Chaos
An Interdisciplinary Journal of Nonlinear Science

Parsimonious physics-informed random projection neural networks for initial value problems of ODEs and index-1 DAEs

Cite as: Chaos 33, 043128 (2023); https://doi.org/10.1063/5.0135903
Submitted: 23 November 2022 • Accepted: 20 March 2023 • Published Online: 13 April 2023

Gianluca Fabiani, Evangelos Galaris, Lucia Russo, et al.

Parsimonious physics-informed random projection neural networks for initial value problems of ODEs and index-1 DAEs

Cite as: Chaos 33, 043128 (2023); doi: 10.1063/5.0135903
Submitted: 23 November 2022 · Accepted: 20 March 2023 ·
Published Online: 13 April 2023

Gianluca Fabiani,[1] Evangelos Galaris,[2] Lucia Russo,[3] and Constantinos Siettos[2,a)]

AFFILIATIONS

[1]Scuola Superiore Meridionale, Largo San Marcellino 10, 80138 Napoli (NA), Italy
[2]Dipartimento di Matematica e Applicazioni "Renato Caccioppoli," Università degli Studi di Napoli Federico II, Corso Umberto I 40, 80138 Napoli (NA), Italy
[3]Istituto di Scienze e Tecnologie per l'Energia e la Mobilità Sostenibili, Consiglio Nazionale delle Ricerche, Via Guglielmo Marconi 4, 80125 Napoli (NA), Italy

[a)]**Author to whom correspondence should be addressed:** constantinos.siettos@unina.it

ABSTRACT

We present a numerical method based on random projections with Gaussian kernels and physics-informed neural networks for the numerical solution of initial value problems (IVPs) of nonlinear stiff ordinary differential equations (ODEs) and index-1 differential algebraic equations (DAEs), which may also arise from spatial discretization of partial differential equations (PDEs). The internal weights are fixed to ones while the unknown weights between the hidden and output layer are computed with Newton's iterations using the Moore–Penrose pseudo-inverse for low to medium scale and sparse QR decomposition with L^2 regularization for medium- to large-scale systems. Building on previous works on random projections, we also prove its approximation accuracy. To deal with stiffness and sharp gradients, we propose an adaptive step-size scheme and address a continuation method for providing good initial guesses for Newton iterations. The "optimal" bounds of the uniform distribution from which the values of the shape parameters of the Gaussian kernels are sampled and the number of basis functions are "parsimoniously" chosen based on bias-variance trade-off decomposition. To assess the performance of the scheme in terms of both numerical approximation accuracy and computational cost, we used eight benchmark problems (three index-1 DAEs problems, and five stiff ODEs problems including the Hindmarsh–Rose neuronal model of chaotic dynamics and the Allen–Cahn phase-field PDE). The efficiency of the scheme was compared against two stiff ODEs/DAEs solvers, namely, `ode15s` and `ode23t` solvers of the MATLAB ODE suite as well as against deep learning as implemented in the DeepXDE library for scientific machine learning and physics-informed learning for the solution of the Lotka–Volterra ODEs included in the demos of the library. A software/toolbox in Matlab (that we call RanDiffNet) with demos is also provided.

We address a machine-learning-based method for the numerical solution of stiff ODEs and index-1 DAEs, thus exploiting the universal approximation properties of random projections with Gaussian Kernels. This is the first time it is shown that a machine learning scheme may be comparable, and in several cases better, in terms of both numerical accuracy, and importantly, computational cost when compared to established/traditional stiff solvers such as ode23t and ode15s of the Matlab ODE suite and also deep-learning PINNs as implemented in the DeepXDE library. Thus, we believe that this work may trigger further developments in the field of scientific machine learning for the numerical solution of differential equations.

I. INTRODUCTION

The interest in using machine learning as an alternative to classical numerical analysis methods[1–4] for the solution of inverse,[5–14] and forward[15–19] problems in differential equations modeling of dynamical systems can be traced back three decades ago. Today, this interest has been boosted together with our need to better understand and analyze the emergent dynamics of complex multiphysics/ multiscale dynamical systems of fundamental theoretical and technological importance.[20] The objectives are mainly two. First, that of the solution to the inverse problem, i.e., that of identifying/discovering hidden macroscopic laws, thus learning nonlinear operators and constructing coarse-scale dynamical

models of ordinary differential equations (ODEs) and partial differential equations (PDEs) and their closures, from microscopic large-scale simulations and/or from multi-fidelity observations.[21–33] Second, based on the constructed coarse-scale models, that of systematically investigating the dynamics by efficiently solving the corresponding differential equations, especially when dealing with stiff problems and PDEs.[24,30,31,34–44] Toward this aim, physics-informed machine learning[20,22–24,30–32,45] has been addressed to integrate available/incomplete information from the underlying physics, thus relaxing the "curse of dimensionality" of machine/ deep-learning schemes. In particular, the term "physics-informed neural networks" (PINNs) was coined[24] to describe NNs that are trained to solve the forward and inverse problem for differential equations, incorporating information about the governing equations, initial and boundary conditions (for PDEs), thus providing analytically the necessary derivatives that are needed for the training phase, using, for example, automatic differentiation. However, failures may arise at the training phase especially in deep learning formulations, while there is still the issue of the corresponding computational cost.[20,46–48] Thus, a bet and challenge is to develop physics-informed machine learning methods that can achieve high approximation accuracy at a low computational cost.

Within this framework and toward this aim, we propose a physics-informed neural network (PIRPNN) scheme based on the concept of random projections[49–54] for the numerical solution to initial-value problems of nonlinear stiff ODEs and index-1 differential algebraic equations (DAEs) as these may also arise from spatial discretization of PDEs. Our scheme consists of a single hidden layer, with Gaussian kernels, in which the weights between the input and hidden layer are fixed to ones. The shape parameters of the Gaussian kernels are random variables drawn i.i.d. from a uniform distribution, for which the bounds and the number of basis functions are "parsimoniously" chosen based on the expected bias-variance trade-off decomposition,[55] that was numerically computed using as reference the van der Pol stiff ODEs. The unknown parameters, i.e., the weights between the hidden and the output layer are estimated by solving a system of nonlinear algebraic equations with quasi-Newton iterations. For low- to medium-scale systems, this task is performed using singular value decomposition (SVD), while for medium- to large-scale systems, we exploit a sparse QR factorization algorithm with L^2 regularization.[56] Furthermore, to facilitate the convergence of Newton's iterations, especially at very stiff regimes and regimes with very sharp gradients, we address an adaptive scheme for adjusting the step-size of integration and a natural continuation method for providing good initial guesses for unknown weights.

We compared the performance of the proposed scheme in terms of both approximation accuracy and computational cost, based on eight benchmark problems, three index-1 DAEs and five stiff problems of ODEs, thus comparing it with `ode23t` and `ode15s` adaptive step-size solvers of the MATLAB ODE suite.[4] In particular, we considered the index-1 DAE Robertson model describing the kinetics of an autocatalytic reaction,[57,58] a non autonomous index-1 DAEs model describing the motion of a bead on a rotating needle,[58] a non autonomous index-1 DAEs model describing the dynamics of a power discharge control problem,[58] the van der Pol model, the Prothero-Robinson stiff ODE, the Hindmarsh–Rose neuronal model,[59] the Belousov–Zabotinsky chemical kinetics stiff ODEs,[60,61] and the Allen–Chan phase-field PDE describing the process of phase separation for generic interfaces[62] discretized in space with central finite differences, thus resulting into a system of stiff ODEs.[63] The comparison is performed both on the grid of points resulting from the corresponding adaptive step-size procedure and on dense grids of equidistant points; the evaluation of the solution on dense grids of equidistant points is required, for example, when one studies the (statistical) properties of chaotic and quasi-periodic dynamics/time series using techniques such as the fast Fourier transform (FFT). The results show that the proposed PIRPNN scheme as implemented here (see Sec. IV) outperforms `ode23t` for almost all benchmark problems, while for low-dimensional systems, it also outperforms `ode15s` when the evaluation of the solution is required in a dense grid of points. The structure of the paper is as follows. In Sec. II, we state the problem and provide some preliminaries on the solution of differential equations with PINNs (Subsection II B); we also discuss briefly the concept of random projections (Subsection II C). Furthermore, in Subsection II D, we describe our approach for the solution of IVPs of index-1 DAEs and ODEs with the use of PIRPNNs and discuss its approximation properties within the framework of the universal approximation theorem of random projections.[50,64] In Sec. III, we address an adaptive step-size scheme for adjusting the interval of integration as well as a numerical natural continuation method for providing "good" initial guesses to facilitate the convergence of Newton's iterations; in Subsection III C, we also describe a way to choose the bounds of the uniform distribution from which the values of the shape parameters are drawn. In Sec. IV, we present the numerical results obtained by applying the proposed approach to the above-mentioned stiff ODE and DAE problems along with a comparison with `ode23t/23t` and `ode15s`. In Subsection IV I, we present a further comparison against a deep learning PINN as implemented in the DeepXDE library for the Lotka–Volterra ODEs. Conclusions are given in Sec. V.

II. METHODS

In what follows, we first describe the problem and present some preliminaries on the use of machine learning for the solution to differential equations and on the concept of random projections for the approximation of continuous functions. We then present the proposed physics-informed random projection neural network (PIRPNN) scheme, and building on previous works,[50] we prove that in principle the proposed network can approximate with any given accuracy any unique continuously differentiable function that satisfies the Picard–Lindelöf Theorem. Finally, we (a) propose an adaptive step-size scheme for adjusting the interval of integration based on the elementary local error control algorithm[65] and (b) address a natural continuation method to facilitate the convergence of Newton's iterations, especially in regimes with high stiffness and very sharp gradients.

A. Description of the problem

Here, we consider initial-value problems (IVPs) of ODEs and index-1 DAEs that may also arise from spatial discretization of PDEs

using, for example, finite differences, finite elements, and spectral methods. In particular, we consider IVPs in the linear implicit form of

$$\boldsymbol{M}\frac{d\boldsymbol{u}(t)}{dt} = \boldsymbol{f}(t, \boldsymbol{u}(t)), \quad \boldsymbol{u}(0) = \boldsymbol{z}. \tag{1}$$

$\boldsymbol{u} \in \mathbb{R}^m$ denotes the set of the states $\{u_1, u_2, \ldots, u_i, \ldots, u_m\}$, $\boldsymbol{M} \in \mathbb{R}^{m\times m}$ is the so-called mass matrix with elements M_{ij}, $\boldsymbol{f}: D \subseteq \mathbb{R} \times \mathbb{R}^m \to \mathbb{R}^m$ denotes a Lipschitz continuous multivariate function, with components $f_i(t, u_1, u_2, \ldots, u_m)$ defined in a closed domain D, and $\boldsymbol{z} \in \mathbb{R}^m$ are the initial conditions. When $\boldsymbol{M} = \boldsymbol{I}$, the system reduces to the canonical form. The above formulation includes problems of DAEs when $\boldsymbol{M}$ is a singular matrix, including semi-explicit DAEs in the form[58]

$$\begin{aligned} \frac{d\boldsymbol{u}(t)}{dt} &= \boldsymbol{f}(t, \boldsymbol{u}(t), \boldsymbol{v}(t)), \quad \boldsymbol{u}(0) = \boldsymbol{z}, \\ \boldsymbol{0} &= \boldsymbol{g}(t, \boldsymbol{u}(t), \boldsymbol{v}(t)), \end{aligned} \tag{2}$$

where now $\boldsymbol{f}: \mathbb{R} \times \mathbb{R}^{m-l} \times \mathbb{R}^l \to \mathbb{R}^{m-l}$), $\boldsymbol{g}: \mathbb{R} \times \mathbb{R}^{m-l} \times \mathbb{R}^l \to \mathbb{R}^l$ and we assume that the Jacobian $\nabla_{\boldsymbol{v}}\boldsymbol{g}$ is nonsingular. In this work, we use physics-informed random projection neural networks for the numerical solution of the above type of IVPs, in which solutions are characterized by both sharp gradients and stiffness.[58,66] At this point, it is worthy to emphasize that stiffness is not connected to the presence of steep gradients. For example, at the regimes where relaxation oscillations of the van der Pol model exhibit very sharp changes resembling discontinuities, the equations are not stiff.[66]

B. Physics-informed machine learning for the solution to differential equations

In this section and for the completeness of presentation, we first give a very brief introduction to the basic concept of physics-informed machine learning for the solution to differential equations in the form of PDEs. The proposed methodology for the solution of ODEs and index-1 DAEs that is the subject of the current work is given in Sec. II D.

Let us assume a set of n_x points $\boldsymbol{x}_i \in \Omega \subset \mathbb{R}^d$ of independent (spatial) variables that define the mesh in the domain Ω, $n_{\partial\Omega}$ points along the boundary $\partial\Omega$ of the domain and n_t points in the time interval, where the solution is sought. For our illustrations, let us consider a time-dependent PDE in the form of

$$\frac{\partial u}{\partial t} = L(\boldsymbol{x}, u, \nabla u, \nabla^2 u), \tag{3}$$

where L is the partial differential operator acting on u satisfying the boundary conditions $Bu = g$, in $\partial\Omega$, where B is the boundary differential operator. Then, the solution with machine learning of the above PDE involves the solution to a minimization problem in the form

$$\begin{aligned} \min_{\boldsymbol{P},\boldsymbol{Q}} E(\boldsymbol{P}, \boldsymbol{Q}) := &\sum_{i=1}^{n_x}\sum_{j=1}^{n_t}\left\| \frac{\partial \Psi}{\partial t}(\cdot) - L(\boldsymbol{x}_i, \Psi(\cdot), \nabla\Psi(\cdot), \nabla^2\Psi(\cdot))\right\|^2 \\ &+ \sum_{j=1}^{n_{\partial\Omega}} \left\| B\Psi(\cdot) - g\right\|^2, \end{aligned} \tag{4}$$

where $\Psi(\cdot) := \Psi(\boldsymbol{x}_i, t_j, \mathcal{N}(\boldsymbol{x}_i, t_j, \boldsymbol{P}, \boldsymbol{Q}))$ represents a machine learning constructed function approximating the solution u at $\boldsymbol{x}_i$ at time t_j and $\mathcal{N}(\boldsymbol{x}_i, t_j, \boldsymbol{P}, \boldsymbol{Q})$ is a machine learning algorithm; $\boldsymbol{P}$ contains the parameters of the machine learning scheme (e.g., for a neural network, the internal weights $\boldsymbol{W}$, the biases $\boldsymbol{B}$, and the weights between the last hidden and the output layer $\boldsymbol{W}^o$), $\boldsymbol{Q}$ contains hyperparameters (e.g., the parameters of the activation functions for a neural network, the learning rate, etc.). In order to solve the optimization problem (4), one usually needs quantities such as the derivatives of $\mathcal{N}(\boldsymbol{x}, \boldsymbol{P}, \boldsymbol{Q})$ with respect to $t, \boldsymbol{x}$ and the parameters of the machine learning scheme, such as the weights and biases. These can be obtained numerically using finite differences or other approximation schemes or by symbolic or automatic differentiation.[40,67]

The above approach can be implemented also for solving systems of ODEs/DAEs as these may also arise by discretizing in space PDEs. For example, if we consider a 1D PDE and a grid of n_x equispaced points x_i, $i = 1, \ldots, n_x$, with a space-step Δx, we can discretize the profile in space and approximate the spatial derivatives using, e.g., central finite differences (FD) to get

$$\begin{aligned} \frac{\partial u(t,x)}{\partial x} &= \frac{u_{i+1}(t) - u_{i-1}(t)}{2\Delta x}, \\ \frac{\partial^2 u(t,x)}{\partial x^2} &= \frac{u_{i+1}(t) - 2u_i(t) + u_{i-1}(t)}{\Delta x^2}. \end{aligned} \tag{5}$$

Therefore, in this case, Eq. (3) can be reduced to a system of n_x DAEs, given by

$$\begin{aligned} \frac{du_i(t)}{dt} &= \tilde{L}(x_i, u_1(t), \ldots, u_{n_x}(t)), \quad i = 2, \ldots, (n_x - 1), \\ \tilde{B}(u_1) &= g(t, x_1), \quad \tilde{B}(u_{n_x} = g(t, x_{n_x}), \end{aligned} \tag{6}$$

where $\tilde{L}, \tilde{B}$ correspond to the discretization of operators L and B, respectively, with FD. Then, the solution of the discretized system can be sought using n_x (space-independent) machine learning constructed functions $\Psi_i(\cdot) := \Psi_i(t_j, \mathcal{N}_i(t, \boldsymbol{P}, \boldsymbol{Q}))$ approximating the solution u_i at time t_j. Thus, the minimization problem given by Eq. (3) reduces to

$$\begin{aligned} \min_{\boldsymbol{P},\boldsymbol{Q}} E(\boldsymbol{P}, \boldsymbol{Q}) := &\sum_{i=1}^{n_x}\sum_{j=1}^{n_t}\left\| \frac{d\Psi_i}{dt}(\cdot) - \tilde{L}(\boldsymbol{x}_i, \Psi_1(\cdot), \ldots, \Psi_{n_x}(\cdot))\right\|^2 \\ &+ \left\|\tilde{B}\Psi_1(\cdot) - g(t_j, x_1)\right\|^2 + \left\|\tilde{B}\Psi_{n_x}(\cdot) - g(t_j, x_{n_x})\right\|^2. \end{aligned} \tag{7}$$

Yet, for deep learning schemes, but even for the simple case of single layer networks, when the number of hidden nodes is large, the solution of the resulting large-scale optimization problem is known to be difficult, often resulting in poor solutions as iterations stuck in local minima (for a detailed discussion about these problems, see e.g., Refs. 46–48.)

