

Time discretization in the solution of parabolic PDEs with ANNs



Francesco Calabrò*, Salvatore Cuomo, Daniela di Serafino, Giuseppe Izzo, Eleonora Messina

Dipartimento di Matematica e Applicazioni "Renato Caccioppoli", Università degli Studi di Napoli Federico II, Member of the INdAM-GNCS Research Group, Italy

ARTICLE INFO

Keywords:

Numerical methods for parabolic PDEs
Scientific machine learning
Extreme learning machine
Physics-informed methods

ABSTRACT

We investigate the resolution of parabolic PDEs via Extreme Learning Machine (ELMs) Neural Networks. An Artificial Neural Network (ANN) is an interconnected group of nodes - where activation functions act- organized in numbered layers. From a mathematical point of view, an ANN represents a combination of activation functions that is linear if only one hidden layer is admitted. In the present paper, the ELMs setting is considered, and this gives that a single hidden layer is admitted and that the ANN can be trained at a modest computational cost as compared to Deep Learning Neural Networks. Our approach addresses the time evolution by applying classical ODEs techniques and uses ELM-based collocation for solving the resulting stationary elliptic problems. In this framework, the θ -method and Backward Differentiation Formulae (BDF) techniques are investigated on some linear parabolic PDEs that are challenging problems for the stability and accuracy properties of the methods. The results of numerical experiments confirm that ELM-based solution techniques combined with BDF methods can provide high-accuracy solutions of parabolic PDEs.

© 2023 The Author(s). Published by Elsevier Inc.
This is an open access article under the CC BY license
(<http://creativecommons.org/licenses/by/4.0/>)

1. Introduction

Mesh-based schemes are widely discussed numerical approaches for solving Partial Differential Equations (PDEs). Finite Difference Methods (FDMs), Finite Element Methods (FEMs), and Finite Volume Methods (FVMs) all belong to this class of numerical methods [33]. They require the generation of grid points or elements, usually the discretization of differential/integral formulas, and the solution of discrete equations, often with iterative algorithms. Mesh-based approaches suffer from several problems: firstly, the shape complexity of the computational domain where the grid generation itself could become very difficult or even infeasible; moreover, the problem discretization could create a bias between the mathematical nature of the PDE and its approximating model.

In recent years, starting from [30], Artificial Neural Networks (ANNs) have been considered an interesting alternative methodology to overcome the drawbacks of mesh-based numerical schemes. ANNs are taken as trial space for the approximation of the solution of a PDE, i.e. the approximate problem solutions are determined by a learning approach that consists

* Corresponding author.

E-mail addresses: calabro@unina.it (F. Calabrò), salvatore.cuomo@unina.it (S. Cuomo), daniela.diserafino@unina.it (D. di Serafino), giuseppe.izzo@unina.it (G. Izzo), eleonora.messina@unina.it (E. Messina).

in defining an optimization algorithm in which losses due to ANN approximations of PDEs and boundary conditions (BCs) are minimized. This approach requires sampling points inside the domain and on the boundary, which can be randomly selected. When the numerical approximation of the unknown solution is guided by the resolution of the underlying PDE, the methods are referred to as physics informed [8,28].

Recently, it has been demonstrated that PDEs can be solved by considering a specific ANN called Extreme Learning Machine (ELM). An ELM is a feed-forward neural network with a single hidden layer that randomly assigns the input layer weights and analytically determines the output weights, see [24]. Thanks to this architecture, the weights of the hidden layer need not be learned. This makes ELMs faster than typical deep neural networks, where optimization methods may lead to prohibitively slow learning speeds. We point out that ELMs are variants of the random projection networks originally proposed in [35], and a review of the relation between this and other theories regarding random networks can be found in [6]. Overall, randomized Neural Networks boost the learning task with benefits on the numerical scheme in terms of efficiency, while maintaining high accuracy. The first use of ELMs for differential problems can be found in [13], where the authors implemented a model called Physics Informed Extreme Learning Machine (PIELM), and then ELMs have been successfully applied to solve Ordinary Differential Equations (ODEs) and stationary PDEs [5,10,12,13,15]. ELMs are mesh-free methods and thus they can easily address complex geometries of PDE domains [13]. Moreover, ELMs are universal approximators [25,26] and hence can potentially represent any continuous function.

In this paper, we consider ELMs for the numerical resolution of a linear parabolic equation. Following what is discussed in [5], we use collocation for the resolution of stationary elliptic problems but address separately the time marching. The resolution of the elliptic problem has provided a very good accuracy with a modest computational cost in situations where classical methods fail to give good results. Moreover, it has very nice properties of generalization and has been applied to nonlinear problems in [15]. As is usually done in time-dependent problems, we aim to separate the time dependence by representing the unknown solution as a combination of stationary functions where the coefficients of the combination change in time.

Now we show how collocation can be applied to our differential problem. Let $u(t, \mathbf{x})$ be the solution to the following:

$$\frac{\partial u}{\partial t} = \mathcal{L}(u) + f(t, \mathbf{x}), \quad \mathcal{B}(u) = g(t, \mathbf{x}), \tag{1}$$

where \mathcal{L} is intended as a linear elliptic operator acting only on the spatial variables, \mathcal{B} is a boundary operator and f, g are given source and boundary data, and in the boundary operator we include also the initial condition. We assume that the data f, g are sufficiently smooth and that the operators \mathcal{L}, \mathcal{B} pose an unique solution, for details see [38] and references therein. We are interested in the numerical resolution of Eq. (1) constructed by splitting the time and the space dependencies. In this paper we develop the approximate solution such that the component in space is an ANN of ELM type, as we present in the next section. Thus, the solution has the following expression:

$$u(t, \mathbf{x}) \approx u^h(t, \mathbf{x}) = \sum_i w_i(t) \sigma_i(\mathbf{x}). \tag{2}$$

The main assumption of our collocation method is that we are able to compute the action of the operators

$$\mathcal{L}(\sigma_i(\mathbf{x})), \quad \mathcal{B}(\sigma_i(\mathbf{x})). \tag{3}$$

From now on we denote by $\{\mathbf{x}_j\}$ the set of points where the previous quantities are evaluated. In the rest of the paper the boundary operator will be of Dirichlet type, so that the satisfaction the boundary conditions both for the stationary problem and the satisfaction of the initial condition for the time problem are of the same type. Then, two strategies are possible for the time marching, and these lead to different classes of methods, as specified below.

1. In the first class we focus on the unknown functions $w_i(t)$ after applying the operator \mathcal{L} (or some approximation to it) to the functions σ_i . In this case we make a semi-discretization in space and then solve a system of ODEs, which is linear if the PDE is linear. In order to describe this approach, we apply the operators $\frac{\partial}{\partial t}$ and \mathcal{L} to the solution in (2):

$$\frac{\partial}{\partial t} u^h(t, \mathbf{x}) = \frac{\partial}{\partial t} \left(\sum_i w_i(t) \sigma_i(\mathbf{x}) \right) = \sum_i \sigma_i(\mathbf{x}) \frac{d}{dt} w_i(t), \tag{4}$$

$$\mathcal{L}(u^h(t, \mathbf{x})) = \mathcal{L} \left(\sum_i w_i(t) \sigma_i(\mathbf{x}) \right) = \sum_i w_i(t) \mathcal{L}(\sigma_i(\mathbf{x})). \tag{5}$$

Now we impose that (4) and (5) solve exactly problem (1) at the given points \mathbf{x}_j . Let $G = (g_{ji})$ be matrix where $g_{ji} = \sigma_i(\mathbf{x}_j)$, and let $L = (l_{ji})$ be the matrix where $l_{ji} = \mathcal{L}(\sigma_i(\mathbf{x}_j))$. With these positions, the final resolution corresponds to solving

$$G \frac{d}{dt} \mathbf{w}(t) = L \mathbf{w}(t) + \mathbf{f}(t) \tag{6}$$

where $\mathbf{w}(t) = (w_i(t))_i$ is the vector of the unknowns and $\mathbf{f}(t) = (f(t, \mathbf{x}_j))_j$ is the vector of the source terms. We refer to [37] for details.

