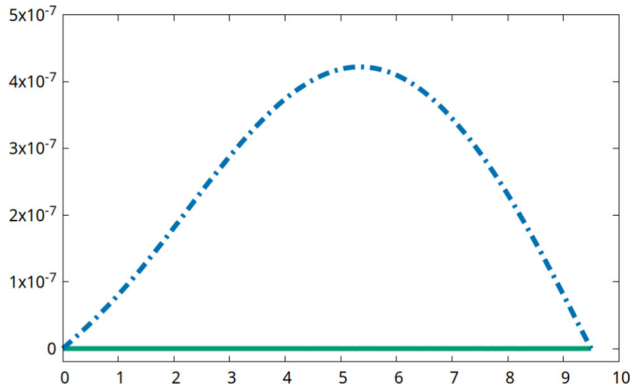

Fig. 2 Absolute deviation $|\Psi_T(x) - \psi(x)|$ (solid line), and the estimated absolute upper bound $|\eta_{ex}(x)|$ (dashed line) for Test Problem 2 with Dirichlet conditions

foundation for future initiatives. It might be of interest to consider the recently introduced functionally weighted neural network [13, 18], which is a special type of feedforward neural network with remarkable extrapolation capabilities. Furthermore, the proposed ideas could be extended in a straightforward manner to treat PDEs in rectangular domains. The challenging problem of PDEs defined inside non-rectangular domains certainly merits attention and requires further investigation.

Supplementary Information The online version contains supplementary material available at https://doi.org/10.1007/s44379-024-00007-7.

Author Contributions Authors Adam D. Kypriadis, Isaac E. Lagaris, Aristidis Likas, and Konstantinos E. Parsopoulos contributed equally to the conceptualization, methodology, validation, formal analysis, and writing of the present work. Additionally, the software development was carried out by Adam D. Kypriadis and Isaac E. Lagaris. All authors have read and approved the final manuscript.

Funding The authors did not receive any funding for this study.

Data Availability All data utilized for this study are available upon request.

Declarations

Competing interests The authors have no competing interests to declare.