C. Random projection neural networks

Random projection neural networks (RPNN) including random vector functional link networks (RVFLNs),[68,69] echo-state neural networks/reservoir computing,[53,70,71] and extreme learning machines[54,72] share common concepts and have been implemented

to relax the "curse of dimensionality" encountered at the training phase.

A fundamental work on random projections that justifies why and how a random projection works is the celebrated Johnson and Lindenstrauss Lemma[49] (for a review and tutorial, see also Ref. 73) stating that for a matrix $\boldsymbol{W} \in \mathbb{R}^{d\times n}$, containing n sample data points $\boldsymbol{w}$ in $\mathbb{R}^d$, there exists a projection $\boldsymbol{F}: \mathbb{R}^d \to \mathbb{R}^k$ defined as

$$\boldsymbol{F}(\boldsymbol{w}) = \frac{1}{\sqrt{k}}\boldsymbol{R}\boldsymbol{w}, \tag{8}$$

where $\boldsymbol{R} = [r_{ij}] \in \mathbb{R}^{k\times d}$ has components that are i.i.d. random variables sampled from a normal distribution, which maps $\boldsymbol{W}$ into a random subspace of dimension $k \geq O\left(\frac{\ln n}{\varepsilon^2}\right)$, where the distance between any pair of points in the embedded space $\boldsymbol{F}(\boldsymbol{W})$ is bounded in $[1-\varepsilon \quad 1+\varepsilon]$, with a high probability. While the JL linear random projection is one of the possible choices, it has been experimentally demonstrated and/or theoretically proven that appropriately constructed nonlinear random projections (that can be modelled by a linear random projection followed by a nonlinear function) may outperform such simple linear random projections as in the JL Lemma (see, e.g., Refs. 50, 51, 74, and 73).

Regarding single-layer feedforward neural networks (SLFNNs), Rosenblatt[75] suggested the use of randomly parametrized activation functions for single-layer structures to simplify computations. Thus, the approximation of a sufficiently smooth function $f(\boldsymbol{x}): \mathbb{R}^d \to \mathbb{R}$ is written as a linear combination of appropriately randomly parametrized family of N basis functions $\phi_i: \mathbb{R}\times\mathbb{R}^p \to \mathbb{R}$ as

$$f(\boldsymbol{x}) \simeq f_N(\boldsymbol{x}) = \sum_{i=1}^{N} w_i^o \phi_i\left(\boldsymbol{w}_i^T\boldsymbol{x} + b_i, \boldsymbol{p}_i\right), \tag{9}$$

where $\boldsymbol{w}_i \in \mathbb{R}^d$ are the weighting coefficients of the inputs, $b \in \mathbb{R}$ are the biases, and $\boldsymbol{p}_i \in \mathbb{R}^p$ are shape parameters of basis functions.

More generally, for SLFNNs with d inputs, k outputs, and N neurons in the hidden layer, the random projection of n samples in the d-dimensional input space $\boldsymbol{X}$ can be written in a matrix-vector form as

$$\boldsymbol{Y}_N = \boldsymbol{\Phi}_N\boldsymbol{W}^o, \quad \boldsymbol{Y} \in \mathbb{R}^{n\times k}, \tag{10}$$

where $\boldsymbol{\Phi}_N \in \mathbb{R}^{n\times N}$ is a random matrix containing the outputs of the hidden layer as shaped by n samples in the d-dimensional space, the randomly parametrized internal weights $\boldsymbol{W} \in \mathbb{R}^{d\times N}$, the biases $\boldsymbol{b} \in \mathbb{R}^N$, and shape parameters of the N activation functions; $\boldsymbol{W}^o \in \mathbb{R}^{N\times k}$ is the matrix containing the weights w_{ij}^o between the hidden and the output layer.

In the early 90s, Schmidt et al. used single layer neural networks with random weights for the hidden layer and least squares to train the output weights.[94] Barron[76] proved that for functions with integrable Fourier transformations, a random sample of the parameters of sigmoidal basis functions from an appropriately chosen distribution results to an approximation error of the order of $1/N$. Igelnik and Pao[68] extended Barron's proof[76] for any family of L^2 integrable basis functions ϕ_i, thus addressing the so-called RVFLNs. For the so-called extreme-learning machines (ELMs), which similarly to RVFLNs are FNNs with randomly assigned internal weights and biases of the hidden layers, Huang *et al.*[54,72] have proved uniform convergence with a probability 1 under certain assumptions. Similar results for one-layer schemes have also been reported in other studies (see, e.g., Refs. 50, 77, and 51). Rahimi and Recht[50,77] proved the following theorem:

Theorem 1 (cf. Lemma 1 in Ref. 50). *Consider the feature functions $\phi(x,\alpha): \mathscr{X}\times\mathscr{U} \to R$ parametrized by some vector $\alpha \in \mathscr{U}$ that satisfy $sup_{x,\alpha}\phi(x,\alpha) \leq 1$. Let $p(\alpha)$ be a probability distribution on $\mathscr{U}$. Define the set of functions*

$$\mathscr{F}_p \equiv \left\{f(x) = \int_{\mathscr{U}} w(\alpha)\phi(x;\alpha)d\alpha, |w(\alpha)| \leq Cp(\alpha)\right\}, \tag{11}$$

and let μ be a probability measure on $\mathscr{X}$. Then, if one takes a function f^* in $\mathscr{F}_p$, and N values $\alpha_1, \alpha_2, \ldots, \alpha_N$ of the shape parameter α drawn i.i.d. from p, then for any $\delta > 0$ with probability at least $1-\delta$ over $\alpha_1, \alpha_2, \ldots, \alpha_N$, there exists a function $\hat{f}$ that lies in the random set

$$\hat{\mathscr{F}}_\alpha \equiv \left\{f(x) = \sum_{j=1}^{N} w_j\phi_j(x;\alpha_j), |w_j| \leq \frac{C}{N}\right\} \tag{12}$$

such that

$$\sqrt{\int_{\mathscr{X}}\left(f^*(\boldsymbol{x}) - \hat{f}(\boldsymbol{x})\right)^2 d\mu(x)} \leq \frac{C}{\sqrt{N}}\left(1 + \sqrt{2\log\frac{1}{\delta}}\right). \tag{13}$$

For a detailed discussion on the pros and cons of function approximation with random basis functions, see Ref. 51.

D. The proposed physics-informed random projection neural network for the solution of ODEs and index-1 DAEs

Here, we propose a physics-informed machine learning scheme based on the concept of random projections, and particularly based on Theorem 1 (see also[50,51,77]) for the solution of IVPs of systems given by Eqs. (1) and (2) in n collocation points in an interval, say $[t_0 \quad t_f]$. Based on the above, the output of the random projection network is spanned by the range $\mathscr{R}(\boldsymbol{\Phi})$, i.e., the column vectors of $\boldsymbol{\Phi}_N$, say $\boldsymbol{\phi}_i \in \mathbb{R}^n$. Hence, the output of the network can be written as

$$\boldsymbol{Y}_N = \sum_{i=1}^{N} w_i^o\boldsymbol{\phi}_i \tag{14}$$

For an IVP of m variables, we construct m such networks.

Let us denote by $\boldsymbol{\Psi}(t, \boldsymbol{W}, \boldsymbol{W}^o, \boldsymbol{P})$ the set of trial functions $\Psi_{Ni}(t, \boldsymbol{w}_i^o, \boldsymbol{p}_i), i = 1, 2, \ldots, m$ that approximate the solution profile u_i at time t, which is defined as

$$\Psi_{Ni}(t, \boldsymbol{w}_i, \boldsymbol{w}_i^o, \boldsymbol{p}_i) = z_i + (t-t_0)\boldsymbol{w}_i^{oT}\boldsymbol{\Phi}_{Ni}(t, \boldsymbol{w}_i, \boldsymbol{p}_i), \tag{15}$$

where $\boldsymbol{\Phi}_{Ni}(t, \boldsymbol{w}_i, \boldsymbol{p}_i) \in \mathbb{R}^N$ is the column vector containing the values of N basis functions at time t as shaped by $\boldsymbol{w}_i$ and $\boldsymbol{p}_i$ containing the values of the parameters of N basis functions and $\boldsymbol{w}_i^o = [w_{1i}^o \; w_{2i}^o \; \ldots \; w_{Ni}^o]^T \in \mathbb{R}^N$ is the vector containing the values of the output weights of the ith network. Note that the above set of functions are continuous functions of t and explicitly satisfy the initial conditions.

For index-1 DAEs, with say $M_{ij} = 0, \forall i \geq l, j = 1, 2, \ldots, m$, or in the semi-explicit form of (2), there are no explicit initial conditions z_i for the variables u_i, $i = l, l+1, \ldots, m$, or the variables $\boldsymbol{v}$ in (2): these values have to satisfy the constraints $f_i(t, \boldsymbol{u}) = 0$, $i \geq l$, (equivalently $\boldsymbol{0} = \boldsymbol{g}(t, \boldsymbol{u}, \boldsymbol{v})$) $\forall t$, starting with consistent initial conditions. Assuming that the corresponding Jacobian matrix of the $f_i(t, \boldsymbol{u}) = 0$, $i = l, l+1, \ldots, m$ with respect to u_i, and for the semi-explicit form (2), $\nabla_{\boldsymbol{v}}\boldsymbol{g}$, is not singular, one has to solve initially at $t = 0$, using, for example, Newton–Raphson iterations, the above nonlinear system of $m - l$ algebraic equations in order to find a consistent set of initial values. Then, one can write the approximation functions of the u_i, $i = l, l+1, \ldots, m$ [or $\boldsymbol{v}$ in the case of semi-explicit form (2)] as in Eq. (15).

With n collocation points in $[t_0 \quad t_f]$, by fixing the values of the interval weights $\boldsymbol{w}_i$ and the shape parameters $\boldsymbol{p}_i$, the loss function that we seek to minimize with respect to the unknown coefficients $\boldsymbol{w}_i^o$ is given by

$$\mathscr{L}(\boldsymbol{W}^o) = \sum_{k=1}^{n}\sum_{i=1}^{m}\sum_{j=1}^{m}\Big(M_{ij}\frac{d\Psi_{Ni}}{dt}(t_k, \boldsymbol{w}_i, \boldsymbol{w}_i^o, \boldsymbol{p}_i) - f_i\big(t_k, \boldsymbol{\Psi}(t_k, \boldsymbol{W}, \boldsymbol{W}^o, \boldsymbol{P})\big)\Big)^2. \tag{16}$$

When the system of ODEs/DAEs results from the spatial discretization of PDEs, we assume that the corresponding boundary conditions have been appropriately incorporated into the resulting algebraic equations explicitly or otherwise can be added in the loss function as algebraic constraints. Here, for each $\mathscr{N}_i$, we take N Gaussian kernels given by:

$$g_{ji}(t, w_{ji}, b_{ji}, \alpha_{ji}, c_j) = e^{-\alpha_{ji}(w_{ji}t + b_{ji} - c_j)^2}, \quad j = 1, \ldots, N, i = 1, \ldots, m. \tag{17}$$

The values of the (hyper) parameters, namely, w_{ji}, b_{ji}, and c_j are set as

$$w_{ji} = 1, \quad b_{ij} = 0, \quad c_j = t_j = t_0 + (j-1)\frac{t_f - t_0}{N-1},$$

while the values of the shape parameters $\alpha_{ji} > 0$ are sampled from an appropriately chosen uniform distribution (see below). The time derivative of Ψ_{Ni} is given by

$$\frac{d\Psi_{Ni}}{dt} = \sum_{j=1}^{N} w_{ji}^o e^{-\alpha_{ji}(t-t_j)^2} - 2(t - t_0)\sum_{j=1}^{N}\alpha_{ji}w_{ji}^o(t - t_j)e^{-\alpha_{ji}(t-t_j)^2}. \tag{18}$$

1. Approximation with the PIRPNN

Here, we show that the PIRPNN given by Eq. (15) is a universal approximator of the solution $\boldsymbol{u}$ of the ODEs in the canonical form or of the index-1 DAEs in the semi-explicit form (2).

Proposition 1. *For the IVP problem (1) in the canonical form or in the semi-explicit form (2) for which the Picard–Lindelöf Theorem[78] holds true, the PIRPNN solution Ψ_{Ni} given by Eq. (15) with N Gaussian basis functions defined by Eq. (17) whose shape parameters α_{ji} are drawn i.i.d. from a uniform distribution across the sample space, converges uniformly to the actual solution profile $\boldsymbol{u}(t)$ in a closed time interval $[t_0 \quad t_f]$ with an upper bound of the order of $O\left(\frac{1}{\sqrt{N}}\right)$ with a probability $1 - \delta$ for any small $\delta > 0$.*

Proof. Assuming that the system in Eq. (1) can be written in the canonical form and the Picard–Lindelöf Theorem[78] holds true, then it exists a unique continuously differentiable function defined on a closed time interval $[t_0 \quad t_f]$ given by

$$u_i(t) = z_i + \int_{t_0}^{t} f_i(s, \boldsymbol{u}(s))ds, \quad i = 1, 2, \ldots m. \tag{19}$$

From Eq. (15), we have

$$\Psi_{Ni}(t) = z_i + (t - t_0)\sum_{j=1}^{N} w_j^o e^{-\alpha_j(t-t_j)^2}. \tag{20}$$

By the change of variables, $\tau = \dfrac{s - t_0}{t - t_0}$, the integral in Eq. (19) becomes

$$\int_{t_0}^{t} f_i(s, \boldsymbol{u}(s))ds = (t - t_0)\int_0^1 f_i(\tau(t - t_0) + t_0, \boldsymbol{u}(\tau(t - t_0) + t_0))d\tau. \tag{21}$$

Hence, by Eqs. (19), (20), and (21), we have

$$I_n(t) \equiv \int_0^1 f_i(\tau(t - t_0) + t_0, \boldsymbol{u}(\tau(t - t_0) + t_0))d\tau \approx \sum_{j=1}^{N} w_j^o e^{-\alpha_j(t-t_j)^2}. \tag{22}$$

Thus, in fact, upon convergence, the PIRPNN provides an approximation of the normalized integral. By Theorem 2.1, we have that in the interval $[t_0 \quad t_f]$, the PIRPNN with the shape parameter of the Gaussian kernel drawn i.i.d. from a uniform distribution provides a uniform approximation of the integral in Eq. (19) in terms of a Monte Carlo integration method as also described in Ref. 68. Hence, as the initial conditions are explicitly satisfied by $\Psi_{Ni}(t)$, we have from Eq. (13) an upper bound for the uniform approximation of the solution profile u_i.

For index-1 DAEs in the semi-explicit form of (2), by the implicit function theorem, we have that the DAE system is *in principle* equivalent to the ODE system in the canonical form,

$$\frac{d\boldsymbol{u}(t)}{dt} = \boldsymbol{f}(t, \boldsymbol{u}(t), \mathscr{H}(t, \boldsymbol{u})), \tag{23}$$

where $\boldsymbol{v}(t) = \mathscr{H}(t, \boldsymbol{u}(t))$ is the unique solution of $\boldsymbol{0} = \boldsymbol{g}(t, \boldsymbol{u}(t), \boldsymbol{v}(t))$. Hence, in that case, the proof of convergence reduces to the one above for the ODE system in the canonical form. □

2. Computation of the unknown weights

For n collocation points, the outputs of each network $\mathcal{N}_i \equiv \mathcal{N}_i(t_1, t_2, \dots t_n, \boldsymbol{w}_i^o, \boldsymbol{p}_i) \in \mathbb{R}^n$, $i = 1, 2, \dots m$ read

$$\mathcal{N}_i = \boldsymbol{R}_i \boldsymbol{w}_i^o,$$
$$\boldsymbol{R}_i \equiv \boldsymbol{R}_i(t_1, \dots, t_n, \boldsymbol{p}_i) = \begin{bmatrix} g_{1i}(t_1) & \cdots & g_{Ni}(t_1) \\ \vdots & \vdots & \vdots \\ g_{1i}(t_n) & \cdots & g_{Ni}(t_n) \end{bmatrix}. \tag{24}$$

The minimization of the loss function (16) is performed over the nm nonlinear residuals F_q,

$$F_q(\boldsymbol{W}^o) = \sum_{j=1}^{m} M_{ij} \frac{d\Psi_{Nj}}{dt_l}(t_l, \boldsymbol{w}_j^o) - f_i(t_l, \Psi_{N1}(t_l, \boldsymbol{w}_1^o), \dots, \Psi_{Nm}(t_l, \boldsymbol{w}_m^o)), \tag{25}$$

where $q = l + (i-1)n$, $i = 1, 2, \dots m$, $l = 1, 2, \dots n$, $\boldsymbol{W}^0 \in \mathbb{R}^{mN}$ is the column vector obtained by collecting the values of all m vectors $\boldsymbol{w}_i^o \in \mathbb{R}^N$, $\boldsymbol{W}^o = [W_k^o] = [\boldsymbol{w}_1^o, \boldsymbol{w}_2^o \dots, \boldsymbol{w}_m^o]^T$, $k = 1, 2, \dots, mN$. Thus, the solution to the above non-linear least squares problem can be obtained, e.g., with Newton-type iterations such as Newton–Raphson, quasi-Newton and Gauss–Newton methods (see, e.g., 79). For example, by setting $\boldsymbol{F}(\boldsymbol{W}^o) = [F_1(\boldsymbol{W}^o) \cdots F_q(\boldsymbol{W}^o) \cdots F_{(nm)}(\boldsymbol{W}^o)]^T$, the update $d\boldsymbol{W}^{o(\nu)}$ at the (ν)th Gauss-Newton iteration is computed by the solution of the linearized system,

$$\nabla_{\boldsymbol{W}^{o(\nu)}} \boldsymbol{F} d\boldsymbol{W}^{o(\nu)} = -\boldsymbol{F}(\boldsymbol{W}^{o(\nu)}), \tag{26}$$

where $\nabla_{\boldsymbol{W}^{o(\nu)}} \boldsymbol{F} \in \mathbb{R}^{nm \times mN}$ is the Jacobian matrix of $\boldsymbol{F}$ with respect to $\boldsymbol{W}^{o(\nu)}$. Note that the residuals depend on the derivatives $\frac{\partial \Psi_{Ni}(\cdot)}{\partial t_l}$ and the approximation functions $\Psi_{Ni}(\cdot)$, while the elements of the Jacobian matrix depend on the derivatives of $\frac{\partial \Psi_{Ni}(\cdot)}{\partial w_{ji}^o}$ as well as on the mixed derivatives $\frac{\partial^2 \Psi_{Ni}(\cdot)}{\partial t_l \partial w_{ji}^o}$. Based on (18), the latter are given by

$$\frac{\partial^2 \Psi_{Ni}}{\partial t_l \partial w_{ji}^o} = \frac{\partial \mathcal{N}_i(t_l, \boldsymbol{w}_i^o, \boldsymbol{p}_i)}{\partial w_{ji}^o} - 2(t_l - t_0)\alpha_{ji}(t_l + b_{ji} - c_j) e^{(-\alpha_{ji}(t_l + b_{ji} - c_j)^2)}. \tag{27}$$