2. In the second class of methods we define

$$\tilde{u}^{[n]}(\mathbf{x}) = \sum_i w_i^{[n]} \sigma_i(\mathbf{x}), \tag{7}$$

where $w_i^{[n]} = w_i(t_n)$ and $\{t_n\}$ is a discretization of the time interval with fixed time spacing Δt . Then the unknowns are the coefficients $w_i^{[n]}$, which are computed by solving a boundary problem. In this case, we first perform a semi-discretization in time and then solve an elliptic stationary problem, which is linear if the PDE is linear. As an example, let us consider the well-known θ -method, which includes the explicit Euler method for $\theta = 0$, the backward (implicit) Euler method for $\theta = 1$ and the trapezoidal (Crank-Nicolson) method for $\theta = 1/2$. At each time step, we look for the function $\tilde{u}^{[n]}(\mathbf{x})$ in the hypothesis that $\tilde{u}^{[n-1]}(\mathbf{x})$ is known, i.e. given or computed. Then, the method solves the following:

$$\frac{\tilde{u}^{[n]}(\mathbf{x}) - \tilde{u}^{[n-1]}(\mathbf{x})}{\Delta t} = \theta [\mathcal{L}(\tilde{u}^{[n]}(\mathbf{x})) + f(t_n, \mathbf{x})] + (1 - \theta) [\mathcal{L}(\tilde{u}^{[n-1]}(\mathbf{x})) + f(t_{n-1}, \mathbf{x})], \tag{8}$$

that is

$$-\theta \Delta t \mathcal{L}(\tilde{u}^{[n]}(\mathbf{x})) + \tilde{u}^{[n]}(\mathbf{x}) = \tilde{u}^{[n-1]}(\mathbf{x}) + \theta \Delta t f(t_n, \mathbf{x}) + (1 - \theta) \Delta t [\mathcal{L}(\tilde{u}^{[n-1]}(\mathbf{x})) + f(t_{n-1}, \mathbf{x})].$$

The last equation can be regarded as an elliptic equation.¹ The right-hand side consists of given functions and quantities that can be explicitly evaluated.

The first class is more natural for problems where the diffusion is dominated by the transport, leading to hyperbolic-type behaviors, or where stiffness arises in time. In these cases, one can easily adopt different time discretizations. Moreover, the first class of methods can be profitably applied when the matrix involved in the problem has a structure that can be exploited. The introduction and study of different time-marching procedures have been proposed also for the parabolic case, for both traditional methods [16,17] and newer ones, such as Isogeometric Analysis [1,2,14] and Physics-Informed Neural Networks [8].

On the other hand, the differential problem that has to be solved in the case of ELMs involves a usually dense (and not a-priori banded) matrix and the properties of this matrix are difficult to establish. In the authors' opinion, the ELM collocation can be more successfully applied in the second class of time-marching methods, as we discuss in this paper. A main advantage is that one can apply different strategies for the resolution of the elliptic problem at different time steps, e.g. by changing the collocation points.

The aim of this work is to explore how the time approximation affects the resolution of time-dependent parabolic PDEs when the time marching is made according to the second class of methods described before. This is done here for the first time for parabolic problems where the resolution at fixed times is done by collocation with ANNs trained by ELM. The use of implicit methods for advancing in time well combines with the proposed collocation solution, as a matter of fact the resolution at fixed time can take information at previous time step in general locations. Indeed, each computed solution at fixed time step is a linear combination of functions, so that this can be evaluated in arbitrary grid points when this solution becomes part of a forcing term for the new resolution. This can be used to adapt the location of the collocation sites if some properties of the problem in hand are known. Other collocation methods, such as the finite difference methods and the time marching procedure in the first class described before cannot in principle change the location of the grid points. For the location of space grid points, we have performed the numerical tests with equispaced, Chebychev, and randomly perturbed equispaced points. In particular, in the last case, the perturbation changes at each time step, so to test the ability of the method to deal with points that vary in time.

For the time marching, we start from the above-mentioned θ -method and then apply Backward Differentiation Formula (BDF) methods to overcome some difficulties, see [7,22]. BDF methods show good accuracy and nice convergence properties while keeping the computational cost unchanged with respect to the implicit θ -method. Moreover, when the time step is chosen to be constant, they are easy to implement.

The numerical tests fully confirm the reported results concerning stability and order of convergence of the proposed time-discretization schemes also when combined with the ELM-based collocation method. In the case of the parabolic problems presented here, this reveals that BDF methods are to be preferred to the well-established Backward Euler or Trapezoidal rules because the latter are unable to deal with accuracy requirements, as they are slowly convergent and can suffer from order reduction or numerical instabilities. Finally, we notice that with the proposed procedure the challenging task of the transition between the initial condition and the rest of the evolution is successfully addressed. In fact, we consider initial conditions that do not satisfy the other boundary conditions and obtain good solutions.

The rest of this paper is organized as follows. In Section 2 we briefly discuss the use of Single hidden-Layer Feedforward Networks (SLFNs) as function approximators, focusing on the selection of the Activation Functions (AFs) and on the

¹ Notice that it reveals to an explicit method – the forward Euler method – in the case $\theta = 0$.

application of ELMs as collocation methods for the solution of PDEs. In Section 3 we present BDF solvers, which are used as time-marching methods in the ELM-based resolution. The results of numerical experiments, reported in Section 4, show that our approach is able to provide accurate solutions to linear, but challenging, parabolic PDE problems, according with the order of the time-marching procedure and the theoretical results on ELMs. Some conclusions are given in Section 5.

In the following, vectors are written in boldface, i.e. \mathbf{v} , while scalars are written in lightface, i.e. v . The i -th entry of a vector \mathbf{v} is denoted v_i and the scalar product of \mathbf{v} and \mathbf{w} is denoted $\mathbf{v} \cdot \mathbf{w}$. Finally, $\|\cdot\|$ indicates either the Euclidean norm of a vector or the L^2 -norm of a function.

2. Use of ANNs as approximators

Today many researchers agree that ANNs learn to approximate functions. More in detail, ANNs are techniques for estimating an unknown function using available observations or collocation points from the function domain [20,32]. The function to be estimated, which transforms inputs into outputs, is often referred to as the target function.

Here we discuss the application of SLFNs with random hidden nodes to a differential problem. A SLFN with N hidden nodes and AF $\phi : \mathbb{R} \rightarrow \mathbb{R}^m$ is a function $\mathcal{F} : \mathbb{R}^d \rightarrow \mathbb{R}^m$ defined as follows:

$$\mathcal{F}(\mathbf{x}) = \sum_{i=1}^N \mathbf{w}_i \phi_i(\mathbf{x}) = \sum_{i=1}^N \mathbf{w}_i \phi(\boldsymbol{\alpha}_i \cdot \mathbf{x} + \beta_i), \tag{9}$$

where $\boldsymbol{\alpha}_i = (\alpha_{i1}, \alpha_{i2}, \dots, \alpha_{id})$ is the weight vector linking the input nodes to the i -th hidden node, $\mathbf{w}_i = (w_{i1}, w_{i2}, \dots, w_{im})$ is the weight vector linking the i -th hidden node to the output nodes, and β_i is a bias of the i -th hidden node. Notice that the functions $\phi_i : \mathbb{R}^d \rightarrow \mathbb{R}^m$ acting are translated and shrunk copies of the reference AF ϕ . Our aim is to study whether the SLFN $\mathcal{F}(\mathbf{x})$ fits the data. More formally, given M arbitrary couples $(\mathbf{x}_j, \boldsymbol{\tau}_j)$, where