References

1. Cybenko G. Approximation by superpositions of a sigmoidal function. Math Control Signals Syst. 1989;2(4):303–14.
2. Hornik K, Stinchcombe M, White H. Multilayer feedforward networks are universal approximators. Neural Netw. 1989;2(5):359–66.
3. Park J, Sandberg IW. Universal approximation using radial-basis-function networks. Neural Comput. 1991;3(2):246–57.
4. Lee H, Kang IS. Neural algorithm for solving differential equations. J Comput Phys. 1990;91(1):110–31.
5. Meade AJ, Fernandez AA. The numerical solution of linear ordinary differential equations by feedforward neural networks. Math Comput Model. 1994;19(12):1–25.
6. Dissanayake GMWM, Phan-Thien N. Neural-network-based approximations for solving partial differential equations. Commun Numer Methods Eng. 1994;10(3):195–201.
7. Lagaris IE, Likas A, Fotiadis DI. Artificial neural network methods in quantum mechanics. Comput Phys Commun. 1997;104:1–14.
8. Lagaris IE, Likas A, Fotiadis DI. Artificial neural networks for solving ordinary and partial differential equations. IEEE Trans Neural Networks. 1998;9:987–1000.
9. Lagaris IE, Likas A, Papageorgiou DG. Neural network methods for boundary value problems with irregular boundaries. IEEE Trans Neural Networks. 2000;11:1041–9.
10. McFall KS, Mahan JR. Artificial neural network method for solution of boundary value problems with exact satisfaction of arbitrary boundary conditions. Trans Neur Netw. 2009;20(8):1221–33.
11. Mall S, Chakraverty S. Application of Legendre neural network for solving ordinary differential equations. Appl Soft Comput. 2016;43:347–56.
12. Mall S, Chakraverty S. Single layer Chebychev neural network model for solving elliptic differential equations. Neural Process Lett. 2017;45:825–40.
13. Blekas KD, Lagaris IE. Artificial neural networks with an infinite number of nodes. J Phys: Conf Ser. 2017;915: 012006.
14. Berg J, Nyström K. A unified deep artificial neural network approach to partial differential equations in complex geometries. Neurocomputing. 2018;317:28–41.
15. Sirignano J, Spiliopoulos K. DGM: A deep learning algorithm for solving partial differential equations. J Comput Phys. 2018;375:1339–64.
16. Raissi M, Perdikaris P, Karniadakis GE. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. J Comput Phys. 2019;378:686–707.
17. Lagari PL, Tsoukalas LH, Safarkhani S, Lagaris IE. Systematic construction of neural forms for solving partial differential equations inside rectangular domains, subject to initial, boundary and interface conditions. Int J Artif Intell Tools. 2020;29:1–12.
18. Blekas KD, Lagaris IE. Functionally weighted neural networks: Frugal models with high accuracy. SN Applied Sciences. 2020;2:1–12.
19. Leake C, Mortari D. Deep theory of functional connections: A new method for estimating the solutions of partial differential equations. Machine Learning and Knowledge Extraction. 2020;2(1):37–55.
20. Aarts LP, Veer P. Neural network method for solving partial differential equations. Neural Process Lett. 2001;14(3):261–71.
21. Sun W, Yuan Y-X. Optimization Theory and Methods: Nonlinear Programming. Dordrecht, The Netherlands: Kluwer Academic Publishers; 2006.
22. Fletcher R. Practical Methods of Optimization. Chichester, West Sussex, UK: John Wiley & Sons Ltd; 2000.
23. Müller J, Zeinhofer M. Notes on exact boundary values in residual minimisation. In: Proceedings of Mathematical and Scientific Machine Learning. Proceedings of Machine Learning Research. 2022; vol. 190, pp. 231–240. PMLR
24. Wang Y, Sun J, Rabczuk T, Liu Y. Dcem: A deep complementary energy method for linear elasticity. International Journal for Numerical Methods in Engineering. 2023; 7585
25. Wang Y, Bai J, Lin Z, Wang Q, Anitescu C, Sun J, Eshaghi MS, Gu Y, Feng X-Q, Zhuang X, Rabczuk T, Liu Y. Artificial intelligence for partial differential equations in computational mechanics: A review. 2024; arXiv:2410.19843.

26. Shampine LF, Reichelt MW. The matlab ode suite. SIAM J Sci Comput. 1997;18(1):1–22.
27. Liu S, Huang X, Protopapas P. Residual-based error bound for physics-informed neural networks. 2023; arXiv:2306.03786.
28. Hairer E, Wanner G. Stiff differential equations solved by Radau methods. J Comput Appl Math. 1999;111:93–111.
29. Papageorgiou DG, Demetropoulos IN, Lagaris IE. Merlin-3.0 a multidimensional optimization environment. Computer Physics Communications. 1998;109:227–49.
30. Papageorgiou DG, Demetropoulos IN, Lagaris IE. Merlin-3.1.1. a new version of the Merlin optimization environment. Computer Physics Communications. 2004;159(1):70–1.
31. Nelder JA, Mead R. A Simplex Method for Function Minimization. Comput J. 1965;7(4):308–13.
32. Powell MJD. A tolerant algorithm for linearly constrained optimization calculations. Math Program. 1989;45(1):547–66.
33. Powell MJD. A new algorithm for unconstrained optimization. In: Rosen JB, Mangasarian KROL, editors. Nonlinear Programming. New York: Academic Press; 1970. p. 31–66.
34. Precious: High Performance Computing Infrastructure. https://www.preciouscloud.eu/
35. Kingma D, Ba J. Adam: A method for stochastic optimization. In: International Conference on Learning Representations. 2014.
36. Liu L, Jiang H, He P, Chen W, Liu X, Gao J, Han J. On the variance of the adaptive learning rate and beyond. 2019; arXiv preprint arXiv:1908.03265.