Thus, the elements of $\nabla_{\boldsymbol{W}^{o(\nu)}} \boldsymbol{F}$ are given by

$$\frac{\partial F_q}{\partial W_p^o} = \frac{\sum_{j=1}^{m} M_{ij} \partial^2 \Psi_{Ni}(\cdot)}{\partial t_l \partial w_{jk}^o} - \frac{\partial f_i(t_l)}{\partial w_{jk}^o}, \tag{28}$$
$$q = l + (i-1)n, p = j + (k-1)N.$$

However, even when $N \geq n$, the Jacobian matrix is expected to be rank deficient or nearly rank deficient, since some of the rows due to the random construction of basis functions can be nearly linear dependent.[51] Thus, the solution of the corresponding system, and depending on the size of the problem, can be solved using, for example, truncated SVD decomposition or QR factorization with regularization. The truncated SVD decomposition scheme leads to the Moore–Penrose pseudoinverse (see also in Ref. 95), and the updates $d\boldsymbol{W}^{o(\nu)}$ are given by

$$d\boldsymbol{W}^{o(\nu)} = -(\nabla_{\boldsymbol{W}^{o(\nu)}} \boldsymbol{F})^{\dagger} \boldsymbol{F}(\boldsymbol{W}^{o(\nu)}),$$
$$(\nabla_{\boldsymbol{W}^{o(\nu)}} \boldsymbol{F})^{\dagger} = \boldsymbol{V}_{\varepsilon} \boldsymbol{\Sigma}_{\varepsilon}^{\dagger} \boldsymbol{U}_{\varepsilon}^T, \tag{29}$$

where $\boldsymbol{\Sigma}_{\varepsilon}^{\dagger}$ is the inverse of the diagonal matrix with singular values of $\nabla_{\boldsymbol{W}^o} \boldsymbol{F}$ above a certain threshold ε, and $\boldsymbol{U}_{\varepsilon}$, $\boldsymbol{V}_{\varepsilon}$ are the matrices with columns corresponding to left and right eigenvectors, respectively. At this point in order to decrease the computational cost, one can implement a Quasi-Newton scheme, thus using the same pseudoinverse of the Jacobian for next iterations until convergence.

For large-scale sparse Jacobian matrices, as those arising, for example, from discretization of PDEs, one can solve the regularization problem using other methods such as sparse QR factorization. Here, to account for the ill-posed Jacobian, we have used a sparse QR factorization with regularization as implemented by SuiteSparseQR, a multifrontal/multithreaded sparse QR factorization package.[56,80]

To summarize, the above scheme provides a *numerical analysis-assisted* PINN in a form that approximates the integral solution of the Picard–Lindelöf theorem based on random projections, thus providing analytically the Jacobian matrix for Newton iterations.

III. PARSIMONIOUS CONSTRUCTION OF THE PIRPNN

A. The adaptive step-size scheme

In order to deal with the presence of stiffness and sharp changes that resemble singularities at the time interval of interest, we propose a adaptive step-size scheme for adjusting the interval of integration as follows. The full time interval of integration $[t_0 \quad t_f]$ is divided into sub-intervals, i.e., $[t_0 \quad t_f] = [t_0 \quad t_1] \cup [t_1 \quad t_2] \cup \dots, \cup [t_k \quad t_{k+1}] \cup \cdots \cup [t_{end-1} \quad t_f]$, where $t_1, t_2, \dots, t_k, \dots, t_{end-1}$ are determined in an adaptive way. This decomposition of the interval leads to the solution of consecutive IVPs. Let us assume that the problem has been solved up to $[t_{k-1} \quad t_k]$; hence, we have found $u_i^{(k-1)}$ and we are seeking to advance $u_i^{(k)}$ in the next interval, say, $[t_k \quad t_{k+1}]$ with a step size of $\Delta t_k = t_{k+1} - t_k$.

At each Newton iteration, say ν (here $\nu \leq \nu_{\max} = 5$), we compute the *normalized residuals (precision to tolerance ratio)* $res_q^{(\nu)}$ for each component $F_q(\boldsymbol{W}^{o(\nu)})$ of $\boldsymbol{F}(\boldsymbol{W}^{o(\nu)})$ as[1,65]

$$res_q^{(\nu)} = \frac{F_q(\boldsymbol{W}^{o(\nu)})}{AbsTol + RelTol \cdot \frac{d\Psi_{Ni}}{dt_l}(t_l, \boldsymbol{w}_i^{o(\nu)})}, \tag{30}$$

where $AbsTol$ is the absolute threshold tolerance, $RelTol$ is the tolerance relative to the size of each derivative component at time t_l, and as in Eq. (25), $q = l + (i-1)n$, $i = 1, 2, \dots m$, $l = 1, 2, \dots n$. Thus, we compute the *approximation error*, say $err^{(\nu)}$, as the l^2-norm of the vector of the normalized residuals $\boldsymbol{res}^{(\nu)} = [res_1^{(\nu)}, res_2^{(\nu)}, \dots, res_{nm}^{(\nu)}]$,

$$err^{(\nu)} = \left\| \boldsymbol{res}^{(\nu)} \right\|_{l^2}. \tag{31}$$

Now, if at the νth iteration, for one $\nu \leq \nu_{max}$, $err^{(\nu)} < 1$, the numerical solution is accepted; otherwise (if up to the last iteration ν_{max}, $err^{(\nu_{max})} \geq 1$), the numerical solution is rejected.

In both cases, the size of the time interval is updated according to the elementary local error control algorithm,[65]

$$\Delta t_k^* = 0.8\gamma \cdot \Delta t_k, \quad \text{with} \quad \gamma = \left(\frac{1}{err}\right)^{\frac{1}{\nu+1}}, \tag{32}$$

where γ is the *scaling factor* and 0.8 is a safe/conservative factor; Δt_k^* is not allowed to increase or decrease a lot, so γ is not higher than a $\gamma_{\max}$ (here set to 4) and smaller than a $\gamma_{\min}$ (here set to 0.1). Thus, if the quasi-Newton scheme does not converge to the desired tolerance within a number of iterations, say $\nu_{\max}$, then the step size is decreased, thus redefining a new guess $t_{k+1}^* = t_k + \Delta t_k^*$ for t_{k+1} and the quasi-Newton scheme is repeated in the interval $[t_k \quad t_{k+1}^*]$. Finally, we note that in the above scheme, the choice of the first subinterval $[t_0 \quad t_1]$ was estimated using an automatic detection code for selecting the starting step as described in Refs. 81 and 82.

B. A continuation method for Newton's iterations

For Newton-type schemes, the speed of the convergence to the solution depends on the choice of the initial guess, here, for the unknown weights. Here, to facilitate the convergence of Newton's iterations, we address a numerical natural continuation method for providing "good" initial guesses for the weights of the PIRPNN.

Suppose that the algorithm has already converged to the solution in the interval $[t_{k-1} \quad t_k]$; we want to provide for the next time interval $[t_k \quad t_{k+1}]$, as computed from the proposed adaptation scheme described above, a good initial guess for the weights of the PIRPNN. We state the following proposition.

Proposition 2. *Let $\boldsymbol{\Psi}(t_k) \in \mathbb{R}^m$ be the solution found with PIRPNN at the end of the time interval $[t_{k-1} \quad t_k]$. Then, an initial guess for the weights of the PIRPNN for the time interval $[t_k \quad t_{k+1}]$ is given by*

$$\hat{W}^o = \frac{d\boldsymbol{\Psi}(t_k)}{dt}\frac{\boldsymbol{\Phi}^T}{||\boldsymbol{\Phi}||_{l_2}^2}, \tag{33}$$

where $\hat{W}^o \in \mathbb{R}^{m\times N}$ is the matrix with the initial guess of the output weights of the m PIRPNNs and $\boldsymbol{\Phi} \in \mathbb{R}^N$ is the vector containing the values of the random basis functions in the interval $[t_k \quad t_{k+1}]$.

Proof. At time t_k, a first-order estimation of the solution, $\boldsymbol{\Psi}(t_{k+1}) \in \mathbb{R}^m$, reads

$$\hat{\boldsymbol{\Psi}}(t_{k+1}) = \boldsymbol{\Psi}(t_k) + \frac{d\boldsymbol{\Psi}(t_k)}{dt}(t_{k+1} - t_k), \tag{34}$$

where $\frac{d\boldsymbol{\Psi}(t_k)}{dt}$ is known. For the next time interval $[t_k \quad t_{k+1}]$, the approximation of the solution with the PIRPNNs is given by

$$\boldsymbol{\Psi}(t_{k+1}) = \boldsymbol{\Psi}(t_k) + (t_{k+1} - t_k)\boldsymbol{W}^o\boldsymbol{\Phi}. \tag{35}$$

By Eqs.(34) and (35), we get

$$\hat{W}^o\boldsymbol{\Phi} = \frac{d\boldsymbol{\Psi}(t_k)}{dt}. \tag{36}$$

It can be easily seen that the truncated SVD of $\boldsymbol{\Phi}$ is given by

$$\boldsymbol{\Phi}_{N\times 1} = \boldsymbol{U}_{N\times 1}\sigma_1, \quad \boldsymbol{U}_{N\times 1} = \frac{\boldsymbol{\Phi}_{N\times 1}}{||\boldsymbol{\Phi}||_{l_2}}, \sigma_1 = ||\boldsymbol{\Phi}||_{l_2}. \tag{37}$$

Thus, the pseudo-inverse of $\boldsymbol{\Phi}$ is $\boldsymbol{\Phi}^\dagger = \frac{\boldsymbol{\Phi}^T}{||\boldsymbol{\Phi}||_{l_2}^2}$. Hence, by Eq. (36), an initial guess for the weights for the time interval $[t_k \quad t_{k+1}]$ is given by

$$\hat{W}^o = \frac{d\boldsymbol{\Psi}(t_k)}{dt}\boldsymbol{\Phi}^\dagger = \frac{d\boldsymbol{\Psi}(t_k)}{dt}\frac{\boldsymbol{\Phi}^T}{||\boldsymbol{\Phi}||_{l_2}^2}. \tag{38}$$

□

C. Estimation of the interval of the uniform distribution based on the variance/bias trade-off decomposition

Based on Eqs. (9) and (14), one has to choose the number N of the basis functions, and the interval, say $\mathscr{U} = [0 \quad \alpha_u]$, from which the values of the shape parameters α_i are drawn based on a probability distribution p. The theorems of uniform convergence (Secs. II C and II D 1) consider the problem from the function approximation point of view. Here, we construct N random vectors by parsimoniously sampling the values of the shape parameter from an appropriately bounded uniform interval for minimizing the two sources of error approximation, i.e., the bias and the variance in order to get good generalization properties. In our scheme, these, over all possible values of the shape parameter α are given by [see Eq. (20)]

$$\begin{aligned} e(t) &= \mathbb{E}\left[\sum_{j=1}^{N} w_j^o e^{-\alpha_j(t-t_j)^2}\right] - I_n(t), \\ \sigma^2(t) &= \mathbb{E}\left[\left(\sum_{j=1}^{N} w_j^o e^{-\alpha_j(t-t_j)^2}\right)^2\right] - \mathbb{E}^2\left[\sum_{j=1}^{N} w_j^o e^{-\alpha_j(t-t_j)^2}\right], \end{aligned} \tag{39}$$

where $\mathbb{E}$ denotes the expectation operator. In the above, overfitting that is connected with a high variance occurs for large values of α and underfitting, which is connected with a high bias (appxoximation error) occurs for small values of α.

The expected value of the kernel $\phi(t - t_j; \alpha) = e^{-a(t-t_j)^2}$, $t \neq t_j$ with respect to the probability density function of the uniform distribution of the random variable α reads

$$\mathbb{E}[\phi(t - t_j; \alpha)] = \int_0^{\alpha_u} f_\alpha(\alpha) e^{-\alpha(t-t_j)^2} d\alpha = \frac{1 - e^{-\alpha_u(t-t_j)^2}}{\alpha_u(t - t_j)^2}. \tag{40}$$

Similarly, the variance is given by

$$\begin{aligned} \sigma^2[\phi(t - t_j; \alpha)] &= \int_{e^{-\alpha_u(t-t_j)^2}}^{1} \phi^2 \frac{1}{\alpha_u(t - t_j)^2}\frac{1}{\phi} d\phi - \mathbb{E}[\phi]^2 \\ &= \frac{1 - e^{-2\alpha_u(t-t_j)^2}}{2\alpha_u(t - t_j)^2} - \mathbb{E}[\phi]^2. \end{aligned} \tag{41}$$

At the limits of $t - t_j = dt = \frac{t_f - t_0}{N}$, from Eqs. (40) and (41), we get

$$\mathbb{E}[\phi(dt;\alpha)] = \frac{N^2}{(t_f - t_0)^2}\frac{1 - e^{-\alpha_u \frac{(t_f - t_0)^2}{N^2}}}{\alpha_u},$$
$$\sigma^2[\phi(dt;\alpha)] = \frac{N^2}{(t_f - t_0)^2}\frac{1 - e^{-2\alpha_u \frac{(t_f - t_0)^2}{N^2}}}{2\alpha_u} - \mathbb{E}[\phi(dt;\alpha)]^2. \tag{42}$$

The above expressions suggest that $\alpha_u = \frac{N^2}{c^2(N)}\frac{1}{(t_f - t_0)^2}$, $c(N) > 0$.

Indeed, our choice of such expression for α_u transform (42), taking rid of the dependence to the time-step $(t - t_0)$ analogously to a renormalization of the input, lead to the following *time-step independent* mean and variance:

$$\mathbb{E}[\phi(dt;\alpha)] = c^2(N)\left(1 - \exp\left(-\frac{1}{c^2(N)}\right)\right),$$
$$\sigma^2[\phi(dt;\alpha)] = c^2(N)\frac{1 - \exp\left(-\frac{2}{c^2(N)}\right)}{2} - \mathbb{E}[\phi(dt;\alpha)]^2. \tag{43}$$

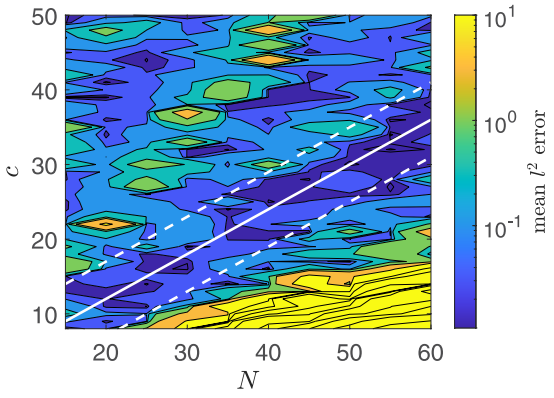
This leaves us with only one parameter $c = c(N)$ to be determined for the "optimal" estimation of the upper bound of $\mathscr{U}$. Here, the value of $c(N)$ is found based on a reference solution, say $\boldsymbol{u}_{ref}$ resulting from the integration of a stiff problem, whose solution profiles contain also sharp gradients.