$$\mathbf{x}_j = (x_{j1}, x_{j2}, \dots, x_{jd}) \in \mathbb{R}^d, \quad \boldsymbol{\tau}_j = (\tau_{j1}, \tau_{j2}, \dots, \tau_{jm}) \in \mathbb{R}^m,$$

and $\mathbf{x}_j \neq \mathbf{x}_i$ for $j \neq i$, these are M samples where the SLFN passes with zero error:

$$\mathcal{F}(\mathbf{x}_j) = \boldsymbol{\tau}_j, \quad j = 1, 2, \dots, M.$$

SLFNs, as a special case of ANNs, are universal approximators in the sense that a feedforward network with a linear output layer and at least one hidden layer with any nonlinear activation function can approximate with good accuracy a given function from a finite-dimensional space to another. Such accuracy depends on the number of nodes employed in the hidden layer and many research addresses such issues, see, e.g., [9,21] and the recent review [29]. Here our main interest is on scalar functions, corresponding to $m = 1$. In this case, the vectors $\boldsymbol{\tau}_i$ become the scalars τ_i and function (9) reads

$$\mathcal{F}(\mathbf{x}) = \sum_{i=1}^N w_i \phi_i(\mathbf{x}) = \sum_{i=1}^N w_i \phi(\boldsymbol{\alpha}_i \cdot \mathbf{x} + \beta_i), \tag{10}$$

where the weights w_i are scalar too.

For the sake of completeness, we report next a version of the universal approximation theorem for the SLFN (10), see [32, Theorem 3.1].

Theorem 1. *Let \mathcal{F} be a SLFN function as in (10), where ϕ is not a polynomial. For any continuous function $f : \mathbb{R}^d \rightarrow \mathbb{R}$, any compact subset $K \subset \mathbb{R}^d$ and any $\varepsilon > 0$, there exist weights and biases $\mathbf{w}_i, \boldsymbol{\alpha}_i, \beta_i$ such that*

$$\max_{\mathbf{x} \in K} |\mathcal{F}(\mathbf{x}) - f(\mathbf{x})| < \varepsilon. \tag{11}$$

This result states that a single-layer network is enough to have universal approximation. In order to get a good approximation, we attempt to minimize the left-hand side of (11) or, more generally, a measure of the distance between \mathcal{F} and f , usually referred to as *loss function*.

Additional hidden layers can help model intricate nonlinear dynamics [31,36]. In deep networks with many layers, an important role is played by Physics Informed Neural Networks (PINNs), often adopted for solving real problems. However, deep learning approaches are characterized by high training costs and by efficiency issues, while we use shallow networks with random projection neurons, leading to ELM networks.

2.1. Selection of the activation functions

A crucial aspect of the learning approach is the selection of AFs, since they can significantly affect the accuracy and efficiency of an ANN. Commonly used AFs are ReLU, leaky ReLU, Sigmoid, Tanh (see, e.g., [18,20]). However, for all non-polynomial AFs an interpolation result can be stated [32, Theorem 5.1]. We report it next.

Theorem 2. Let \mathcal{F} be a SLFN function as in (10), where ϕ is not a polynomial. For any M distinct points \mathbf{x}_j and associated data τ_j , there exists a choice of M coefficients $\alpha_i = (\alpha_{i1}, \dots, \alpha_{id}) \in \mathbb{R}^d$, $w_i \in \mathbb{R}$, $\beta_i \in \mathbb{R}$ such that

$$\sum_{i=1}^M w_i \phi(\alpha_i \cdot \mathbf{x}_j + \beta_i) = \tau_j, \quad j = 1, \dots, M.$$

In other words, for an SLFN architecture with $N = M$ hidden nodes, it is possible to approximate M samples with zero mean error.

In this work we choose the AFs in the class of sigmoid functions that verify the hypotheses of Theorems 1–2. A sigmoid usually takes values between 0 and 1 and it is widely used for models where a probability has to be predicted as an output. Specifically, our choice for the AF is the logistic sigmoid function:

$$\sigma_i(\mathbf{x}) = \sigma(\alpha_i \cdot \mathbf{x} + \beta_i) = \frac{1}{1 + \exp(-\alpha_i \cdot \mathbf{x} - \beta_i)}. \tag{12}$$

The derivatives of (12) with respect to the independent variable \mathbf{x} can be easily computed. Note that if one takes two functions σ_i and σ_j , where at least one of the parameters is different, then these functions are linearly independent, see [27]. Moreover, each σ_i is a planar-wave Ridge function [32], so that the behavior of the function can be derived easily as an extension of the univariate case $d = 1$. With this simplification, we can state the following:

- σ_i has an inflection point at $x = -\frac{\beta_i}{\alpha_i}$, which we call the center C_i of the sigmoid function;
- σ_i is monotone, $\lim_{x \rightarrow -\infty} \sigma_i = 0$ and $\lim_{x \rightarrow +\infty} \sigma_i = 1$ if α_i is positive, the other way if α_i is negative. Moreover, the range where the values are between 0.05 and 0.95 is $\left[C_i - \frac{2.945}{\alpha_i}, C_i + \frac{2.945}{\alpha_i} \right]$.

Notice that one obtains Heaviside-like functions if the internal weights α_i are large, or almost-linear functions if the α_i 's are small. In our case, the use of both kind of functions can help approximate steep gradients and global behaviors. These functions are an example of AFs in the class of those that verify the hypotheses of Theorems 1 and 2.

2.2. Shallow networks and ELMs

The aim of the overall network is to have nice properties of reproduction while maintaining small the number of unknowns.

One possible strategy is to use shallow networks, e.g. single hidden-layer networks with an increasing number of neurons in the hidden layer. In the PINNs context, shallow ANNs have been considered a good choice with respect to deep learning methodologies. Among these shallow networks, ELMs are the ones where internal parameters are fixed randomly and only the external weights w_i are trainable parameters. In [26, Theorem 2.1] an approximation result needed in our setting is given. We report it next.

Theorem 3. Let (\mathbf{x}_j, τ_j) , $j = 1, \dots, M$, be a set of points such that $\mathbf{x}_{j_1} \neq \mathbf{x}_{j_2}$ if $j_1 \neq j_2$, and let $u_N(\mathbf{x}) = \sum_{i=1}^N w_i \sigma_i(\mathbf{x}) = \sum_{i=1}^N w_i \sigma(\alpha_i \cdot \mathbf{x} + \beta_i)$ be an ELM network with $N < M$ neurons such that the internal weights α_i and the biases β_i are randomly generated independently from the data, according to any continuous probability distribution. Then, for all $\varepsilon > 0$ there exists a choice of the weights w_i such that

$$\| (u_N(\mathbf{x}_j) - \tau_j)_j \| < \varepsilon \quad \text{with probability 1,}$$

where $(u_N(\mathbf{x}_j) - \tau_j)_j$ denotes the vector with components $u_N(\mathbf{x}_j) - \tau_j$. Moreover, if $N = M$ then w_i , $i = 1, \dots, N$, can be found such that

$$\| (u_N(\mathbf{x}_j) - \tau_j)_j \| = 0 \quad \text{with probability 1.}$$

In particular, the above theorem states that if the number of hidden neurons is equal to the number of data points, then the interpolation error is zero with probability 1. Unlucky cases, i.e. with probability 0, are related to the unsolvence of the points. In the case $d = 1$ the unsolvence hypothesis is included in the request that the points are distinct.

The interpolation property of Theorem 2 can be extended to a convergence result, as proved in [25] (see also [24, Theorem 2]).

Theorem 4. Let $\phi : \mathbb{R}^d \rightarrow \mathbb{R}$ be a continuous function. Then there exists a sequence of ELM network functions $u_N(\mathbf{x}) = \sum_{i=1}^N w_i \sigma(\alpha_i \cdot \mathbf{x} + \beta_i)$ such that:

$$\lim_{N \rightarrow \infty} \|\phi - u_N\| = 0.$$

A fundamental challenge in all the works about neural networks is to find an optimal choice of parameters that satisfies a desired tolerance. As commented before, with ELMs it is possible to focus on the optimization of the external weights

w_i only, reducing the computational cost and the training time. Thus, the weights w_i can be seen as the coefficients of the linear combination defining $u_N(\mathbf{x})$, while α_i and β_i are internal weights and biases that yield a variation of the sigmoid function.