Thus, in order to calibrate the hyperparameters of the scheme, *once and for all*, we have chosen as a reference solution the one resulting from the van der Pol (vdP) ODEs given by

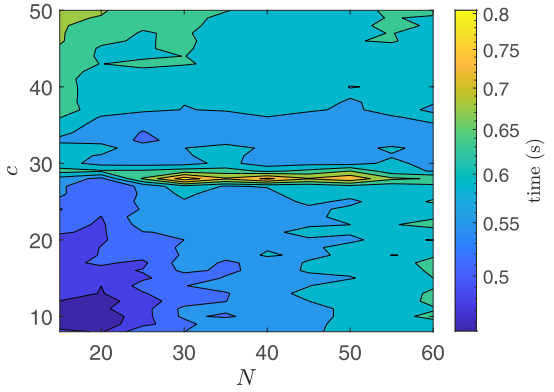
$$\frac{du_1}{dt} = u_2, \quad \frac{du_2}{dt} = \mu(1 - {u_1}^2)u_2 - u_1, \tag{44}$$

for $\mu = 100$ and $u_1(0) = 2$, $u_2(0) = 0$ as initial conditions; the time interval was set to $[0 \quad 3\mu]$, i.e., approximately three times the period of relaxation oscillations, which for $\mu \gg 1$, is $T \approx \mu(3 - 2\ln 2)$. The particular choice of $\mu = 100$ results to a stiff problem, containing also very sharp gradients resembling approximately a discontinuity in the solution profile within the integration interval. The reference solution was obtained using the `ode15s` with absolute and relative error tolerances set to 1×10^{-14}. In order to estimate the optimal values (c, N) (while we fixed $n = 20$), we computed the bias (B)-variance(V) trade-off loss $\mathscr{L}_{BV}$ function for u_2 (whose amplitude for the particular setting is about 75 times bigger than the amplitude of u_1) using 600 000 equidistant points t_k in $[0 \quad 3\mu]$ and running each PIRPNN configuration 100 times. Thus, the B-V trade-off loss is given by

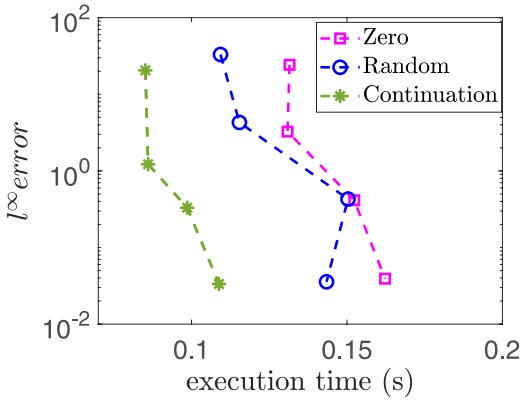
$$\mathscr{L}_{BV} = (B(\Psi_{N2}))^2 + V(\Psi_{N2}) = \mathbb{E}_\alpha((\Psi_{N2} - u_{ref,2})^2),$$
$$B(\Psi_{N2}) = \mathbb{E}_\alpha\left(\sum_{k=1}(\Psi_{N2}(t_k, \alpha, \boldsymbol{w}_2^o) - u_{ref,2}(t_k))\right),$$
$$V(\Psi_{N2}) = \mathbb{E}_\alpha\left(\sum_{k=1}(\Psi_{N2}(t_k, \alpha, \boldsymbol{w}_2^o) - u_{ref,2}(t_k))^2\right), \tag{45}$$



(a)



(b)



(c)

FIG. 1. Numerical solution of the van der Pol ODEs (44) with $\mu = 100$ using the PIRPNN in the interval $[0 \quad 3\mu]$ with respect to c and N for $n = 20$; *RelTol* and *AbsTol* were set to 1×10^{-06}. (a) Bias-Variance trade-off loss (45) with respect to the reference solution as obtained with `ode15s` with *AbsTol* and *RelTol* set to 1×10^{-14}. (b) Computational times (s). (c) Numerical approximation accuracy (indicatively l^∞ error for u_2) vs execution times with the proposed continuation method (green) and without continuation [thus initializing all weights to zero (magenta) or randomly (blue)].

where expectation is estimated over the 100 runs. Based on the above, the parsimonious selection of the values of the variables N and c giving the best numerical accuracy and minimum computational cost are $N = 20$, $c = 12$ [see Figs. 1(a) and 1(b)]. We note that the above parsimonious optimal values are fixed once and for all for all benchmark problems considered here.

Finally, in order to demonstrate the efficiency of the proposed continuation approach, in Fig. 1(c), we illustrate the l^∞ numerical approximation accuracy (indicatively for u_2) with respect to the reference solution vs the required execution times with and without (thus setting all weights to zero or randomly as initial guesses for Newton iterations) the proposed continuation method. As shown, the implementation of the proposed continuation scheme results in significantly better performances.

IV. NUMERICAL IMPLEMENTATION AND RESULTS

We implemented the proposed numerical scheme in MATLAB 2020b. Numerical results were obtained running on an Intel Xeon Gold 6240R CPU @2.40GHz frequency and 35.75 MB of cache. Each execution is a single-thread; thus, parallel encoding is not used. The `For-loop` for the formation of the Jacobian matrix $\nabla_{w^o} \boldsymbol{F}$ was implemented via a MEX file calling a C function and the Moore–Penrose pseudoinverse was computed with the MATLAB built-in function `pinv`, with default tolerance. In all our computations, we have used a fixed number of collocation points $n = 20$ and a number of basis functions $N = 20$ with $c = 12$ as discussed above. We note that a different choice of n would result in different values of c, thus affecting step-size adaptation.

For assessing the performance of the proposed scheme, we considered eight benchmark problems. In particular, we considered three index-1 DAEs: the Robertson[57,58] model of chemical kinetics, a non-autonomous model of mechanics,[58] and a non-autonomous power discharge control model,[58] and five stiff systems of ODEs: the Prothero–Robinson, the van der Pol model, the Hindmarsh–Rose neuronal model, the Belousov–Zhabotinsky chemical model,[60,61] and the Allen–Chan metastable PDE phase-field model[62,63] discretized in space with central FD. The performance of the proposed scheme was compared against two adaptive step-size solvers of the MATLAB ODE suite,[4] namely, `ode15s` and `ode23t`, appropriate for stiff ODEs and index-1 DAEs, thus using the analytical Jacobian. Moreover, we compared the performance of the scheme with a deep learning PINN as implemented in the DeepXDE library for scientific machine learning and physics-informed learning[40] for the solution of the Lotka–Volterra ODEs included in the demos of the library.

In order to estimate the numerical approximation error, we used as reference solution the one computed with `ode15s` setting the relative and absolute tolerances to 1×10^{-14} and 1×10^{-16}, respectively. To this aim, we computed l^2 and l^∞ norms of approximation errors, and the mean absolute approximation error (MAE), between the computed solutions using various schemes and the reference solutions using grids of say M equidistant points in the time intervals of interest. In the following, we report the aforementioned error metrics, $||\boldsymbol{\varepsilon}||_{l^2} = \sqrt{\sum_{j=1}^{M} \varepsilon_j{}^2}$, $||\boldsymbol{\varepsilon}||_{l^\infty} = \max_{j=1}^{M} |\varepsilon_j|$, $MAE = \frac{1}{M}\sqrt{\sum_{j=1}^{M} |\varepsilon_j|}$, for the component of the solution for which the absolute error was maximum. Finally, we ran each solver for a wide range of relative tolerances *reltol*, thus setting the absolute tolerances $abstol = reltol \cdot 10^{-3}$. For each case, we ran ODE solvers for 30 times and computed the median, maximum, and minimum computational times. We note that a direct comparison of `ode15s` and `ode23t` solvers and our scheme, based only on the relative and absolute tolerances (that is fixing them and check which one results in the best numerical approximation accuracy), cannot be done as these tolerances/convergence errors for `ode15s` and `ode23t` are *at the level of the solution*, while for our PIRPNN scheme, they are *at the level of the differential operator/first derivative*.

Finally, we underline that the proposed PIRPNN scheme provides an approximate solution in the form of an analytical function that can be evaluated explicitly at any point in the interval, while with the `odesuite` solvers, for the evaluation of the solution at any point in the interval, one needs to resort to the computationally expensive interpolation (as implemented by the function `deval`[83] in particular). Thus, if one needs to evaluate the solution in a *dense grid* of equidistant points in order to perform tasks such as the analysis of chaotic and quasiperiodic dynamics using, for example, FFT, the computational cost can be considerably high. Therefore, the comparison between solvers was made both on the grid of points resulting from the corresponding adaptive step-size methods and on dense grids of equidistant points.

A. Case study 1: Prothero–Robinson problem

Our first problem is the Prothero–Robinson stiff ODE benchmark problem[3,84] given by

$$\frac{du}{dt} = \lambda(u - \phi(t)) + \phi'(t), \quad \lambda < 0. \tag{46}$$

Its analytical solution is $u(t) = \phi(t)$. The problem becomes stiff for $\lambda \ll -1$. For our numerical simulations, we chose $\phi(t) = sin(t)$, $u(0) = \phi(0) = sin(0) = 0$, and $[0, 2\pi]$ as the time interval where the solution is sought, while the parameter λ controlling the stiffness is set equal to -1×10^5.

Note that although the analytical solution is simple and smooth, i.e., does not exhibit any steep gradient, the problem is very difficult to solve with a non-stiff solver (for example, using the ode45 Matlab solver employing the adaptive Dormand–Prince scheme).

Figures 2(a)–2(f) depict l^2, l^∞, and mean absolute (MAE) numerical approximation accuracy with respect to the analytical solution vs required computational times. Figures 2(a)–2(c) depict the computational times of the corresponding adaptive step-size solution procedure. Figures 2(d)–2(f) depict the computational times of various solvers when the solution is sought in a grid of 10 000 equidistant points in $[0 \quad 2\pi]$. Finally, Figs. 3(a) and 3(b) depict the l^2 numerical approximation accuracy (indicatively for u_2) with respect to the reference solution vs the number of adaptive steps [Fig. 3(a)], the number of function evaluations [Fig. 3(b)].

As it is shown, the PIRPNN outperforms `ode23t` in all metrics, while for all practical purposes, its performance is equivalent to the `ode15s` with respect to computational times resulting from the corresponding adaptive step-size solution procedure. Besides, when the solution is sought in the dense grid of equidistant points, the computational times resulting from the implementation of the `odesuite` solvers are much larger than the ones resulting from

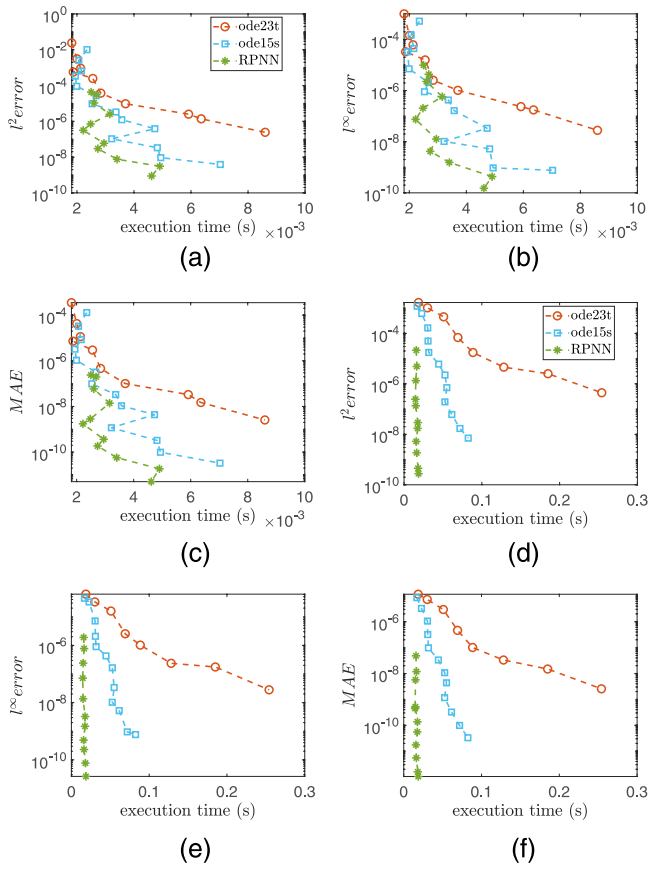


FIG. 2. The Prothero–Robinson[84] benchmark stiff ODE problem with $\lambda = -1 \times 10^5$, see Eq. (46). (a)–(c) l^2, l^∞, and mean absolute (MAE) numerical approximation errors with respect to the analytical solution $u(t) = \sin(t)$ vs execution times (s) of various schemes using adaptation. (d)–(f) l^2, l^∞, and mean absolute (MAE) numerical approximation errors with respect to the analytical solution vs execution times (s) when the solution is sought *in a grid of 10 000 equidistant points* in [0 2π] times (s).

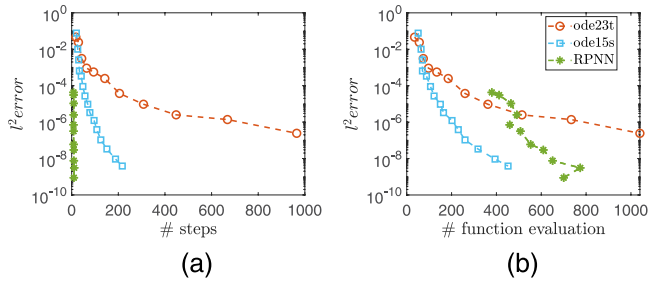


FIG. 3. The Prothero–Robinson[84] stiff ODE with $\lambda = -1 \times 10^5$, see Eq. (46). l^2 numerical approximation error vs (a) the number of adaptive steps and (b) the number of function evaluations.

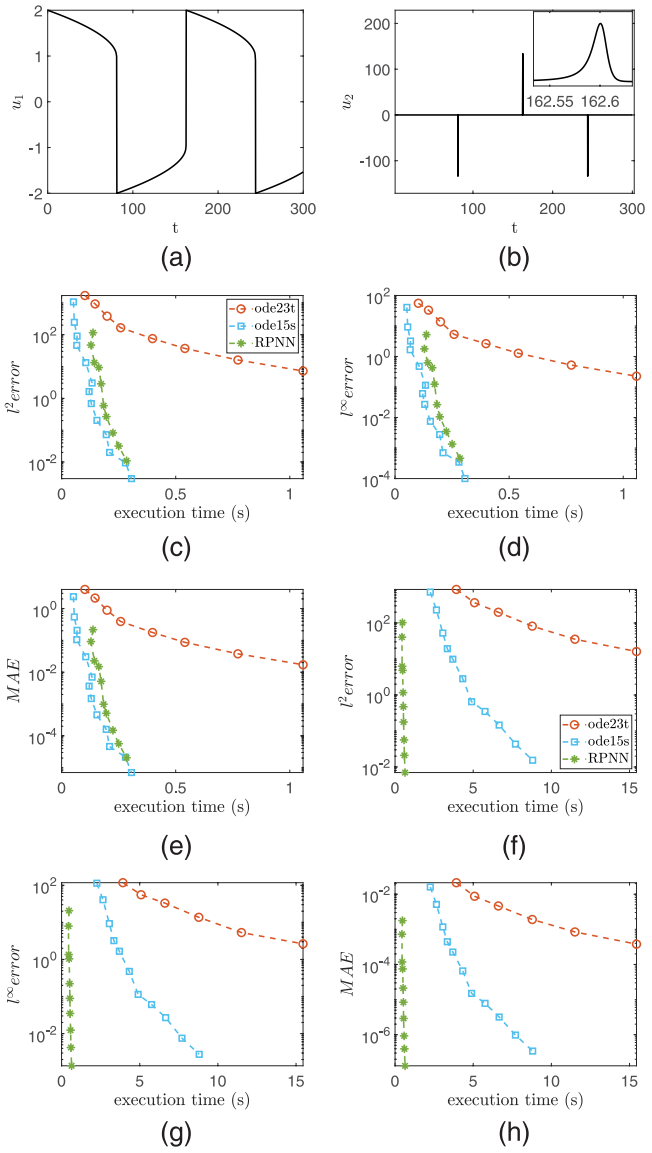


FIG. 4. The vdP[66] ODEs with $\mu = 100$, see Eq. (44). The numerical reference solution is obtained in the time interval [0 3μ] with `ode15s` with relative and absolute tolerances set to 1×10^{-14} and 1×10^{-16}, respectively. (a) Reference solution for u_1 and (b) reference solution for u_2 with a zoom close to a steep gradient. (c)–(e) l^2, l^∞, and mean absolute (MAE) numerical approximation errors (indicatively for u_2) with respect to the reference solution vs execution times (s) of the corresponding *adaptive step-size solution procedure*. (f)–(h) l^2, l^∞, and mean absolute (MAE) numerical approximation errors (indicatively for u_2) with respect to the reference solution vs execution times (s) when the solution is sought *in a grid of 600 000 logarithmically equidistant points* in [0 $4 \cdot 3\mu$] times (s).

the implementation of the proposed PIRPNN scheme. As it is also shown in Fig. 3, our scheme, compared to both `ode15s` and `ode23t`, is more efficient in terms of the number of adaptive steps needed to compute the solution, while it needs more function evaluations than `ode15s` and less than `ode23t`.

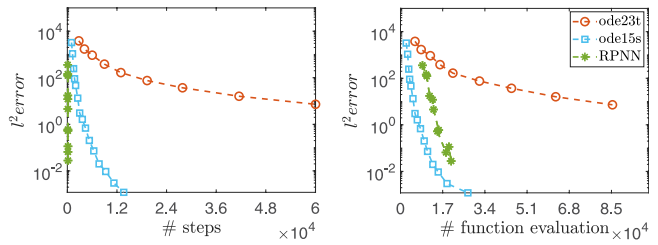


FIG. 5. The vdP[66] ODEs with $\mu = 100$, see Eq. (44). l^2 numerical approximation error (indicatively for u_2) vs (a) the number of adaptive steps and (b) the number of function evaluations.

B. Case study 2: The van der Pol model

Our second benchmark problem is the stiff van der Pol system of ODEs (44) introduced in Sec. III. Figures 4(a) and 4(b) depict the reference solution profiles for u_1, u_2, with $\mu = 100$ in the time interval $[0 \quad 3\mu]$ as obtained with `ode15s` with the relative and absolute tolerances set to 1×10^{-14} and 1×10^{-16}, respectively. The relaxation oscillations of the vdP model exhibit both very sharp gradients resembling discontinuities and stiffness.[66] Figures 4(c)–4(h) depict the l^2, l^∞ and mean absolute (MAE) numerical approximation accuracy (indicatively for u_2) with respect to the reference solution vs the required computational times. Figures 4(c)–4(e) depict the computational times of the corresponding adaptive step-size solution procedure; upon convergence, the approximation errors are computed based on a uniform grid of 600 000 points in $[0 \quad 3\mu]$. Figures 4(f)–4(h) depict the computational times of the various solvers when the solution is sought in a grid of 600,000 equidistant points (such a number of points is required in order to appropriately trace uniformly the solution in the interval of interest due to the presence of very steep gradients). Finally, Figs. 5(a) and 5(b) depict the l^2 numerical approximation accuracy (indicatively for u_2) with respect to the reference solution vs the number of adaptive steps [Fig. 5(a)], the number of function evaluations [Fig. 5(b)].

As it is shown, the PIRPNN outperforms `ode23t` in all metrics, while for all practical purposes its performance is equivalent to the `ode15s` with respect to the computational times resulting from the corresponding adaptive step-size solution procedure. Besides, when the solution is sought in the dense grid of equidistant points, the computational times resulting from the implementation of the `odesuite` solvers are much larger than the ones resulting from the implementation of the proposed PIRPNN scheme.

As it is shown, our scheme, compared to both `ode15s` and `ode23t`, is more efficient in terms of the number of adaptive steps needed to compute the solution, while it needs a comparable number of function evaluations with `ode15s` and significantly less than `ode23t`.

C. Case study 3: The Robertson index-1 DAEs

The Robertson model describes the kinetics of an autocatalytic reaction.[57] This system of three DAEs is part of the benchmark problems considered in Ref. 58. The set of reactions reads

$$A \xrightarrow{k_1} B, \quad B + C \xrightarrow{k_2} A + C, \quad 2B \xrightarrow{k_3} B + C, \tag{47}$$

where A, B, and C are chemical species and $k_1 = 0.04$, $k_2 = 10^4$, and $k_3 = 3 \times 10^7$ are reaction rate constants. Assuming that the total mass of the system is conserved, we have the following system of index-1 DAEs:

$$\frac{dA}{dt} = -k_1 A + k_2 BC, \quad \frac{dB}{dt} = +k_1 A - k_2 BC - k_3 B^2, \quad A + B + C = 1, \tag{48}$$

where A, B, and C denote the concentrations of $[A]$, $[B]$, and $[C]$, respectively. In our simulations, we set $A(0) = 1$, $B(0) = 0$ as initial conditions of concentrations, and we solve in the time interval $[0 \quad 4 \times 10^6]$ as in Ref. 85.

Figures 6(a) and 6(b) show the solution profiles of A, B, and C as obtained with `ode15s` with relative and absolute tolerances set to 1×10^{-14} and 1×10^{-16}, respectively. Figures 6(c)–6(h) depict the l^2, l^∞, and mean absolute (MAE) numerical approximation errors (indicatively for A) upon convergence of the corresponding adaptive step-size procedure, with respect to the reference solution using 40 000 logarithmically equidistant points in the interval $[0 \quad 4 \times 10^6]$. Figures 6(c)–6(e) depict the computational times of the corresponding adaptive step-size solution procedure, while Figs. 6(f)–6(h) depict the computational times required when the solution is sought in a grid of 40 000 logarithmically equidistant points in the interval $[0 \quad 4 \cdot 10^6]$.

Finally, Figs. 7(a) and 7(b) depict the l^2 numerical approximation accuracy (indicatively for A) with respect to the reference solution vs the number of adaptive steps [Fig. 7(a)], the number of function evaluations [Fig. 7(b)].

As it is shown, the proposed PIRPNN scheme outperforms `ode23t` in all metrics, but compared with the `ode15s`, the computational times resulting from the adaptive step-size solution procedure are larger. However, when the solution is sought in the denser grid of points, the performance of the PIRPNN scheme is equivalent to one of the `ode15s` solver. As it is shown, our scheme is comparable to `ode15s` and significantly more efficient than `ode23t` in terms of number of adaptive steps needed to compute the solution, while it needs a bigger number of function evaluations than `ode15s` and significantly less than `ode23t`.

D. Case study 4: Hindmarsh–Rose neuronal model

This is a system of three ODEs that study the spiking-bursting behavior of the membrane potential of a neuron.[59] The equations are as follows:

$$\frac{dx}{dt} = y - x^3 + 3x - z + I, \quad \frac{dy}{dt} = 1 - 5x^2 - y, \quad \frac{dz}{dt} = r[4(x + 1.6) - z], \tag{49}$$

where for $r = 0.005$ and $I = 3.25$, the behavior is chaotic. We select as initial conditions $x(0) = -1$, $y(0) = -3.5$, and $z(0) = 3$.

Figures 8(a) and 8(b) illustrate the reference solution profiles for (indicatively) x, y in the interval $[0 \quad 1000]$ as obtained with `ode15s` with both relative and absolute tolerances set to 1×10^{-14}

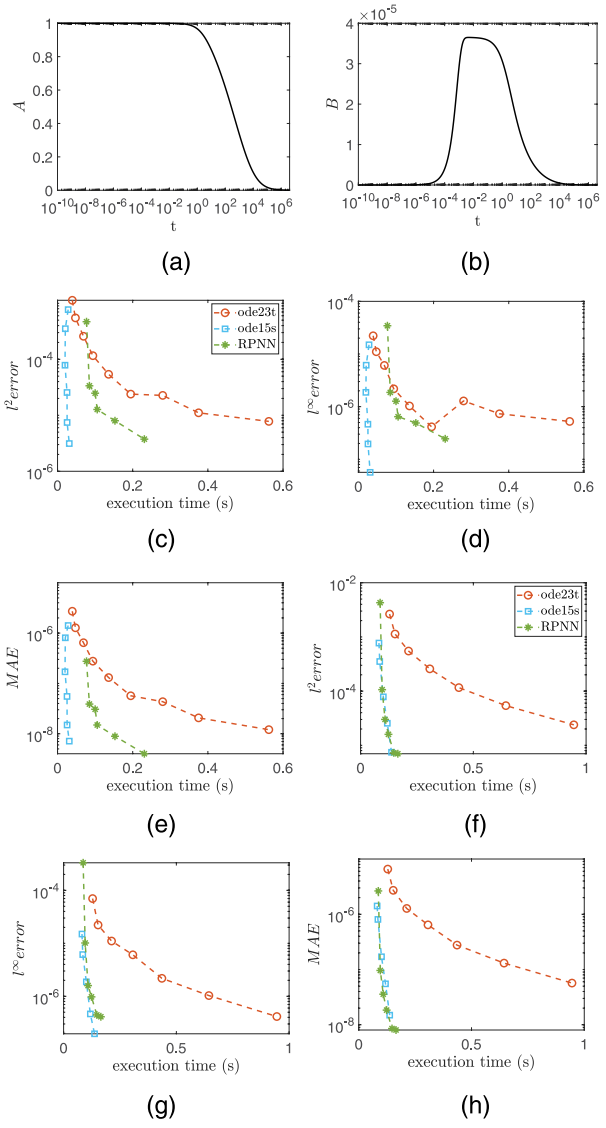


FIG. 6. The Robertson[58] index-1 DAEs, see Eq. (48). The reference solution is obtained in the time interval [0 $4 \cdot 10^6$] with `ode15s` with relative and absolute tolerances set to 1×10^{-14} and 1×10^{-16}, respectively. (a) and (b) Reference solution profiles (indicatively for A, B). (c)–(e) l^2, l^∞ and mean absolute (MAE) numerical approximation errors (indicatively for A) with respect to the reference solution vs execution times (s) of the corresponding *adaptive step-size solution procedure*. (f)–(h) l^2, l^∞, and mean absolute (MAE) numerical approximation errors (indicatively for A) with respect to the reference solution vs execution times (s) when the solution is sought *in a grid of 40 000 logarithmically equidistant points* in [0 4×10^6].

and 1×10^{-16}, respectively. Figures 8(c)–8(h) depict the l^2, l^∞, and mean absolute (MAE) errors (indicatively for y) upon convergence of the corresponding adaptive step-size procedure with respect to the reference solution using 40, 000 equidistant grid points in the interval [0 1000]. Figures 8(c)–8(e) depict the computational times

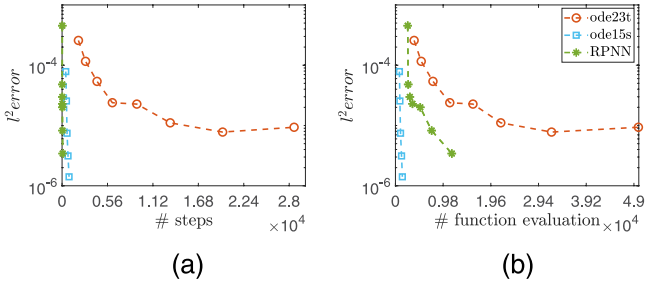


FIG. 7. The Robertson[58] DAEs, see Eq. (48). l^2 numerical approximation error (indicatively for A) vs (a) the number of adaptive steps and (b) the number of function evaluations.

of the adaptive step-size solution procedure, while Figs. 8(f)–8(h) depict the computational times required when the solution is sought in the dense grid of points.

Finally, Figs. 9(a) and 9(b) depict the l^2 numerical approximation accuracy (indicatively for y) with respect to the reference solution vs the number of adaptive steps [Fig. 9(a)], the number of function evaluations [Fig. 9(b)].

As it is shown, the proposed PIRPNN outperforms the `ode23t` solver in all metrics, while for all practical purposes, its performance is comparable to one of the `ode15s` solver. When the solution is sought in the mesh of 40,000 equidistant points, the PIRPNN scheme also outperforms `ode15s`.

To this end, we note that the solution of the particular DAEs' problem exhibits both stiffness and steep gradients as the vdP model where the PIRPNN outperforms both `ode15s` and `ode23t`. As it is also shown, our scheme, compared to both `ode15s` and `ode23t`, is more efficient in terms of number of adaptive steps needed to compute the solution, while it needs more function evaluations than `ode15s` and significantly less than `ode23t`.

E. Case study 5: A mechanics non autonomous index-1 DAEs model

This is a non-autonomous system of five index-1 DAEs and it is part of the benchmark problems presented in Ref. 58. It describes the motion of a bead on a rotating needle subject to the gravity, friction, and centrifugal forces. The equations of motion are as follows:

$$\begin{aligned} \frac{du_1}{dt} &= u_2, \quad \frac{du_2}{dt} = -10u_2 + \sin(t + pi/4)u_5, \\ \frac{du_3}{dt} &= u_4, \quad \frac{du_4}{dt} = -10u_4 - \cos(t + pi/4)u_5 + 1, \\ 0 &= g_{pp} + 20g_p + 100g, \end{aligned} \tag{50}$$

where $g = \cos(t + pi/4)u_3 - \sin(t + pi/4)u_1$, $g_p = \cos(t + pi/4)(u_4 - u_1) + \sin(t + pi/4)(-u - 2 - u_3)$, and $g_{pp} = \cos(t + pi/4)(\frac{du_4}{dt} - \frac{du_1}{dt} - u_2 - u_3 + \sin(t + pi/4)(-\frac{du_2}{dt} - \frac{du_3}{dt} - u_4 + u_1)$. The initial conditions are $u_1(0) = 1$, $u_2(0) = -6$, $u_3(0) = 1$, and $u_4(0) = -6$ and a consistent initial condition for $u_5(0) = -10.606\,601\,717\,798\,20$ was found with Newton–Raphson at $t = 0$, with a tolerance of 1×10^{-16}.

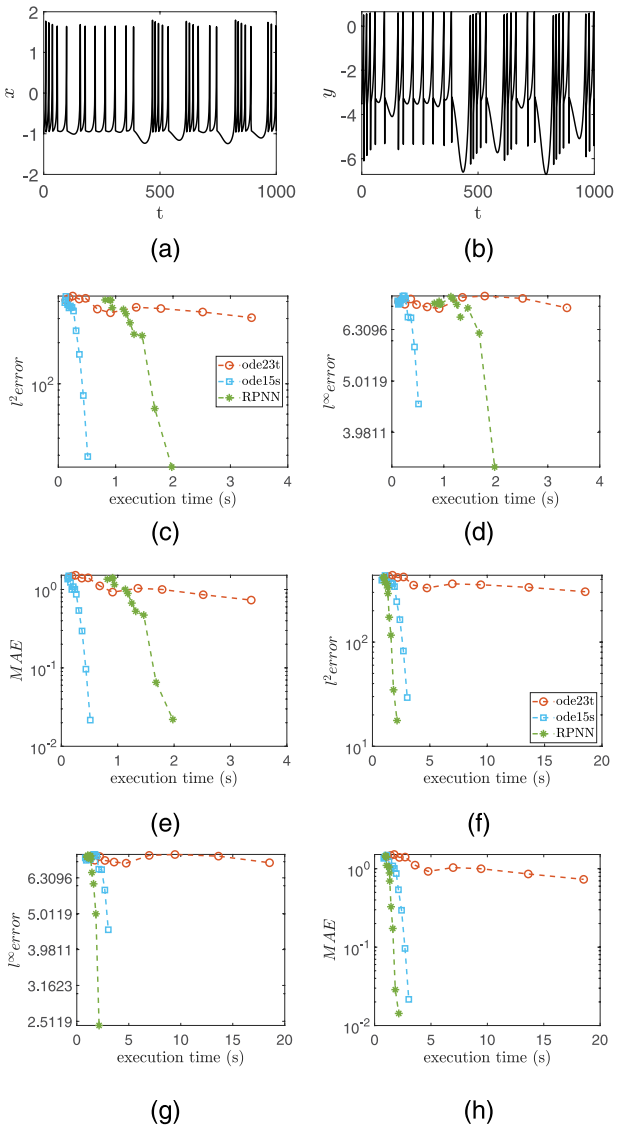


FIG. 8. Hindmarsh–Rose[59] model with $r = 0.005$ and $I = 3.25$, see Eq. (49). The numerical reference solution is obtained in the interval [0 1000] with `ode15s` with relative and absolute tolerances set to 1×10^{-14} and 1×10^{-16}, respectively. (a)–(b) Reference profiles for (indicatively) x, y. (c)–(e) l^2, l^∞, and mean absolute (MAE) numerical approximation errors (indicatively for y) with respect to the reference solution vs execution times (s) of the corresponding *adaptive step-size solution procedure*. (f)–(h) l^2, l^∞, and mean absolute (MAE) numerical approximation errors (indicatively for y) with respect to the reference solution vs execution times (s) when the solution is sought *in a grid of 40 000 equidistant points*.

Figures 10(a)–10(d) illustrate the reference solution profiles for (indicatively) u_1, u_2, u_3, and u_5 in the interval [0 15] as obtained with `ode15s` with both relative and absolute tolerances set to 1×10^{-14} and 1×10^{-16}, respectively. Figures 10(e)–10(j) depict the l^2, l^∞, and mean absolute (MAE) errors (indicatively for u_5)

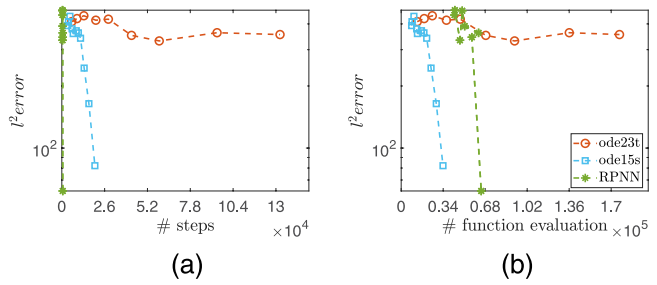


FIG. 9. The Hindmarsh–Rose[59] model with $r = 0.005$ and $I = 3.25$, see Eq. (49). l^2 numerical approximation error (indicatively for y) vs (a) the number of adaptive steps and (b) the number of function evaluations.

upon convergence of the corresponding adaptive step-size procedure, with respect to the reference solution using 15 000 equidistant grid points in the interval [0 15]. Figures 10(e)–10(g) depict the computational times of the adaptive step-size solution procedure, while Figs. 10(h)–10(j) depict the computational times required when the solution is sought in the dense grid of points. Finally, Figs. 11(a) and 11(b) depict the l^2 numerical approximation accuracy (indicatively for u_5) with respect to the reference solution vs the number of adaptive steps [Fig. 11(a)], the number of function evaluation [Fig. 11(b)].

As it is shown, the proposed PIRPNN outperforms the `ode23t` solver in all metrics, while its performance is for all practical purposes equivalent to one of the `ode15s` solver, while when the solution is sought in the mesh of 15 000 equidistant points, the PIRPNN scheme also outperforms `ode15s`.

To this end, we note, that the solution of the particular DAEs problem exhibits both stiffness and steep gradients as the vdP model where the PIRPNN outperforms both `ode15s` and `ode23t`. As it is shown, our scheme is comparable with `ode15s` and significantly more efficient than `ode23t` in terms of number of adaptive steps needed to compute the solution, while it needs a comparable number of function evaluations with `ode15s` and significantly less than `ode23t`.