Firstly, we present the way in which the parameters α_i and β_i are taken, and then we discuss the computation of the weights w_i . The internal weights α_i are chosen randomly and uniformly in a range that depends on the number of neurons N . By fixing the domain of the differential problem to have unitary length and following the analysis in [5], we choose

$$\alpha_i = \mathbf{rand} \left(\left[-\frac{N-10}{10} - 4, \frac{N-10}{10} + 4 \right] \right), \tag{13}$$

where $\mathbf{rand}([a, b])$ denotes for each i a vector with components sampled from a uniform distribution in the interval $[a, b]$. The biases β_i are set so that the functions σ_i are “non-flat” in the considered domain, as shown at the end of Section 2.1.

The trainable parameters of the network are the weights w_i . In general ANNs, the weights are computed by minimizing the loss function, e.g. by applying stochastic gradient-based approaches that back-propagate the error and adjust the weights through specific directions [4]. More recently, second-order stochastic optimization methods have been widely investigated to get better performances than first-order methods, especially when ill-conditioned problems must be solved, see, e.g., [11] and the references therein. Nevertheless, there are still difficulties in using these approaches, such as the setting of the so-called hyperparameters and the significant increase in computing time when the number of nodes in the hidden layer grows.

In our case, by collocating the linear problem (1) we obtain a linear dependence on the unknown weights w_i and we refer to such discrete problem via a collocation matrix C . Following Theorems 3 and 4, we can choose a number of equations smaller than the number of unknowns, obtaining an underdetermined linear system that can be solved as a least squares problem, which plays the role of a loss function. By computing the minimum-norm least squares solution [3] we not only obtain existence and uniqueness of the weights but also ensure minimum training error and the smallest norm of the weight vector, which are important properties for an ELM [26]. This solution can be obtained by using a Complete Orthogonal Decomposition (COD) [3] of the collocation matrix C :

$$CP = QRZ^T,$$

where Q and Z are orthogonal matrices, R is upper trapezoidal matrix and P is a permutation matrix. We note that COD has good stability properties even in the case C has (numerical) rank $r < N$ [23]. On the other hand, in general, COD does not tend to zero the entries of the solution, thus producing functions u_N where most of the coefficients w_i are likely to be nonzero.

Alternative solutions to the undetermined linear system can be computed by using, e.g., the QR factorization with column pivoting [3]:

$$CP = QR,$$

where Q is an orthogonal matrix, R is an upper triangular matrix and P is a permutation matrix. In this case, at least $M - N$ coefficients w_i are set equal to zero, generally yielding a sparser solution than in the previous case.

Preliminary numerical experiments have shown that the difference between the PDE solutions obtained with the two approaches are practically negligible. Therefore, in our tests we compute the minimum-norm least squares solution to the underdetermined linear system.

3. Time-marching scheme

In the literature on numerical methods for diffusion equations much attention has been paid to the construction and analysis of stable and accurate approximation schemes [7,16,34]. In this work we compare the trapezoidal (8) and BDF methods according to the second approach described in the introduction, thus we follow the notation introduced in Eq. (7). Given

$$\tilde{u}^{[0]}(x), \tilde{u}^{[1]}(x), \dots, \tilde{u}^{[k-1]}(x), \tag{14}$$

a k -step BDF applied to problem (1) can be written as

$$\Delta t b_k \mathcal{L}(\tilde{u}^{[n+k]}(x)) + a_k \tilde{u}^{[n+k]}(x) = - \sum_{j=0}^{k-1} a_j \tilde{u}^{[n+j]}(x) + \Delta t b_k f(t_{n+k}, x), \tag{15}$$

where $n = 0, 1, \dots$ and the coefficient a_j , $j = 0, \dots, k$, and b_k are listed in Table 1 for $k \leq 6$. It is worth noting that the 1-step BDF is the Backward Euler method.

BDF methods are L -stable for $k = 1, 2$ and $L(\alpha)$ -stable for $k = 3, \dots, 6$, with stability angle α reported in the last column of Table 1. It is also well known that they are not zero-stable for $k > 6$. Definitions and further details can be found in [19]. The L -stability and $L(\alpha)$ -stability make the methods suitable for ODEs whose solutions present high-frequency components. Furthermore, since BDF methods involve the right-hand side evaluation only at the right end of the current step, they allow the preservation of the elliptic structure in the time-discretized operator. In particular, the elliptic operator on the left-hand

Table 1
Coefficients and $L(\alpha)$ -stability angles for the BDF methods.

k	a_6	a_5	a_4	a_3	a_2	a_1	a_0	b_k	α
1						1	-1	1	90°
2					1	$-\frac{4}{3}$	$\frac{1}{3}$	$\frac{2}{3}$	90°
3				1	$-\frac{18}{11}$	$\frac{9}{11}$	$-\frac{2}{11}$	$\frac{6}{11}$	86.03°
4			1	$-\frac{48}{25}$	$\frac{36}{25}$	$-\frac{16}{25}$	$\frac{3}{25}$	$\frac{12}{25}$	73.35°
5		1	$-\frac{300}{137}$	$\frac{300}{137}$	$-\frac{200}{137}$	$\frac{75}{137}$	$-\frac{12}{137}$	$\frac{60}{137}$	51.84°
6	1	$-\frac{360}{147}$	$\frac{450}{147}$	$-\frac{400}{147}$	$\frac{225}{147}$	$-\frac{72}{147}$	$\frac{10}{147}$	$\frac{60}{147}$	17.84°

side is balanced by a source term involving known quantities that will be referred to as right-hand side and denoted by $RHS^{[n]}$.

In order to give an estimate of the error in time, we follow [38, Chapter 10] and rewrite the BDF time discretization (15) of (1) as

$$\bar{\partial}_k \tilde{u}^{[n]} = \mathcal{L}(\tilde{u}^{[n]}) + f^n,$$

with $\tilde{u}^{[n]} \approx u(n\Delta t, x)$, $f^n = f(n\Delta t, x)$. Here $\bar{\partial}_k$ is the backward difference operator given by $\bar{\partial}_k \tilde{u}^{[n]} = \frac{1}{\Delta t} \sum_{j=0}^k \frac{a_{k-j}}{b_k} \tilde{u}^{[n-j]}$. Letting $u^n = u(n\Delta t, x)$, we have

$$\bar{\partial}_k u^n = \mathcal{L}(u^n) + f^n + \tau^n,$$

with $\tau^n = \bar{\partial}_k u^n - \partial_t u^n$ the consistency error. Then Lemma 10.1 and Theorem 10.1 in [38] provide the result reported next.

Theorem 5. Let $k \leq 6$ and assume that the solution u of (1) is sufficiently smooth. Then $\forall n \geq k$

$$\|u^n - \tilde{u}^{[n]}\| \leq c \sum_{j=0}^{k-1} \|u^j - \tilde{u}^{[j]}\| + c \Delta t^k \int_{t_0}^{n\Delta t} \left\| \frac{\partial^{k+1} u}{\partial t^{k+1}} \right\| ds,$$

where c is a positive constant.

From Theorem 5 it is clear that if the starting values (14) are accurate enough, then the k -step BDF method (15) has order k . Thus, a suitable strategy is needed to provide the first $k - 1$ approximations. We refer to this strategy as *starting procedure*. The starting values (14) can be approximated by means of a $(k - 1)$ -step BDF method applied with a reduced stepsize $\Delta t/m$, with an integer, suitably chosen m . As described in the previous sections, following [5] we choose the ELM collocation at given points for the discretization of the steady-state problems. The total error is then the time-marching error term plus the contribution of the ELM collocation error.