F. Case study 6: Power discharge control index-1 DAEs problem

This is a non-autonomous model of six index-1 DAEs, and it is part of the benchmark problems considered in Ref. 58. The governing equations are as follows:

$$\begin{aligned}
&\frac{du_1}{dt} = \frac{(u_2 - u_1)}{20}, \quad \frac{du_2}{dt} = -\frac{(u_4 - 99.1)}{75}, \\
&\frac{du_3}{dt} = \mu - u_6, \quad 0 = 20u_5 - u_3, \\
&0 = (3.35 - 0.075u_6 + 0.001u_6^2) - \frac{u_4}{u_5}, \\
&0 = \frac{u_3}{400}\frac{du_3}{dt} + \frac{\mu\mu_p}{(1.2u_1)^2} - \frac{du_1}{dt}\frac{\mu^2}{(1.44u_1)^3}, \\
&\mu = 15 + 5\tanh(t - 10), \quad \mu_p = \frac{5}{\cosh^2(t - 10)}.
\end{aligned} \tag{51}$$

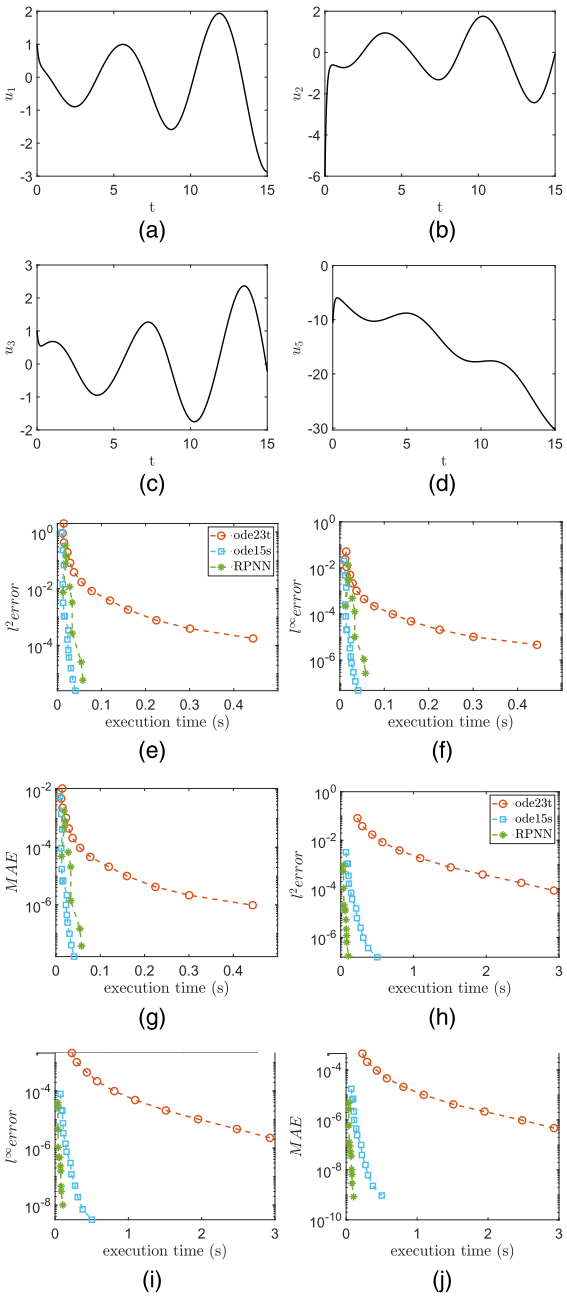


FIG. 10. The mechanics problem of non-autonomous index-1 DAEs,[58] see Eq. (50) . The numerical reference solution is obtained in the interval [0 15] with `ode15s` with relative and absolute tolerances set to 1×10^{-14} and 1×10^{-16}, respectively. (a)–(d) Reference profiles for u_1, u_2, u_4, and u_5. (e)–(g) l^2, l^∞, and mean absolute (MAE) numerical approximation errors (indicatively for u_5) with respect to the reference solution vs execution times (s) of the corresponding *adaptive step-size solution procedure*. (h)–(j) l^2, l^∞, and mean absolute (MAE) numerical approximation errors (indicatively for u_5) with respect to the reference solution vs execution times (s) when the solution is sought *in a grid of 15 000 equidistant points*.

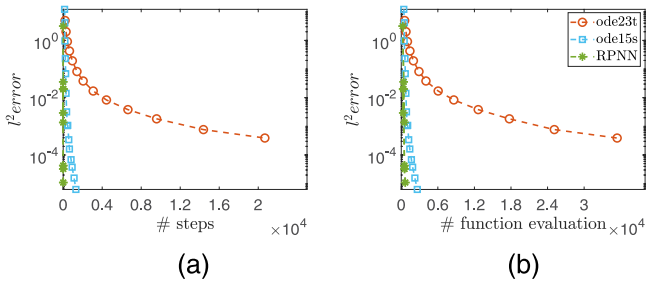


FIG. 11. The mechanics problem of non-autonomous index-1 DAEs,[58] see Eq. (50). l^2 numerical approximation error (indicatively for u_5) vs (a) the number of adaptive steps and (b) the number of function evaluations.

The initial conditions are $u_1(0) = u_2(0) = 0.25$, $u_3(0) = 734$, and consistent initial conditions ($u_4(0) = 99.089\,994\,920\,02$, $u_5(0) = 36.7$ and $u_6(0) = 10.000\,002\,516\,71$) were found with Newton–Raphson with a tolerance of 1×10^{-16}.

Figures 12(a)–12(d) depict the reference solution profiles for u_1, u_2, u_3, and u_4 as obtained with `ode15s` with both relative and absolute tolerances set to 1×10^{-14} and 1×10^{-16}, respectively. Similarly, to the Robertson model, the solution profiles do not exhibit very steep gradients. Figures 12(e)–12(j) depict the l^2, l^∞, and mean absolute (MAE) approximation errors (indicatively for u_3) upon convergence of the corresponding adaptive step-size procedure, with respect to the reference solution on the basis of 40, 000 equidistant points in the interval [0 40]. Figures 12(e)–12(g) depict the computational times of the corresponding adaptive step-size procedure, while Figs. 12(h)–12(j) depict the required computational times when the solution is sought in a grid of 40 000 equidistant points.

Furthermore, Figs. 13(a) and 13(b) depict the l^2 numerical approximation accuracy (indicatively for u_3) with respect to the reference solution vs the number of adaptive steps [Fig. 13(a)] and the number of function evaluations [Fig. 13(b)].

As it is shown, the proposed PIRPNN scheme outperforms the `ode23t` solver in all metrics for higher numerical approximation accuracy, while the best performance with respect to the corresponding adaptive step-size procedure is the one obtained with the `ode15s` solver. However, when the solution is sought in the grid of 40 000 equidistant points, the PIRPNN outperforms `ode15s`. Finally, our scheme is comparable to `ode15s` and significantly more efficient than `ode23t` in terms of the number of adaptive steps needed to compute the solution and is comparable to `ode15s` and significantly more efficient than `ode23t` regarding the number of function evaluations.

G. Case study 7: The Belousov–Zhabotinsky stiff ODEs

The Belousov–Zhabotinsky chemical reactions model is given by the following system of seven ODEs:[61,86]

$$\frac{dA}{dt} = -k_1 AY, \quad \frac{dY}{dt} = -k_1 AY - k_2 XY + k_5 Z,$$
$$\frac{dX}{dt} = k_1 AY - k_2 XY + k_3 BX - 2k_4 X^2, \quad \frac{dP}{dt} = k_2 XY, \qquad (52)$$
$$\frac{dB}{dt} = -k_3 BX, \quad \frac{dZ}{dt} = k_3 BX - k_5 Z, \quad \frac{dQ}{dt} = k_4 X^2.$$

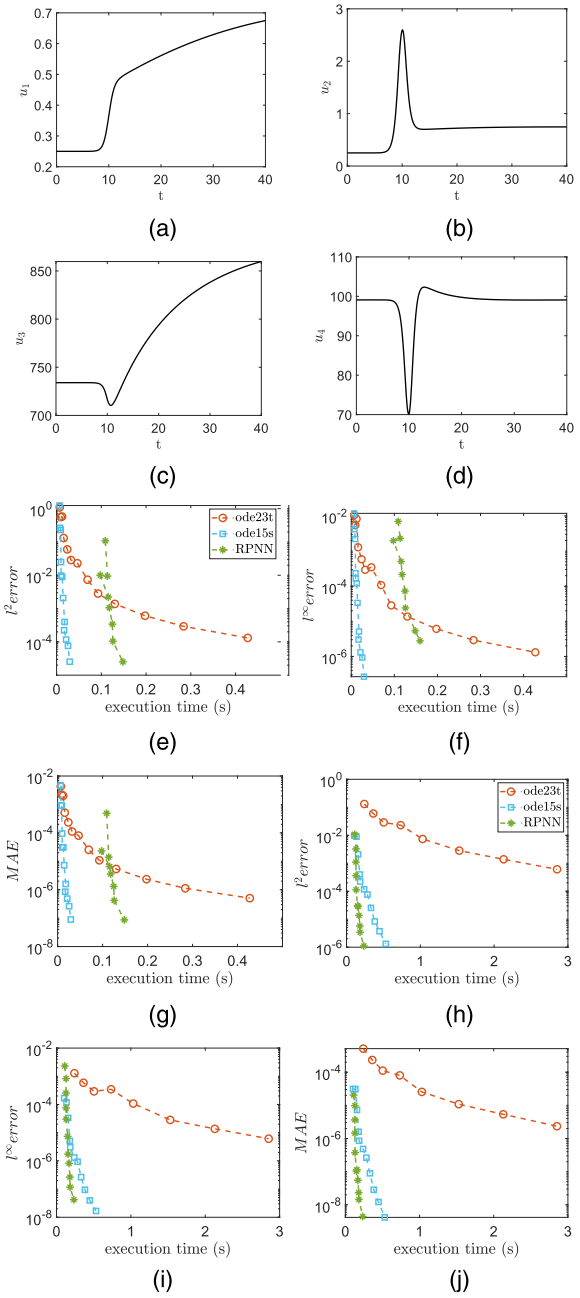


FIG. 12. Power discharge control non-autonomous index-1 DAEs problem,[58] see Eq. (51). The reference solution is obtained in the interval [0 40] with `ode15s` with relative and absolute tolerances set to 1×10^{-14} and 1×10^{-16}, respectively. (a)–(d) Reference profiles for u_1, u_2, u_3, *and*u_4. (e)–(g) l^2, l^∞, and mean absolute (MAE) numerical approximation errors (indicatively for u_3) with respect to the reference solution vs execution times (s) of the corresponding *adaptive step-size solution procedure*. (h)–(j) l^2, l^∞, and mean absolute (MAE) numerical approximation errors (indicatively for u_3) with respect to the reference solution vs execution times (s) when the solution is sought *in a grid of 40 000 equidistant points*.

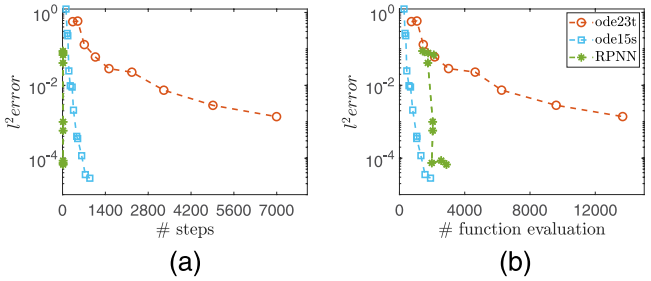


FIG. 13. Power discharge control non-autonomous index-1 DAEs problem,[58] see Eq. (51). l^2 numerical approximation error (indicatively for u_3) vs (a) the number of adaptive steps and (b) the number of function evaluations.

A, B, P, Q, X, Y, Z are the concentrations of chemical species and $k_1 = 4.72$, $k_2 = 3 \times 10^9$, $k_3 = 1.5 \times 10^4$, $k_4 = 4 \times 10^7$, and $k_5 = 1$ are the reaction coefficients. The initial conditions are set as $A(0) = B(0) = 0.066$, $Y(0) = X(0) = P(0) = Q(0) = 0$, $Z(0) = 0.002$.

Figures 14(a)–14(d) depict the solution profiles of A, Y, X, and B as obtained with `ode15s` with both relative and absolute tolerances set to 1×10^{-14} and 1×10^{-16}, respectively. As reported in Ref. 87 for an acceptable solution, one needs to set relative and absolute tolerances at least of 1×10^{-07}.

Figures 14(e)–14(j) depict the l^2, l^∞, and mean absolute (MAE) approximation errors (indicatively for B) upon convergence of the corresponding adaptive step-size procedure with respect to the reference solution in 80 000 equidistant points in the interval [0 40]. Figures 14(e)–14(g) depict the computational times of the corresponding adaptive step-size procedure, while Figs. 14(h)–14(j) depict the computational times required when the solution is sought in the uniform grid of 80 000 points in [0 40].

Furthermore, Figs. 15(a) and 15(b) depict the l^2 numerical approximation accuracy (indicatively for B) with respect to the reference solution vs the number of adaptive steps [Fig. 15(a)] and the number of function evaluations [Fig. 15(b)].

As shown, the proposed PIRPNN is less efficient than both `ode23t` and `ode15s`, when comparing the computational times based on the corresponding adaptive step-size procedure. Here, the relatively higher computational cost is due to the considerably bigger size of the Jacobian (here of size 140×140) formed by the proposed PIRPNN scheme at each Newton iteration, compared to the ones that `ode15s/ode23t` solvers process (here of size 7×7). Also, the component X of the solution has a very small amplitude (its maximum amplitude is less than 1×10^{-05}) while exhibits a very sharp gradient in its solution profile. Therefore, this may need a particular treatment/weighting of the corresponding residuals. However, considering also the computational times required when the solution is sought in a mesh of 80 000 equidistant points, the PIRPNN outperforms the `ode23t` solver, and it is for any practical purposes comparable to the `ode15s`. Finally, our scheme, compared to both `ode15s` and `ode23t`, is much more efficient in terms of the number of adaptive steps needed to compute the solution, while it needs more number of function evaluations than `ode15s` and less than `ode23t`.

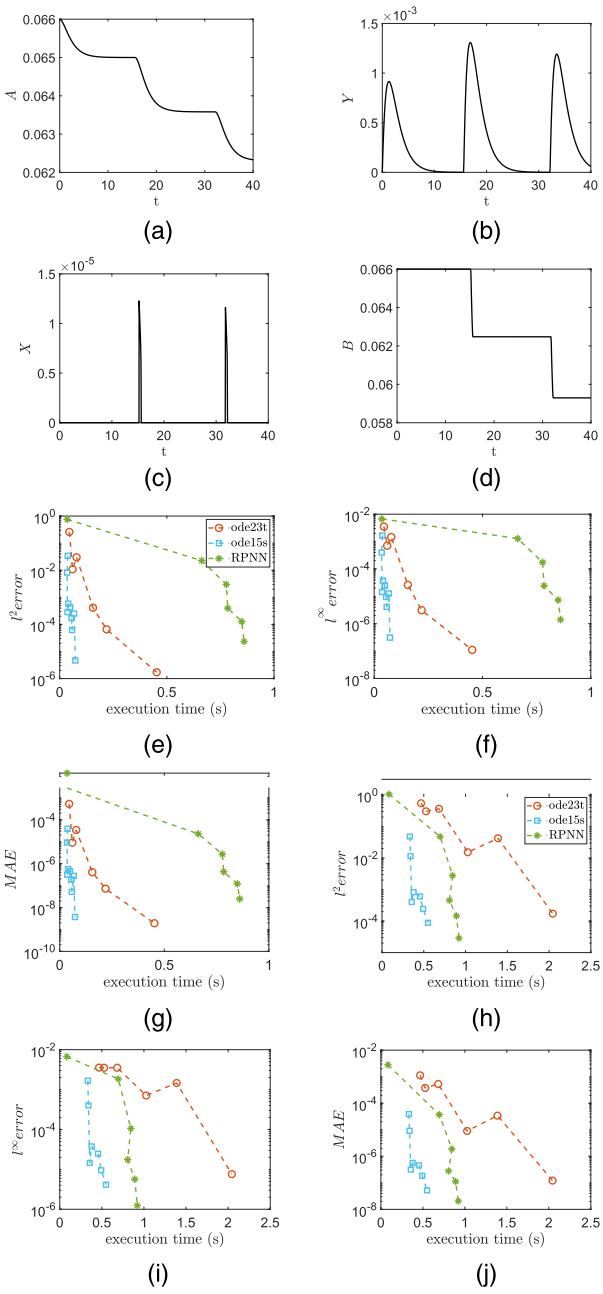


FIG. 14. The Belousov–Zhabotinsky[60,61] model, see Eq. (52). The numerical reference solution is obtained in the time interval [0 40] using the `ode15s` solver with relative and absolute tolerances set to 1×10^{-14} and 1×10^{-16}, respectively. (a)–(d) Reference solution profiles for (indicatively) A, Y, X, B. (e)–(g) l^2, l^∞, and mean absolute (MAE) numerical approximation errors (indicatively for B) with respect to the reference solution vs execution times (s) of the corresponding *adaptive step-size solution procedure*. (h)–(j) l^2, l^∞, and mean absolute (MAE) numerical approximation errors (indicatively for B) with respect to the reference solution vs execution times (s) when the solution is sought *in a grid of 80 000 equidistant points in* [0 40].

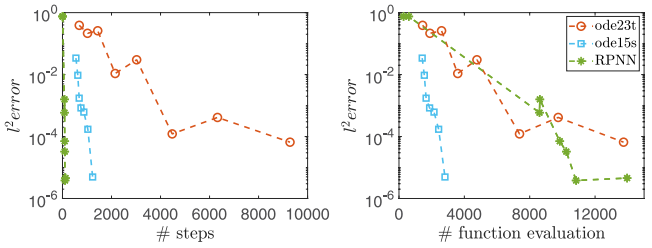


FIG. 15. The Belousov–Zhabotinsky[60,61] model, see Eq. (52). l^2 numerical approximation error (indicatively for B) vs (a) the number of adaptive steps and (b) the number of function evaluations.

H. Case study 8: The Allen–Cahn PDE phase-field model

The Allen–Cahn equation is a famous reaction-diffusion PDE that was proposed in Ref. 62 as a phase-field model for describing the dynamics of the mean curvature flow. Here, for our illustrations, we considered a one-dimensional formulation given by[63]

$$\begin{aligned} \frac{\partial u}{\partial t} &= \nu \frac{\partial^2 u}{\partial x^2} + u - u^3, \quad x \in [-1 \quad 1], \\ u(-1,t) &= -1, \quad u(1,t) = 1, \end{aligned} \tag{53}$$

with initial condition $u(x,0) = 0.53\,x + 0.47\,\sin(-1.5\,\pi\,x)$. Here, we integrate until $t = 70$. For $\nu = 0.01$, the solution is stiff,[63] thus exhibiting a metastable behavior with an initial two-hill configuration that disappears close to $t = 40$ with a fast transition to a one-hill stable solution, as depicted in Fig. 16(a). For our illustrations, we used an equally spaced grid of 102 points in space and second-order-centered finite differences. Hence, (53) becomes a system of 100 ODEs,

$$\begin{aligned} \frac{\partial u_i}{\partial t} &= \nu \frac{(u_{i+1} - 2u_i + u_{i-1})}{dx^2} + u_i - u_i^3, \\ u_0 &= -1, \quad u_{101} = 1. \end{aligned} \tag{54}$$

Here, for the implementation of the PIRPNN, we have used a sparse QR decomposition as implemented in the SuiteSparseQR.[56,80] Figures 16(b) depict the absolute numerical approximation error when using the PIRPNN for relative and absolute tolerances set to 1×10^{-03} and 1×10^{-06}, respectively. Figures 16(c)–16(h) depict the l^2, l^∞, and mean absolute (MAE) numerical approximation errors upon convergence of the corresponding adaptive step-size procedure with respect to the reference solution in 70 000 × 102 equidistant points in the time [0 70] and in the space interval [−1 1], respectively. Figures 16(c)–16(e) depict the computational times of the corresponding adaptive step-size procedure, while Figs. 14(f)–14(h) depict the computational times required when the solution is sought in the uniform spatiotemporal grid of 102 × 70 000 points.

Furthermore, Figs. 17(a) and 17(b) depict the l^2 numerical approximation accuracy with respect to the reference solution vs

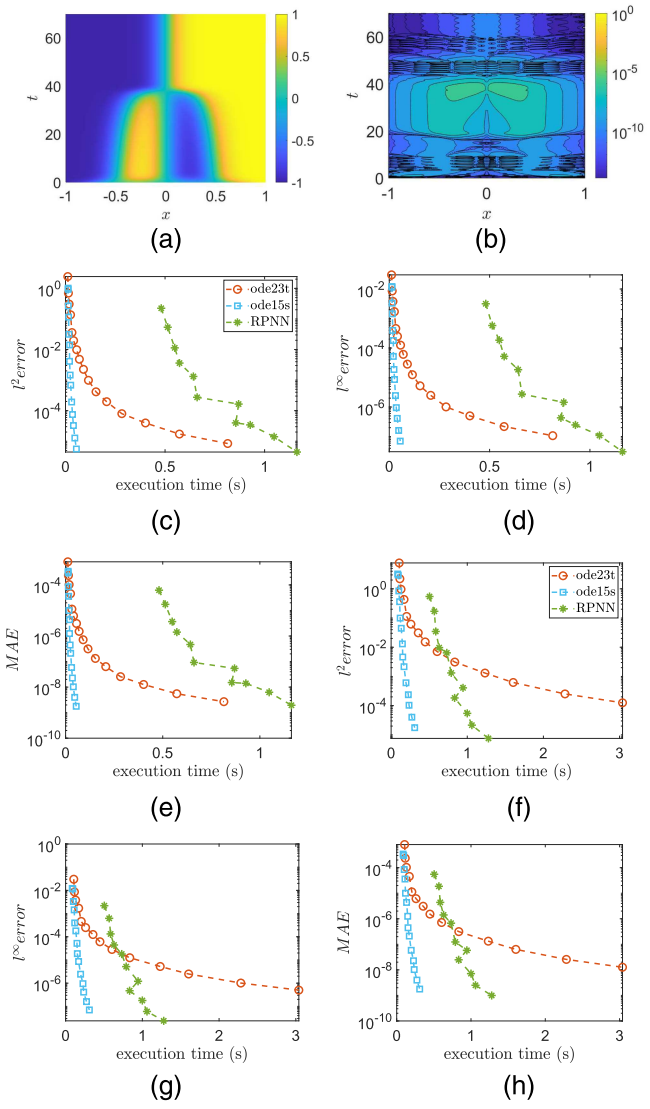


FIG. 16. Allen–Cahn PDE[63] discretized in space with central FD [see Eq. (54)]. The reference solution is obtained for $\nu = 0.01$ in the time interval [0 70] with the `ode15s` solver, with relative and absolute tolerances set to 1×10^{-14} and 1×10^{-16}, respectively. (a) Contour plot of the reference solution. (b) Contour plot of absolute approximation error computed with the PIRPNN using 1×10^{-03} for the relative tolerance and 1×10^{-06} for the absolute tolerance. (c)–(e) l^2, l^∞, MAE approximation errors with respect to the reference solution vs execution times (s) of the corresponding *adaptive step-size procedure*. (f)–(h) l^2, l^∞, and MAE numerical approximation errors with respect to the reference solution vs execution times (s) when the solution is sought *in the spatiotemporal grid of* $102 \times 70\,000$ *equidistant points*.

(a) the number of adaptive steps and (b) the number of function evaluations.

As shown, the proposed PIRPNN scheme is less efficient than both `ode23t` and `ode15s` when considering the computational times resulting from the corresponding adaptive step-size procedure. However, when considering the computational times required when the solution is sought in the mesh of 70 000 equidistant points in time, the PIRPNN outperforms `ode23t` but still is less efficient than `ode15s`. However, our proposed scheme, compared to both `ode15s` and `ode23t`, is more efficient in terms of number of adaptive steps needed to compute the solution, while it needs more number of function evaluations with `ode15s` and significantly less than `ode23t`. The relatively higher computational cost is due to the considerable bigger size of the Jacobian required by the proposed scheme at each Newton iteration (here of size 2000×2000) compared to the Jacobian processed by `ode15s/ode23t` (here of size 100×100). Thus, to speed up the computations in a subsequent work, we aim in a future work at exploiting the arsenal of matrix-free methods in the Krylov subspace[88,89] such as Newton-GMRES for the solution of such large-scale problems.

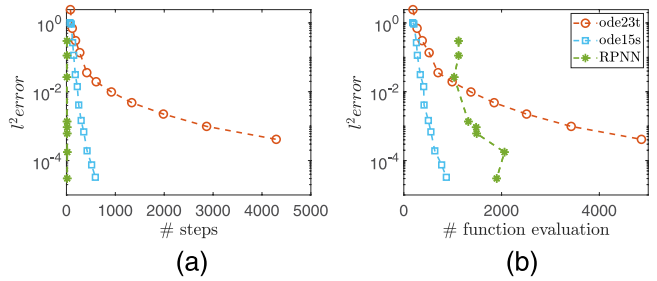


FIG. 17. Allen–Cahn PDE[63] discretized with FD, see Eq. (54). l^2 numerical approximation error vs (a) the number of adaptive steps and (b) the number of function evaluations (rhs of the discretized PDE).

I. Comparison with the DeepXDE library: The Lotka–Volterra ODEs

In this section, we compare the performance of the proposed scheme with a deep learning PINN, as implemented in the DeepXDE library.[40] In particular, we consider a demo of the DeepXDE library for the solution of the Lotka–Volterra ODEs reading,

$$\begin{aligned} \frac{dr}{dt} &= \frac{R}{U}(2Ur - 0.04U^2rp), \\ \frac{dp}{dt} &= \frac{R}{U}(0.02U^2rp - 1.06Up), \\ r(0) &= \frac{100}{U}; \qquad p(0) = \frac{15}{U}, \end{aligned} \tag{55}$$

where the parameters are set as $U = 200, R = 20$, and the solution is sought in the time interval $[0, 1]$. Figure 18 shows the solution profiles obtained with the `ode15s` solver with relative and absolute tolerances set to 1×10^{-14} and 1×10^{-16}, respectively.

For the considered demos, DeepXDE uses 3000 training residual points inside the domain and 3000 points for testing the ODE

residual, which is given by

$$\begin{aligned} \frac{dr}{dt} - \frac{R}{U}(2Ur - 0.04U^2rp) = 0, \\ \frac{dp}{dt} - \frac{R}{U}(0.02U^2rp - 1.06Up) = 0. \end{aligned} \tag{56}$$

The neural network architecture employed in the DeepXDE demo is a deep feed-forward one with six hidden layers with 64 neurons each and hyperbolic tangent as activation function. Furthermore, in order to enforce the prediction to be periodic and thus more accurate, in the demo, the time input is first projected in a seven-dimensional feature layer, which is given by

$$t \to \begin{bmatrix} t & \sin(t) & \sin(2t) & \sin(3t) & \sin(4t) & \sin(5t) & \sin(6t) \end{bmatrix}. \tag{57}$$

Finally, to hard constrain the DeepXDE to satisfy the initial conditions, the 2-dim output $y = (y_1, y_2)$ of the neural network is transformed as

$$\hat{r} = \frac{100}{U} + y_1 \tanh(t), \quad \hat{p} = \frac{15}{U} + y_2 \tanh(t). \tag{58}$$

The default optimization procedure implements the Adam algorithm with a learning rate 0.001 and 50 000 iterations, and then optimization continues with the L-BFGS algorithm in order to achieve a higher accuracy. Please note that the DeepXDE approach is not adaptive and the solution is sought directly in the entire interval (i.e., without any step-size adaptation).

Given the above, from a computational point of view, it is clear that it is much more efficient to proceed in an adaptive-step employing a single-hidden layer RPNN with only 20 neurons (i.e., the only unknown weights are the ones that connect the hidden layer to the output) that can be computed using a Newton-scheme with pseudo-inverse of the Jacobian vs a DeepXDE neural network with $(7 \times 64 + 64^2 \times 5 + 64 \times 2 + 64 \times 6 + 2) = 21\,442$ unknowns (i.e., all the weights and biases need to be learned) that is trained by many iterations of the Adam+L-BFGS algorithms.

In Table I, we compare the performance of the two physics-informed machine learning schemes in terms of mean computational time in seconds and l^2, l^∞, and MAE for (indicatively) the r component with respect to the reference solution computed with `ode15s` setting relative and absolute tolerances to 1×10^{-14} and 1×10^{-16}, respectively. In particular, for the proposed scheme, we have select a range of four relative tolerances, ranging from 1×10^{-03} to 1×10^{-06} for the PIRPNN, and we have varied the number of hidden layer and neurons of the DeepXDE using (a) $3 \times (8)$, (b) 4×16, (c) 5×32, and (d) 6×64 as deep network architectures. The best result, in terms of the numerical approximation accuracy, among the different DeepXDE architectures is taken with the 6×64 structure. This corresponds to a comparable accuracy resulting from the proposed PIRPNN when relatively large tolerances (reltol = 1×10^{-03} or 1×10^{-04}). But in terms of computational times, the propoed PIRPNN can obtain this approximation in just 7.93×10^{-02} (s) vs the 1.73×10^{3} (s) times needed by the DeepXDE, that is, our scheme is around 20 000 times faster for getting a comparable numerical accuracy.

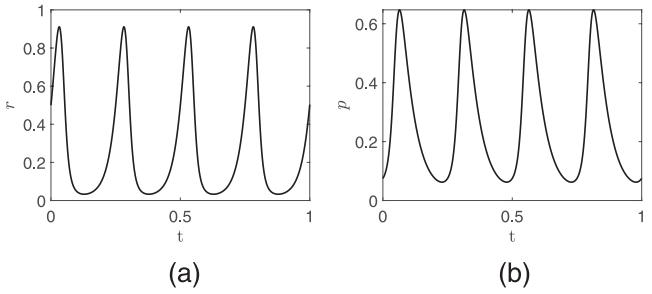


FIG. 18. Lotka–Volterra[40] ODEs in the interval [0, 1], see Eq. (55). (a) r(t), (b) p(t). The numerical reference solution is obtained in the time interval [0 1] using the `ode15s` solver with relative and absolute tolerances set to 1×10^{-14} and 1×10^{-16}, respectively.

V. DISCUSSION

We presented a physics-informed random projection neural network approach for the numerical solution of stiff ODEs and index-1 DAEs. The proposed scheme is a "numerical analysis-assisted" one, in the sense that we have incorporated an adaptive step-size as in the traditional stiff solvers and a continuation method (a concept borrowed from the numerical bifurcation analysis theory) for providing good initial guesses to facilitate the convergence

TABLE I. Lotka–Volterra[40] ODEs in the interval [0, 1], see Eq. (55). Mean computational time in seconds (s) and approximation errors (l^2-norm, l^∞-norm and MAE) for (indicatively) the r component w.r.t. the reference solution computed with `ode15s` with relative and absolute tolerances set to 1×10^{-14} and 1×10^{-16}, respectively. The PIRPNN solutions are computed with relative tolerances ranging from 1×10^{-03} to 1×10^{-06}, and the DeepXDE PINN solutions with 3, 4, 5, 6 hidden layers with 8, 16, 32, 64 neurons, respectively.

		TIME (s)	l^2-error	l^∞-error	MAE
RPNN	tol = 1×10^{-03}	6.75×10^{-02}	2.11×10^{-02}	8.72×10^{-04}	1.22×10^{-04}
	tol = 1×10^{-04}	7.93×10^{-02}	2.33×10^{-03}	9.75×10^{-05}	1.38×10^{-05}
	tol = 1×10^{-05}	1.19×10^{-01}	1.50×10^{-04}	6.27×10^{-06}	8.92×10^{-07}
	tol = 1×10^{-06}	1.24×10^{-01}	1.00×10^{-05}	4.14×10^{-07}	6.29×10^{-08}
DeepXDE	3×8	6.39×10^{2}	2.31×10^{1}	6.50×10^{-01}	1.71×10^{-01}
	4×16	4.04×10^{2}	2.97×10^{00}	1.24×10^{-01}	1.70×10^{-02}
	5×32	1.20×10^{3}	3.95×10^{-01}	1.63×10^{-02}	2.32×10^{-03}
	6×64	1.73×10^{3}	8.03×10^{-03}	2.99×10^{-04}	5.04×10^{-05}

of Newton iterations. Furthermore, in the case of sparse systems, we exploit state-of-the-art numerical analysis methods such as sparse QR decomposition with regularization. The numerical results on eight benchmark problems show that the scheme arises as a promising alternative to well-established ODE solvers for stiff problems and appears to be much more efficient in terms of the computational cost than deep-learning machine learning schemes for the solution of ODEs. Future work will be focused on the further development and application of the scheme for solving large-scale systems of stiff ODEs, DAEs, as well as PDEs. For this task, we aim at integrating ideas from other methods such as DASSL,[90] CSP,[91] and matrix-free methods in the Krylov-subspace[88,89] in order to speed up computations or even recursively recurrent neural networks that yield iterations similar to Newton–Krylov solvers.[92] We also intend to compare its performance on more benchmark ODEs and DAEs against other machine-learning based schemes, including (except from other physics-informed schemes), for example, the recently proposed Runge–Kutta Neural Network integrator for the solution of ODEs.[93]

ACKNOWLEDGMENTS

This work was supported by the Italian program "Fondo Integrativo Speciale per la Ricerca (FISR)"—No. FISR2020IP 02893/B55F20002320001.

AUTHOR DECLARATIONS

Conflict of Interest

The authors have no conflicts to disclose.

Author Contributions

Gianluca Fabiani: Data curation (lead); Formal analysis (equal); Investigation (lead); Methodology (supporting); Software (lead); Validation (lead); Writing – original draft (supporting); Writing – review & editing (equal). **Evangelos Galaris:** Data curation (equal); Formal analysis (supporting); Investigation (supporting); Software (supporting); Validation (supporting). **Lucia Russo:** Conceptualization (equal); Formal analysis (equal); Methodology (supporting); Supervision (equal); Validation (supporting); Writing – original draft (supporting); Writing – review & editing (equal). **Constantinos Siettos:** Conceptualization (lead); Formal analysis (lead); Methodology (lead); Supervision (lead); Writing – original draft (lead); Writing – review & editing (equal).

DATA AVAILABILITY

Data sharing is not applicable to this article as no new data were created or analyzed in this study. Our Matlab software/toolbox for the solution of stiff ODEs and index-1 DAEs, that we call RanDiffNET, is publicly available at https://github.com/GianlucaFabiani/RPNN_for_Stiff_ODEs. Please cite this paper, when you use or modify it for research purposes.