In Algorithm 1 we describe the main steps required for the resolution. Since in our experiments we consider $d = 1$, i.e. a

Algorithm 1: ELM collocation for the parabolic PDE with a k -step BDF.

Data: the starting values (14), the number of neurons N , the collocation points x_j , $j = 1, \dots, M$.

Result: the external ELM weights $w_i^{[n]}$, with $i = 1, \dots, N$ and $n = k, \dots, Nt$, which provide the collocated solution $\tilde{u}^{[n]}(x)$ in (7)

INITIALIZATION;

Choose randomly α_i and β_i , $i = 1, \dots, N$, according to (13). ;

DISCRETE COUNTERPART OF THE PARABOLIC OPERATOR ;

Compute the collocation matrix $C \in \mathbb{R}^{M \times N}$ by evaluating the left-hand side in (15) at the internal collocation points and appending the collocated boundary conditions.;

TIME LOOP;

for $n = k : Nt$ **do**

 Compute $RHS^{[n]} \in \mathbb{R}^M$ by evaluating the right-hand side in (15) at the collocation points. ;

 Find $w^{[n]} \in \mathbb{R}^N$ that solves the linear problem $Cw^{[n]} = RHS^{[n]}$.;

end

scalar variable x , we focus on this case. In principle the resolution at a fixed time step can be done with different methods, e.g. one can change the number or the parameters of the involved AFs, or the number and the location of the collocation points. By the way, as can be seen from the pseudo-code (statements 2–4), if no changes are made, the linear problem that has to be solved at each time step involves the same matrix, so that this has to be constructed only once.

4. Numerical experiments

Our test set consists of the following classes of problems:

- (a) stiff parabolic equations,
- (b) problems where the boundary conditions are discontinuous,
- (c) problems where the solution decays very rapidly.

To obtain these problems, we used the one-dimensional heat equation with different diffusion coefficients and boundary conditions, which lead to challenging problems for the stability and accuracy properties of the methods [7]. The exact solution of these problems is known and we computed the approximate solution up to the final time t_f using different values of the fixed time step length Δt . For the resolution at each time step, we used collocation with an ELM function consisting of N neurons. Collocation was done on a space grid of $M = N/2$ equispaced points by evaluating the exact derivatives of the activation functions, as in [5]. In order to strengthen that the method can benefit from the change of the location of the space grid, we also report some results with different space grid points, see Fig. 5. For computing the final error, we used the l_∞ norm of the difference between the approximate and the exact solution evaluated on 5000 equispaced points in the spatial domain at the final time t_f . In four cases, we have reported also the convergence of the trapezoidal rule when applied with the 3-points stencil Finite Difference (FD) method applied. When the comparison is done in the equispaced case, we use 200 equispaced points. In the last case we apply the finite difference using the same points used by our collocation method, see Fig. 5. In all cases we report that the method is unable to achieve high precision, moreover in the case where the points are not equispaced we report divergence of the technique.

All the numerical experiments were performed using MATLAB R2021b. The results were slightly affected by the choice of the parameters of the ELM functions, which were randomly generated as described in Section 2.2. In our computations, these random parameters were obtained by using the MATLAB `randn` function. We found close behaviors with different initializations of `randn`, thus we decided to present the results obtained with `rng(1000)` for reproducibility issues, where `rng` is the function specifying the seed for the random number generator.

In the plots, we report the errors with respect to the number Nt of stationary problems solved, which includes, for a fair comparison with one-step methods, the computational effort due to the starting procedures for the BDF methods. In the tests, we used the starting procedure described in Section 3, with $m = 8$. The choice of $m = 8$ was done in order to have a constant value that gives starting values that do not affect the behavior of our methods, thus was made a-posteriori after considering $m = 2, 4$. We chose to report the number of stationary problems solved because it represents the number of constructed underdetermined linear systems and hence it can be taken as representative of the overall computational cost. Our tests compare the behavior of Backward Euler (BE), Trapezoidal Rule (TR), and BDF of order 2–4 (referred to in the plots as BDF2-BDF4). We did not consider the higher-order methods BDF5 and BDF6 because if a modest number of collocation points is needed - as it is in our case - then the cost of the starting procedure dominates the overall performance.

According with the discussion in Section 2.2, the minimum-norm least-squares solutions of the underdetermined linear systems were computed, by using the MATLAB `lsqminnorm` function with rank tolerance 10^{-15} , which implements the COD algorithm. Of course, if the size of the collocation matrix C is very large, this choice may be computationally very expensive, in terms of both time and memory. However, this is not the case for the experiments reported in this work, which are aimed at a methodological investigation.

Test problem (a) is the following:

$$\begin{cases} \frac{\partial u}{\partial t} = \frac{\partial^2 u}{\partial x^2}, & x \in [0, 1], t \in [0, 1], \\ u(x, 0) = \sin(\pi x) + \sin(\gamma \pi x), & u(0, t) = u(1, t) = 0, \end{cases} \quad (16)$$

whose exact solution, with γ as a parameter, is:

$$u(t, x) = e^{-\pi^2 t} \sin(\pi x) + e^{-\gamma^2 \pi^2 t} \sin(k\pi x).$$

When γ increases, the second addend in the exact solution decays rapidly and oscillates. For this reason, the problem is referred to as stiff when $\gamma > 5$, while it is a “standard” test problem in the other cases.

The first numerical test that we present was carried out with $\gamma = 3$ and regards the convergence of the method used to solve the stationary problems, i.e. the ELM collocation when increasing the number N of neurons. In Fig. 1 we present the error computed at the final time t_f . In the different panels, we consider three different time-marching methods to solve the problem. Two facts can be noticed:

- the convergence is very fast: we can conjecture a spectral convergence;
- the error stops decreasing when it reaches the maximum accuracy of the method in time: in that case, the latter begins to prevail.

The three lines correspond to three different choices of Δt and the fact that they coincide for small numbers of neurons confirms that at the beginning of the convergence history the error of the space discretization prevails. The overall accuracy can be compared with that reported in the error plot in the left panel of Fig. 2, where the same problem is solved. By looking also at the right panel of Fig. 2, corresponding to $\gamma = 5$, we see that on these nonstiff problems our procedure has the expected behavior: the order of convergence is achieved; the TR method slightly over-performs BDF2 because of its smaller error constant; the starting procedure for the BDF methods gives a shift of the initial points, which is negligible when Nt grows; the error reaches its limit, given by the accuracy of the resolution by collocation.

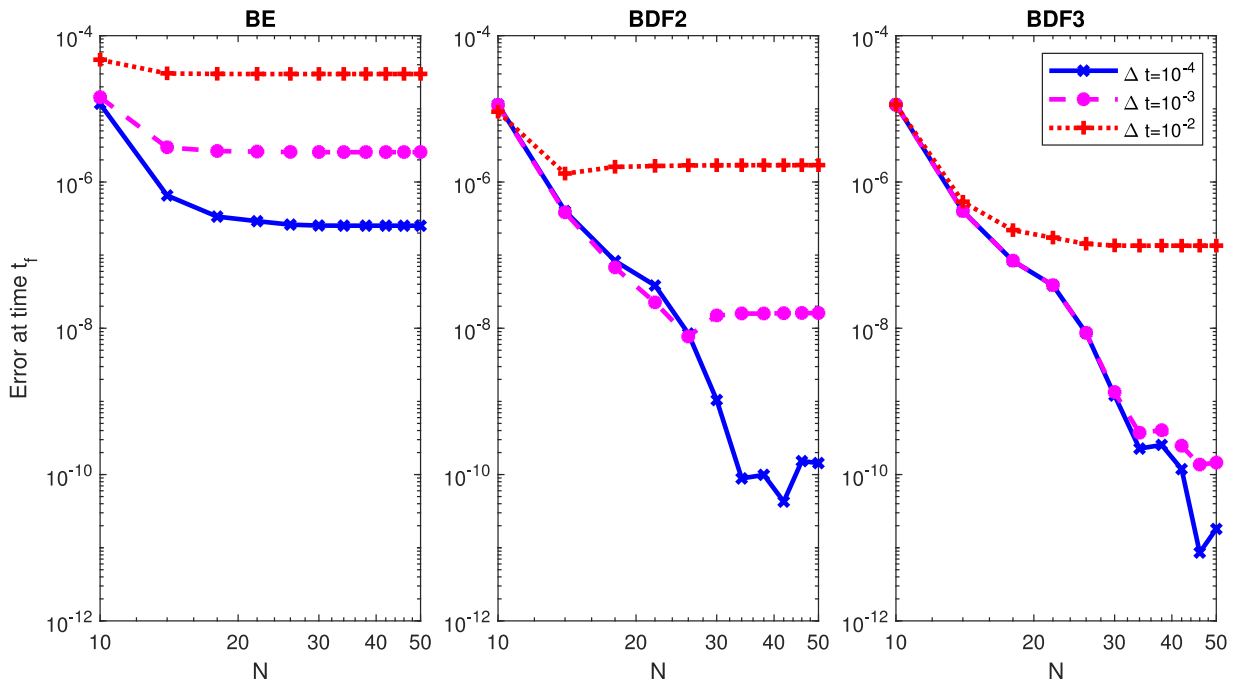


Fig. 1. Computed error for problem (16) with $\gamma = 3$ at the final time. We use the BE, BDF2 and BDF3 methods in time with three different choices of Δt and solve the problem by increasing the number of neurons N .

In Fig. 3 we report the errors obtained while solving Problem (16) with $\gamma = 10$. The errors in the left panel correspond to $N = 40$, while those in the right panel to $N = 50$. This is a stiff problem, for which the trapezoidal method gives poor performances in the case of larger Δt values, see also [7], while the BDF methods show a regular behavior because of their better stability properties. This is also a difficult problem for the collocation method, being the exact solution highly oscillating. For this reason the final accuracy is poor for $N = 40$ and improves for $N = 50$. In Fig. 5 at the left panel we use a spatial grid with $N = 40$ Chebyshev points and test the convergence of the proposed method; in this case we can see that the resolution in space is much more accurate and the error reaches lower values.

Problem (b) (with discontinuous boundary conditions) is:

$$\begin{cases} \frac{\partial u}{\partial t} = \frac{\partial^2 u}{\partial x^2}, & x \in [0, 2], t \in [0, 1.2], \\ u(x, 0) = 1, & u(0, t) = u(2, t) = 0. \end{cases} \tag{17}$$

The exact solution² to problem (17) is:

$$u(t, x) = \sum_{n=1}^{\infty} [1 - (-1)^n] \frac{2}{n\pi} \sin\left(\frac{n\pi x}{2}\right) \exp\left(\frac{-n^2\pi^2 t}{4}\right).$$

Numerical results for this test case are reported in the left panel of Fig. 4. As already noticed in [7], the trapezoidal rule suffers from order reduction, behaving as an order-1 method, while the BDF solvers behave as expected. Moreover, the BDF accuracy in this case is much higher with the same computational cost.

Problem (c) (with a solution that decays very rapidly) is:

$$\begin{cases} \frac{\partial u}{\partial t} = 5 \frac{\partial^2 u}{\partial x^2}, & x \in [0, 1], t \in [0, 1], \\ u(x, 0) = \sin(\pi x), & u(0, t) = u(1, t) = 0, \end{cases} \tag{18}$$

and its solution is:

$$u(t, x) = e^{-5\pi^2 t} \sin(\pi x).$$

Numerical results for this test case are reported in Fig. 4, in the right panel. One can observe once again that the methods can achieve very high accuracy and maintain the properties seen in the standard test cases. For this final problem, we report the convergence also in the case where the space grid is changed at each time step. In Fig. 5 at the right panel we

² The solution is given in the form of a series, but its terms decay very rapidly, so that for our numerical tests the approximate solution obtained with 20 terms is exact up to machine precision.

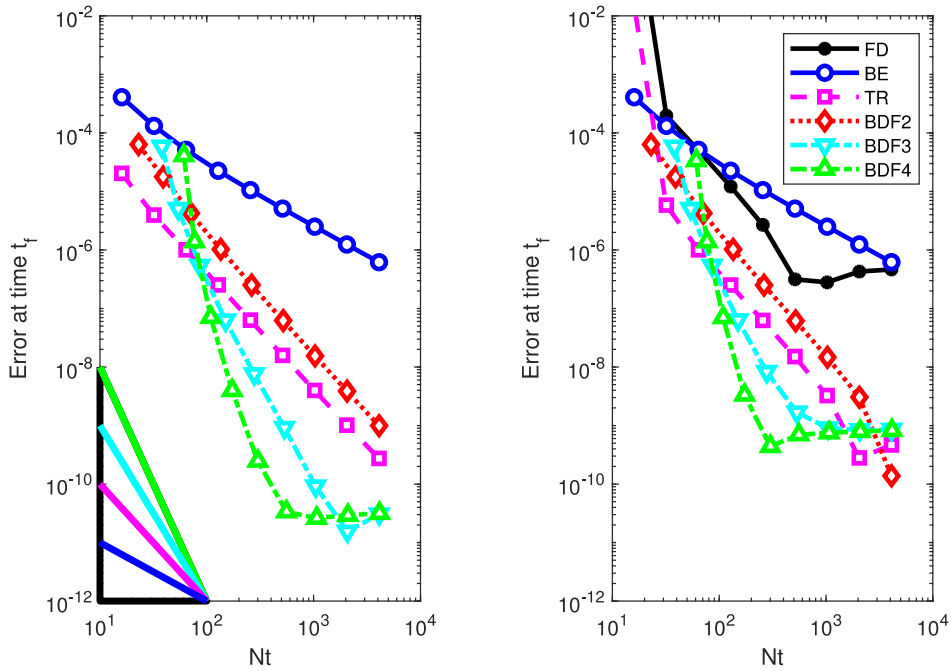


Fig. 2. Computed error for problem (16) with $\gamma = 3$ (left) and $\gamma = 5$ (right) at the final time. The solutions are computed with $N = 40$ neurons, decreasing Δt . In abscissae, we report the number Nt of linear systems solved, in ordinates the absolute value of exact error. On the left panel, the triangles in the bottom-left represent the expected convergence rates. On the right panel, in black, the convergence of the Finite Difference method with trapezoidal rule..