REFERENCES

[1]C. W. Gear, *Numerical Initial Value Problems in Ordinary Differential Equations* (Prentice-Hall, Englewood Cliffs, NJ, 1971), pp. 1–253.
[2]K. E. Brenan, S. L. Campbell, and L. R. Petzold, *Numerical Solution of Initial-Value Problems in Differential-Algebraic Equations* (SIAM, 1995).
[3]C. Gear, "An introduction to numerical methods for odes and daes," in *Real-Time Integration Methods for Mechanical System Simulation* (Springer, 1990), pp. 115–126.
[4]L. F. Shampine and M. W. Reichelt, "The MATLAB ODE suite," SIAM J. Scientific Comput. **18**, 1–22 (1997).
[5]K. Krischer, R. Rico-Martinez, I. G. Kevrekidis, H. Rotermund, G. Ertl, and J. Hudson, "Model identification of a spatiotemporally varying catalytic reaction," Aiche J. **39**, 89–98 (1993).
[6]S. F. Masri, A. G. Chassiakos, and T. K. Caughey, "Identification of nonlinear dynamic systems using neural networks," J. Appl. Mech. **60**, 123–133 (1993).
[7]T. Chen and H. Chen, "Universal approximation to nonlinear operators by neural networks with arbitrary activation functions and its application to dynamical systems," IEEE Trans. Neural Netw. **6**, 911–917 (1995).
[8]A. Taprantzis, C. Siettos, and G. Bafas, "Fuzzy control of a fluidized bed dryer," Drying Technol. **15**, 511–537 (1997).
[9]R. González-García, R. Rico-Martinez, and I. G. Kevrekidis, "Identification of distributed parameter systems: A neural net based approach," Comput. Chem. Eng. **22**, S965–S968 (1998).
[10]C. Siettos, C. Kiranoudis, and G. Bafas, "Advanced control strategies for fluidized bed dryers," Drying Technol. **17**, 2271–2291 (1999).
[11]D. Sagias, E. Sarafis, C. I. Siettos, and G. V. Bafas, "Design of a model identification fuzzy adaptive controller and stability analysis of nonlinear processes," Fuzzy Sets Syst. **121**, 169–179 (2001).
[12]C. I. Siettos, G. V. Bafas, and A. G. Boudouvis, "Truncated chebyshev series approximation of fuzzy systems for control and nonlinear system identification," Fuzzy Sets Syst. **126**, 89–104 (2002).
[13]C. I. Siettos and G. V. Bafas, "Semiglobal stabilization of nonlinear systems using fuzzy control and singular perturbation methods," Fuzzy Sets and Systems **129**, 275–294 (2002).
[14]A. Alexandridis, C. Siettos, H. Sarimveis, A. Boudouvis, and G. Bafas, "Modelling of nonlinear process dynamics using Kohonen's neural networks, fuzzy systems and chebyshev series," Comput. Chem. Eng. **26**, 479–486 (2002).
[15]H. Lee and I. S. Kang, "Neural algorithm for solving differential equations," J. Comput. Phys. **91**, 110–131 (1990).
[16]M. Dissanayake and N. Phan-Thien, "Neural-network-based approximations for solving partial differential equations," Commun. Numer. Meth. Eng. **10**, 195–201 (1994).
[17]A. J. Meade, Jr., and A. A. Fernandez, "The numerical solution of linear ordinary differential equations by feedforward neural networks," Math. Computer Modell. **19**, 1–25 (1994).
[18]R. Gerstberger and P. Rentrop, "Feedforward neural nets as discretization schemes for ODEs and DAEs," J. Computat. Appl. Math. **82**, 117–128 (1997).
[19]I. E. Lagaris, A. Likas, and D. I. Fotiadis, "Artificial neural networks for solving ordinary and partial differential equations," IEEE Trans. Neural Netw. **9**, 987–1000 (1998).
[20]G. E. Karniadakis, I. G. Kevrekidis, L. Lu, P. Perdikaris, S. Wang, and L. Yang, "Physics-informed machine learning," Nat. Rev. Phys. **3**, 422–440 (2021).
[21]J. Bongard and H. Lipson, "Automated reverse engineering of nonlinear dynamical systems," Proc. Natl. Acad. Sci. U.S.A. **104**, 9943–9948 (2007).
[22]M. Raissi, P. Perdikaris, and G. E. Karniadakis, "Machine learning of linear differential equations using gaussian processes," J. Comput. Phys. **348**, 683–693 (2017).
[23]M. Raissi, P. Perdikaris, and G. E. Karniadakis, "Numerical Gaussian processes for time-dependent and nonlinear partial differential equations," SIAM J. Scientific Comput. **40**, A172–A198 (2018).
[24]M. Raissi, P. Perdikaris, and G. E. Karniadakis, "Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations," J. Comput. Phys. **378**, 686–707 (2019).

[25]S. Rudy, A. Alla, S. L. Brunton, and J. N. Kutz, "Data-driven identification of parametric partial differential equations," SIAM J. Appl. Dynam. Syst. **18**, 643–660 (2019).
[26]T. Bertalan, F. Dietrich, I. Mezić, and I. G. Kevrekidis, "On learning hamiltonian systems from data," Chaos **29**, 121107 (2019).
[27]H. Arbabi, J. E. Bunder, G. Samaey, A. J. Roberts, and I. G. Kevrekidis, "Linking machine learning with multiscale numerics: Data-driven discovery of homogenized equations," JOM **72**, 4444–4457 (2020).
[28]S. Lee, M. Kooshkbaghi, K. Spiliotis, C. I. Siettos, and I. G. Kevrekidis, "Coarse-scale pdes from fine-scale observations via machine learning," Chaos **30**, 013141 (2020).
[29]P. R. Vlachas, J. Pathak, B. R. Hunt, T. P. Sapsis, M. Girvan, E. Ott, and P. Koumoutsakos, "Backpropagation algorithms and reservoir computing in recurrent neural networks for the forecasting of complex spatiotemporal dynamics," Neural Netw. **126**, 191–217 (2000).
[30]W. Chen, Q. Wang, J. S. Hesthaven, and C. Zhang, "Physics-informed machine learning for reduced-order modeling of nonlinear problems," J. Comput. Phys. **446**, 110666 (2021).
[31]Y. Chen, B. Hosseini, H. Owhadi, and A. M. Stuart, "Solving and learning nonlinear pdes with gaussian processes," J. Comput. Phys. **447**, 110668 (2021).
[32]L. Lu, P. Jin, G. Pang, Z. Zhang, and G. E. Karniadakis, "Learning nonlinear operators via deeponet based on the universal approximation theorem of operators," Nat. Machine Intelligence **3**, 218–229 (2021).
[33]S. Kim, W. Ji, S. Deng, Y. Ma, and C. Rackauckas, "Stiff neural ordinary differential equations," Chaos **31**, 093122 (2021).
[34]J. Han, A. Jentzen, and E. Weinan, "Solving high-dimensional partial differential equations using deep learning," Proc. Natl. Acad. Sci. U.S.A. **115**, 8505–8510 (2018).
[35]G. Fabiani, F. Calabrò, L. Russo, and C. Siettos, "Numerical solution and bifurcation analysis of nonlinear partial differential equations with extreme learning machines," J. Scientific Comput. **89**, 44 (2021).
[36]F. Calabrò, G. Fabiani, and C. Siettos, "Extreme learning machine collocation for the numerical solution of elliptic PDEs with sharp gradients," Computer Meth. Appl. Mech. Eng. **387**, 114188 (2021).
[37]S. Dong and Z. Li, "Local extreme learning machines and domain decomposition for solving linear and nonlinear partial differential equations," Computer Meth. Appl. Mech. Eng. **387**, 114129 (2021).
[38]S. Dong and Z. Li, "A modified batch intrinsic plasticity method for pre-training the random coefficients of extreme learning machines," J. Comput. Phys. **445**, 110585 (2021).
[39]W. Ji, W. Qiu, Z. Shi, S. Pan, and S. Deng, "Stiff-pinn: Physics-informed neural network for stiff chemical kinetics," J. Phys. Chem. A **125**, 8098–8106 (2021).
[40]L. Lu, X. Meng, Z. Mao, and G. E. Karniadakis, "DeepXDE: A deep learning library for solving differential equations," SIAM Rev. **63**, 208–228 (2021).
[41]E. Schiassi, R. Furfaro, C. Leake, M. De Florio, H. Johnston, and D. Mortari, "Extreme theory of functional connections: A fast physics-informed neural network method for solving ordinary and partial differential equations," Neurocomputing **457**, 334–356 (2021).
[42]M. De Florio, E. Schiassi, and R. Furfaro, "Physics-informed neural networks and functional interpolation for stiff chemical kinetics," Chaos **32**, 063107 (2022).
[43]Y. Lu, R. Maulik, T. Gao, F. Dietrich, I. G. Kevrekidis, and J. Duan, "Learning the temporal evolution of multivariate densities via normalizing flows," Chaos **32**, 033121 (2022).
[44]S. Dong and J. Yang, "On computing the hyperparameter of extreme learning machines: Algorithm and application to computational pdes, and comparison with classical and high-order finite elements," J. Comput. Phys. **463**, 111290 (2022).
[45]X. Meng, Z. Li, D. Zhang, and G. E. Karniadakis, "PPINN: Parareal physics-informed neural network for time-dependent PDEs," Computer Meth. Appl. Mech. Eng. **370**, 113250 (2020).
[46]H. Larochelle, Y. Bengio, J. Louradour, and P. Lamblin, "Exploring strategies for training deep neural networks," J. Machine Learning Res. **10**, 1–40 (2009).
[47]S. Wang, Y. Teng, and P. Perdikaris, "Understanding and mitigating gradient flow pathologies in physics-informed neural networks," SIAM J. Scientific Comput. **43**, A3055–A3081 (2021).
[48]S. Wang, X. Yu, and P. Perdikaris, "When and why pinns fail to train: A neural tangent kernel perspective," J. Comput. Phys. **449**, 110768 (2022).
[49]W. B. Johnson and J. Lindenstrauss, "Extensions of Lipschitz mappings into a Hilbert space," Contemp. Math. **26**, 189–206 (1984).
[50]A. Rahimi and B. Recht, "Weighted sums of random kitchen sinks: Replacing minimization with randomization in learning," in *Advances in Neural Information Processing Systems 21 (NIPS 2008)* (Citeseer, 2008), pp. 1313–1320.
[51]A. N. Gorban, I. Y. Tyukin, D. V. Prokhorov, and K. I. Sofeikov, "Approximation with random bases: Pro et Contra," Inf. Sci. **364**, 129–145 (2016).
[52]Y.-H. Pao and Y. Takefuji, "Functional-link net computing: Theory, system architecture, and functionalities," Computer **25**, 76–79 (1992).
[53]H. Jaeger, "Adaptive nonlinear system identification with echo state networks," Adv. Neural Inform. Process. Syst. **15**, 609–616 (2002).
[54]G.-B. Huang, Q.-Y. Zhu, and C.-K. Siew, "Extreme learning machine: Theory and applications," Neurocomputing **70**, 489–501 (2006).
[55]M. Belkin, D. Hsu, S. Ma, and S. Mandal, "Reconciling modern machine-learning practice and the classical bias–variance trade-off," Proc. Natl. Acad. Sci. U.S.A. **116**, 15849–15854 (2019).
[56]T. A. Davis, *Direct Methods for Sparse Linear Systems* (SIAM, 2006).
[57]H. Robertson, "The solution of a set of reaction rate equations," in *Numerical Analysis: An Introduction* (Academic Press, London, 1966), pp. 178–182.
[58]L. F. Shampine, M. W. Reichelt, and J. A. Kierzenka, "Solving index-1 daes in matlab and simulink," SIAM Rev. **41**, 538–552 (1999).
[59]J. L. Hindmarsh and R. Rose, "A model of neuronal bursting using three coupled first order differential equations," Proc. R. Soc. London Ser. B **221**, 87–102 (1984).
[60]B. P. Belousov, "A periodic reaction and its mechanism," in *Oscillations and Traveling Waves in Chemical Systems*, edited by R. J. Field and M. Burger (Wiley, New York, 1985).
[61]A. M. Zhabotinsky, "Periodic course of the oxidation of malonic acid in a solution (Studies on the kinetics of Beolusov's reaction)," Biofizika **9**, 306–311 (1964).
[62]S. M. Allen and J. W. Cahn, "A microscopic theory for antiphase boundary motion and its application to antiphase domain coarsening," Acta Metall. **27**, 1085–1095 (1979).
[63]L. N. Trefethen, *Spectral Methods in MATLAB* (SIAM, 2000).
[64]G.-B. Huang, L. Chen, and C. K. Siew, "Universal approximation using incremental constructive feedforward networks with random hidden nodes," IEEE Trans. Neural Netw. **17**, 879–892 (2006).
[65]G. Söderlind, "Automatic control and adaptive time-stepping," Numer. Algorithms **31**, 281–310 (2002).
[66]L. F. Shampine and C. W. Gear, "A user's view of solving stiff ordinary differential equations," SIAM Rev. **21**, 1–17 (1979).
[67]A. G. Baydin, B. A. Pearlmutter, A. A. Radul, and J. M. Siskind, "Automatic differentiation in machine learning: A survey," J. Machine Learn. Res. **18**, 1–43 (2018).
[68]B. Igelnik and Y.-H. Pao, "Stochastic choice of basis functions in adaptive function approximation and the functional-link net," IEEE Trans. Neural Netw. **6**, 1320–1329 (1995).
[69]D. Husmeier, "Random vector functional link (RVFL) networks," in *Neural Networks for Conditional Probability Estimation* (Springer, 1999), pp. 87–97.
[70]D. J. Gauthier, E. Bollt, A. Griffith, and W. A. Barbosa, "Next generation reservoir computing," Nat. Commun. **12**, 1–8 (2021).
[71]L. A. Thiede and U. Parlitz, "Gradient based hyperparameter optimization in echo state networks," Neural Netw. **115**, 23–29 (2019).
[72]G.-B. Huang, "An insight into extreme learning machines: Random neurons, random features and kernels," Cognit. Computat. **6**, 376–390 (2014).
[73]B. Ghojogh, A. Ghodsi, F. Karray, and M. Crowley, "Johnson-lindenstrauss lemma, linear and nonlinear random projections, random fourier features, and random kitchen sinks: Tutorial and survey," arXiv:2108.04172 (2021).
[74]R. Giryes, G. Sapiro, and A. M. Bronstein, "Deep neural networks with random Gaussian weights: A universal classification strategy?," IEEE Trans. Signal Process. **64**, 3444–3457 (2016).
[75]F. Rosenblatt, *Perceptions and the Theory of Brain Mechanisms* (Spartan Books, 1962).
[76]A. R. Barron, "Universal approximation bounds for superpositions of a sigmoidal function," IEEE Trans. Inf. Theory **39**, 930–945 (1993).

[77]A. Rahimi and B. Recht, "Uniform approximation of functions with random bases," in *2008 46th Annual Allerton Conference on Communication, Control, and Computing* (IEEE, 2008), pp. 555–561.
[78]P. Collins, and O. U. K. M. Inst., *Differential and Integral Equations: Part II* (University of Oxford Mathematical Institute, 1988).
[79]N. De Villiers and D. Glasser, "A continuation method for nonlinear regression," SIAM J. Numer. Anal. **18**, 1139–1154 (1981).
[80]T. A. Davis, "Algorithm 915, suitesparseqr: Multifrontal multithreaded rank-revealing sparse QR factorization," ACM Trans. Math. Software (TOMS) **38**, 1–22 (2011).
[81]I. Gladwell, L. Shampine, and R. Brankin, "Automatic selection of the initial step size for an ode solver," J. Computat. Appl. Math. **18**, 175–192 (1987).
[82]E. Hairer, S. P. Norsett, and G. Wanner, *Solving Ordinary, Differential Equations I, Nonstiff Problems, with 135 Figures*, 2nd ed. (Springer-Verlag, 2000), Vol. 1.
[83]W. H. Enright, K. R. Jackson, S. P. Nørsett, and P. G. Thomsen, "Interpolants for Runge-Kutta formulas," ACM Trans. Math. Software (TOMS) **12**, 193–218 (1986).
[84]A. Prothero and A. Robinson, "On the stability and accuracy of one-step methods for solving stiff systems of ordinary differential equations," Math. Comput. **28**, 145–162 (1974).
[85]L. Métivier and P. Montarnal, "Strategies for solving index one dae with non-negative constraints: Application to liquid–liquid extraction," J. Comput. Phys. **231**, 2945–2962 (2012).
[86]R. Belusov, "Periodicheski deistvuyushchaya reaktsia i ee mekhanism {Periodically acting reaction and its mechanism}," in *Sbornik referatov po radiotsionnoi meditsine (Collection of Abstracts on Radiation Medicine)* (Medgiz, Moscow, 1958), pp. 145–147.
[87]V. Shulyk, O. Klymenko, and I. Svir, "Numerical solution of stiff odes describing complex homogeneous chemical processes," J. Math. Chem. **43**, 252–264 (2008).
[88]P. N. Brown, A. C. Hindmarsh, and L. R. Petzold, "Using Krylov methods in the solution of large-scale differential-algebraic systems," SIAM J. Scientific Comput. **15**, 1467–1488 (1994).
[89]C. T. Kelley, *Iterative Methods for Optimization* (SIAM, 1999).
[90]L. R. Petzold, "Description of dassl: A differential/algebraic system solver," Tech. Rep. (Sandia National Labs., Livermore, CA (USA), 1982).
[91]M. Hadjinicolaou and D. A. Goussis, "Asymptotic solution of stiff pdes with the CSP method: The reaction diffusion equation," SIAM J. Scientific Comput. **20**, 781–810 (1998).
[92]D. T. Doncevic, A. Mitsos, Y. Guo, Q. Li, F. Dietrich, M. Dahmen, and I. G. Kevrekidis, "A recursively recurrent neural network (R2N2) architecture for learning iterative algorithms," arXiv:2211.12386 (2022).
[93]Y. Guo, F. Dietrich, T. Bertalan, D. T. Doncevic, M. Dahmen, I. G. Kevrekidis, and Q. Li, "Personalized algorithm generation: A case study in learning ode integrators," SIAM J. Scientific Comput. **44**, A1911–A1933 (2022).
[94]W. F. Schmidt, M. A. Kraaijveld, and R. P. W. Duin, "Feedforward neural networks with random weights," *Proceedings, 11th IAPR International Conference on Pattern Recognition. Vol.II. Conference B: Pattern Recognition Methodology and Systems*, The Hague, Netherlands, 30 Aug.–3 Sep. 1992 (IEEE, 1992).
[95]D. S. Broomhead and D. Lowe "Multivariable functional interpolation and adaptive networks," Complex Systems **2**, 321–355 (1988).