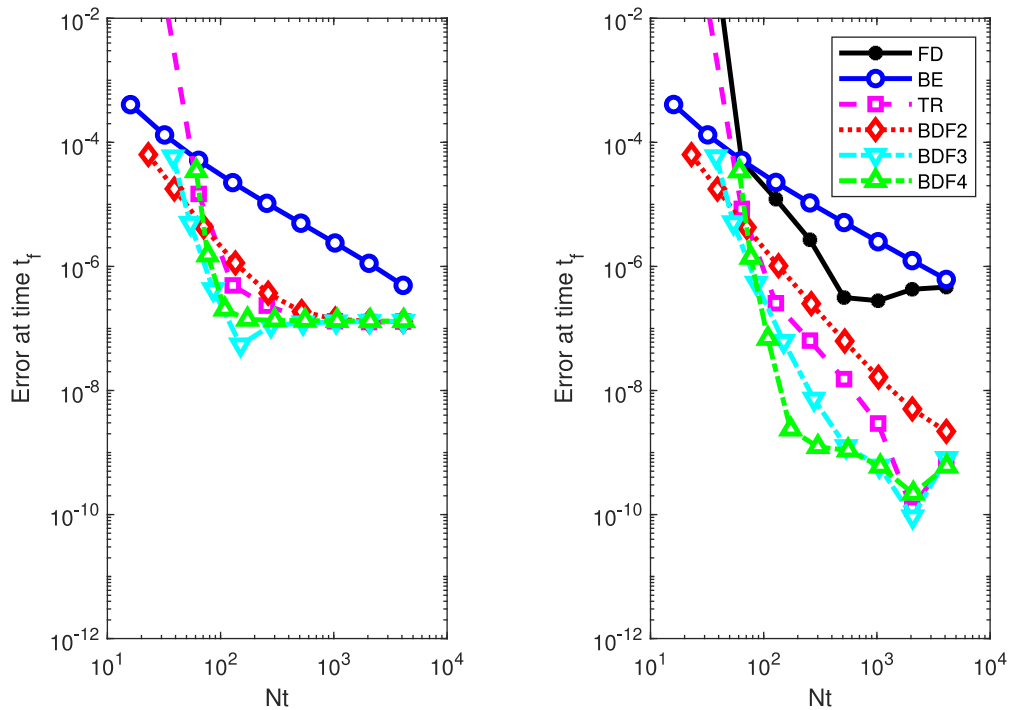


Fig. 3. Computed error for problem (16) with $\gamma = 10$ at the final time. The solutions are calculated with $N = 40$ (left) and $N = 50$ neurons (right), decreasing Δt . In abscissae we report the number Nt of linear systems solved, in ordinates the absolute value of the error. On the right panel, in black, the convergence of the Finite Difference method with trapezoidal rule..

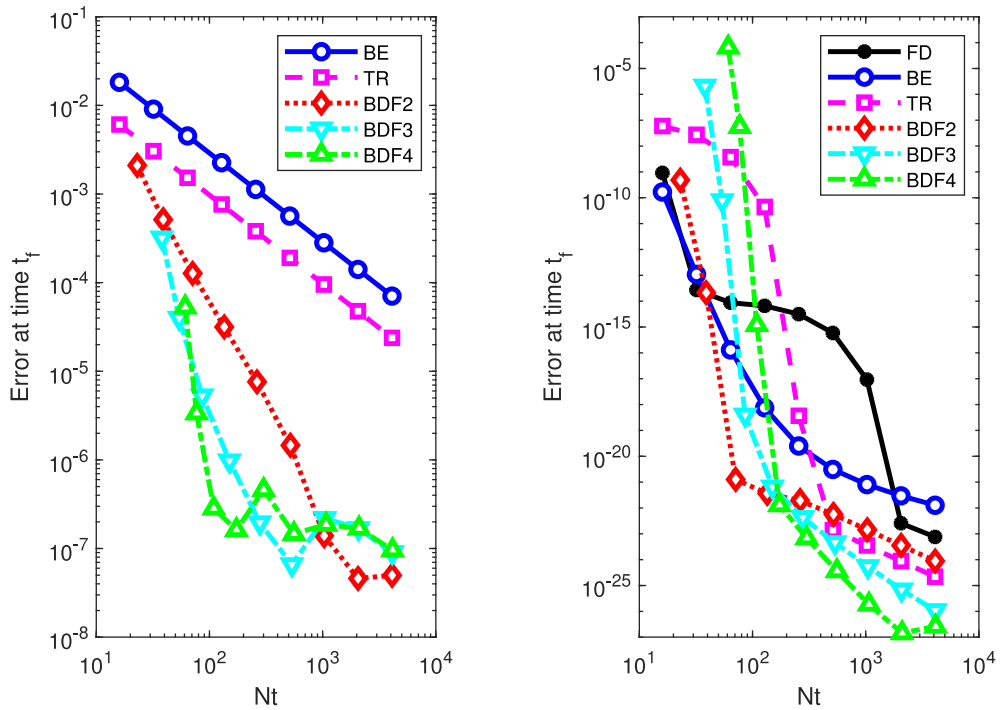


Fig. 4. Computed error for Problem (17) (left panel) and for Problem (18) (right panel) at the final time. The solutions are calculated with $N = 40$ neurons, decreasing Δt . In abscissae we report the number Nt of linear systems solved, in ordinates the absolute value of the error.

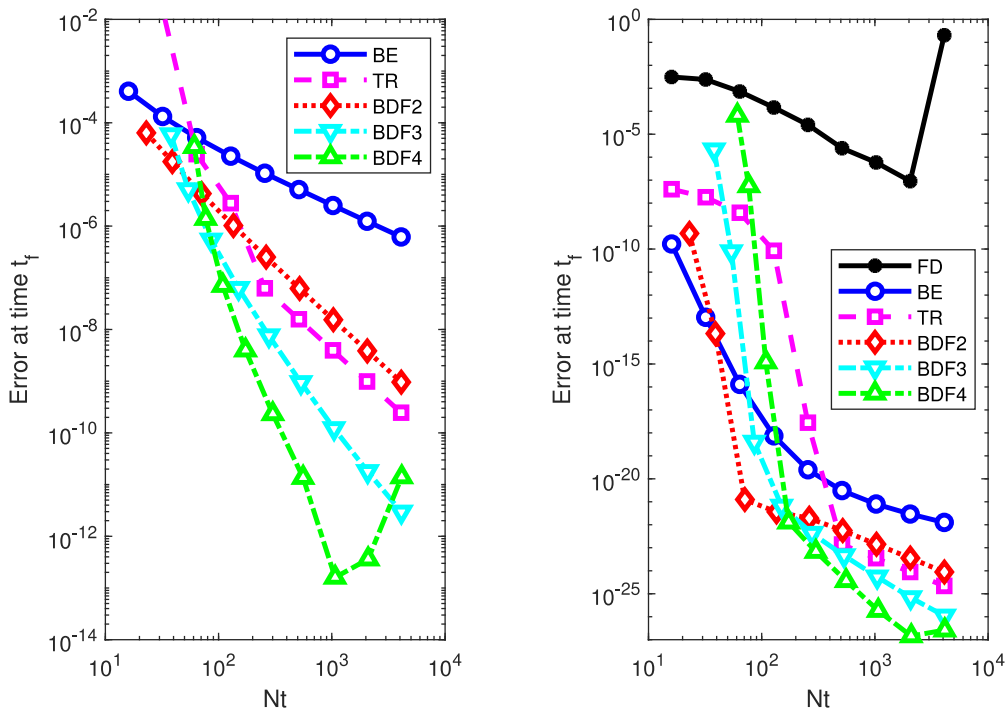


Fig. 5. Computed error at the final time: on the left panel for Problem (16) with $\gamma = 10$ using a space grid of Chebychev points; on the right panel for Problem (18) using a space grid obtained by a random perturbation of equispaced points where the perturbation is changed at each time step. The solutions are calculated with $N = 40$ neurons, decreasing Δt . In abscissae we report the number Nt of linear systems solved, in ordinates the absolute value of the error. The Finite Difference solution on the right panel is computed using a 3-point stencil, employing the identical points utilized for collocation.

consider space grid obtained by a random perturbation of equispaced points where the perturbation is changed at each time step.

5. Conclusions and future work

Scientific Machine Learning (SML) is a research field in which Artificial Intelligence methodologies have been employed to solve innovative manners problems modeled by PDEs. A large number of easy-to-use methods, based on ANNs, allow researchers to deal with complex PDEs efficiently. In this paper, we designed a numerical scheme belonging to the class of ELM methods for solving time-dependent parabolic PDE problems. The proposed approach addresses time evolution by applying time-marching techniques and adopts the collocation for solving the resulting stationary elliptic problems. Regarding the elliptic component of the PDE, we observed good accuracy with a limited computational cost in situations where classical methods fail to give good results.

The main goal of this work was to explore how time approximation affects the resolution of time-dependent PDEs by ELM-based collocation, using classical methods to discretize the problem in time. To this aim, some linear parabolic PDEs that are challenging problems for the stability and accuracy properties of the methods were considered. The θ -method and BDF techniques were investigated. We observed that BDF methods have good accuracy and convergence properties while keeping the same computational cost as the implicit θ -method. Moreover, the time-discretization schemes used in the space-collocation method have promising properties in terms of stability and order of convergence, which are confirmed by numerical tests. We also concluded that, in our numerical framework, BDF methods of order 2–4 were to be preferred to Backward Euler or Trapezoidal rules, because the latter are unable to deal with high accuracy requirements, as they are slowly convergent and can suffer from order reduction or numerical instability.

Data availability

No data was used for the research described in the article.

Acknowledgement

This paper is dedicated to one of the authors, Prof. Daniela di Serafino whose untimely death is a great loss to us all. This work was partially supported by the Istituto Nazionale di Alta Matematica - Gruppo Nazionale per il Calcolo Scientifico (INdAM-GNCS), Italy. The authors thank the editors and reviewers for their valuable suggestions and comments.

References

- [1] P. Behnoudfar, Q. Deng, V. Calo, Higher-order generalized- α methods for parabolic problems, 2021, ArXiv preprint arXiv:2102.05910.
- [2] P. Behnoudfar, G. Loli, A. Reali, G. Sangalli, V. Calo, Explicit high-order generalized- α methods for isogeometric analysis of structural dynamics, *Comput Methods Appl Mech Eng* 389 (2022) 114344.
- [3] A. Björck, Numerical methods for least squares problems, SIAM, Philadelphia (1996), PA.
- [4] L. Bottou, F. Curtis, J. Nocedal, Optimization methods for large-scale machine learning, *SIAM Rev.* 60 (2) (2018) 223–311.
- [5] F. Calabrò, G. Fabiani, C. Siettos, Extreme learning machine collocation for the numerical solution of elliptic PDEs with sharp gradients, *Comput Methods Appl Mech Eng* 387 (2021) 114188.
- [6] W. Cao, X. Wang, Z. Ming, J. Gao, A review on neural networks with random weights, *Neurocomputing* 275 (2018) 278–287.
- [7] J. Cash, Two new finite difference schemes for parabolic equations, *SIAM J Numer Anal* 21 (3) (1984) 433–446.
- [8] S. Cuomo, V. Schiano, D. Cola, F. Giampaolo, G. Rozza, M. Raissi, F. Piccialli, Scientific machine learning through physics-informed neural networks: Where we are and what's next, 2022, ArXiv preprint arXiv:2201.05624.
- [9] G. Cybenko, Approximation by superpositions of a sigmoidal function, *Mathematics of Control, Signals and Systems* 2 (1989) 303–314.
- [10] M.D. Florio, E. Schiassi, F. Calabrò, R. Furfaro, Physics-informed neural networks for 2nd order odes with sharp gradients, *J Comput Appl Math* (2023) 1–10.
- [11] D.d. Serafino, N. Krejić, N. Krklec, N.K. Jerinkić, M. Viola, LSOS: Line-search Second-Order Stochastic optimization methods for nonconvex finite sums, 2021, ArXiv preprint arXiv:2007.15966.
- [12] S. Dong, Z. Li, Local extreme learning machines and domain decomposition for solving linear and nonlinear partial differential equations, *Comput Methods Appl Mech Eng* 387 (2021) 114129.
- [13] V. Dwivedi, B. Srinivasan, Physics informed extreme learning machine (PIELM)-a rapid method for the numerical solution of partial differential equations, *Neurocomputing* 391 (2020) 96–118.
- [14] J. Evans, R. Hiemstra, T. Hughes, A. Reali, Explicit higher-order accurate isogeometric collocation methods for structural dynamics, *Comput Methods Appl Mech Eng* 338 (2018) 208–240.
- [15] G. Fabiani, F. Calabrò, L. Russo, C. Siettos, Numerical solution and bifurcation analysis of nonlinear partial differential equations with extreme learning machines, *J Sci Comput* 89 (2) (2021) 1–35.
- [16] I. Faragó, R. Horváth, A review of reliable numerical models for three-dimensional linear parabolic problems, *Int J Numer Methods Eng* 70 (1) (2007) 25–45.
- [17] D. Gilchrist, J. Evans, A note on the conservation properties of the generalized- α method, 2022, ArXiv preprint arXiv:2202.04568.
- [18] I. Goodfellow, Y. Bengio, A. Courville, Deep learning MIT press, 2016.
- [19] E. Hairer, S.N. rsett, G. Wanner, Solving ordinary differential equations. I. nonstiff problems, volume 8 of Springer Series in Computational Mathematics (1993). Second edition
- [20] C. Higham, D. Higham, Deep learning: an introduction for applied mathematicians, *SIAM Rev.* 61 (4) (2019) 860–891.
- [21] K. Hornik, M. Stinchcombe, H. White, Multilayer feedforward networks are universal approximators, *Neural Networks* 2 (5) (1989) 359–366.
- [22] M. Hosea, L. Shampine, Analysis and implementation of TR-BDF2, *Appl. Numer. Math.* 20 (1–2) (1996) 21–37.
- [23] P. Hough, S. Vavasis, Complete orthogonal decomposition for weighted least squares, *SIAM J. Matrix Anal. Appl.* 18 (2) (1997) 369–392.
- [24] G. Huang, G.-B. Huang, S. Song, K. You, Trends in extreme learning machines: a review, *Neural Networks* 61 (2006) 32–48.

- [25] G.-B. Huang, L. Chen, C. Siew, Universal approximation using incremental constructive feedforward networks with random hidden nodes, *IEEE Trans. Neural Networks* 17 (4) (2006) 879–892.
- [26] G.-B. Huang, Q.-Y. Zhu, C.-K. Siew, Extreme learning machine: theory and applications, *Neurocomputing* 70 (1–3) (2006) 489–501.
- [27] Y. Ito, Nonlinearity creates linear independence, *Adv Comput Math* 5 (1) (1996) 189–203.
- [28] G. Karniadakis, I. Kevrekidis, L. Lu, P. Perdikaris, S. Wang, L. Yang, Physics-informed machine learning, *Nature Reviews Physics* 3 (2021) 422–440.
- [29] A. Kratsios, The universal approximation property: characterizations, existence, and a canonical topology for deep-learning, *Ann Math Artif Intell* 89 (5–6) (2021) 435–469.
- [30] I. Lagaris, A. Likas, D. Fotiadis, Artificial neural networks for solving ordinary and partial differential equations, *IEEE Trans. Neural Networks* 9 (5) (1998) 987–1000.
- [31] H. Mhaskar, T. Poggio, Deep vs. shallow networks: An approximation theory perspective, *Analysis and Applications* 14(06) 829–848.
- [32] A. Pinkus, Approximation theory of the MLP model, *Acta Numerica* 8 (1999) 143–195.
- [33] A. Quarteroni, 2, Springer, 2009.
- [34] H. Ramos, J.V. Aguiar, An almost l-stable bdf-type method for the numerical solution of stiff odes arising from the method of lines, *Numerical Methods for Partial Differential Equations: An International Journal* 23 (5) (2007) 1110–1121.
- [35] W. Schmidt, M. Kraaijveld, R. Duin, Feed forward neural networks with random weights, In *Proceedings 11th IAPR International Conference on Pattern Recognition*. Vol.II. Conference B: Pattern Recognition Methodology and Systems (1992) 1–4.
- [36] S. Sengupta, S. Basak, P. Saikia, S. Paul, V. Tsalavoutis, F. Atiah, V. Ravi, A. Peters, A review of deep learning with special emphasis on architectures, applications and recent trends, *Knowl Based Syst* 194 (2020) 105596.
- [37] V. Thomée, Finite difference methods for linear parabolic equations, *Handbook of Numerical Analysis* 1 (1990) 5–196.
- [38] V. Thomée, Galerkin finite element methods for parabolic problems, *Springer Science & Business Media* 25 (2007).