Neurocomputing 155 (2015) 277–285

Contents lists available at ScienceDirect

Neurocomputing

journal homepage: www.elsevier.com/locate/neucom

A constrained integration (CINT) approach to solving partial differential equations using artificial neural networks

Keith Rudd, Silvia Ferrari*

Laboratory for Intelligent Systems and Controls (LISC), Department of Mechanical Engineering & Materials Science, Duke University, United States

ARTICLE INFO

Article history:
Received 1 July 2013
Received in revised form
30 July 2014
Accepted 30 November 2014
Communicated by R.W. Newcomb
Available online 10 December 2014

Keywords:
Neural networks
Initial-boundary value problem
Galerkin methods
Spectral methods
Partial differential equations
Irregular domains

ABSTRACT

This paper presents a novel constrained integration (CINT) method for solving initial boundary value partial differential equations (PDEs). The CINT method combines classical Galerkin methods with a constrained backpropogation training approach to obtain an artificial neural network representation of the PDE solution that approximately satisfies the boundary conditions at every integration step. The advantage of CINT over existing methods is that it is readily applicable to solving PDEs on irregular domains, and requires no special modification for domains with complex geometries. Furthermore, the CINT method provides a semi-analytical solution that is infinitely differentiable. In this paper the CINT method is demonstrated on two hyperbolic and one parabolic initial boundary value problems with a known analytical solutions that can be used for performance comparison. The numerical results show that, when compared to the most efficient finite element methods, the CINT method achieves significant improvements both in terms of computational time and accuracy.

1. Introduction

This paper presents a novel method for solving initial boundary value partial differential equations (PDEs) using constrained integration (CINT). The method, referred to as CINT, combines traditional Galerkin methods with constrained backpropagation (CPROP) for artificial neural networks (ANNs) [1,2]. Several methods have been proposed for using ANNs to provide a functional representation of numerical solutions to PDEs. One of the advantages of representing numerical PDE solutions by ANNs is that the ANN solution provides a closed-form representation of the solution that is infinitely differentiable. Additionally, the ANN solution is represented by a small number of adjustable parameters that can be modified by incremental training algorithms [3], which require less memory than methods such as finite difference (FD) or finite element (FE) [4,5]. Also, unlike FD or FE tabular solutions, the ANN solution is valid over the entire domain, eliminating the need for interpolation.

Numerous methods have been proposed to solve PDEs using ANNs. One approach is to formulate an error function by applying the differential operator to the ANN and evaluating the resulting function at collocation points within the domain. Incremental training of the network weights is then used to minimize the error function. Initial and boundary conditions have been enforced by adding a penalty term to the error function [6]. As with all penalty function methods [7], this approach can display slow convergence and poor accuracy.

An alternate technique for enforcing boundary conditions is to use a problem-specific ansatz that has been tailored to automatically satisfy the boundary condition, while also containing an ANN that is adjusted to satisfy the PDE. This approach has been shown to solve boundary value problems (BVPs) to a high degree of accuracy [8–10]. However, slow convergence in the iterative training of network weights has been observed in this and other ANN based methods. For example, hundreds [5,10] or thousands [6,11] of training iterations may be needed to converge to the levels of accuracy obtainable by FE or FD methods.

One approach that has been used to overcome slow convergence is to combine ANNs with Galerkin methods to train the network output weights. In [12] the ANN output weights were found using an inner product rather than a more traditional training method, such as backpropogation, genetic algorithm, or particle swarm optimization [13,14]. A similar approach was used in [15,16] to solve the Hamilton–Jacobi–Bellman (HJB) equation, and in [17] to analyze bifurcations of a cellular nonlinear network. In these cases, the ANN output weights were treated as functions of time and an inner product was used to transform the PDEs into systems of ordinary differential equations (ODEs). The output weights were then found by integrating the resulting ODEs. Using

* Corresponding author. Tel.: +1 919 660 5310.
E-mail address: silvia.ferrari@duke.edu (S. Ferrari).

http://dx.doi.org/10.1016/j.neucom.2014.11.058

Galerkin methods for training ANNs has been shown to improve solution accuracy and computational time when compared to traditional training methods [12,15,17]. However, one disadvantage of continuous Galerkin methods is that PDEs solved over non-rectangular or irregular domains typically require a domain transformation [18] or domain decomposition [19], which greatly increase the difficulty of obtaining an approximate solution. A smoothed boundary method was proposed in [20] to overcome the difficulties associated with irregular domains for PDE problems with zero-flux boundary conditions. In smoothed boundary methods the domain is embedded into a box and a smoothing term is used to encode the boundary conditions into a modified PDE that can then be solved using standard Galerkin methods.

This paper presents a novel CINT solution method that is broadly applicable to solving initial boundary value problems (IBVPs) on irregular domains, without the need to perform a domain transformation or decomposition, or modify the PDE as done in [20]. The CINT method is applicable to problems with Dirichlet, Neumann, and/or Robin boundary conditions. Similarly to the inner product based methods in [15,17], the CINT method approximates the solution using a single-layer ANN with time-dependent output weights. But, while the methods in [15,17] do not directly address how boundary conditions are satisfied, the CINT method utilizes CPROP, a technique recently developed for preserving prior knowledge in ANNs [21,22], to constrain the ANN so as to satisfy the boundary conditions.

In this paper, the CINT method is demonstrated on three well-known PDEs: the wave equation with Neumann boundary conditions, the wave equation with Dirichlet boundary conditions, and the heat/diffusion equation with Dirichlet boundary conditions. In each problem, the performance of the CINT method is compared to that of the MATLAB© FE PDE solver. The FE method was chosen for comparison because of its wide use and capability to be applied to problems with complex geometries [20]. The results show that computation time required by the CINT method is about 20% less than that required by the FE method for hyperbolic problems. Although the CINT method requires $O(N \log N)$ computations at each time step, where N is the number of training or collocation points, the CINT method remains stable and accurate with an average time step between two and four times larger than the average time step needed by the MATLAB© FE method.

The following section provides a brief description of spectral and Galerkin methods for IBVPs. The CINT method is presented in Section 3, and the numerical results and comparison with FE methods are presented in Section 4.

2. Background on Galerkin methods

Galerkin methods belong to the class of numerical PDE solution methods known as spectral methods, which approximate the PDE solution using a linear combination of basis functions. Because of their efficiency and accuracy, they are routinely implemented in many applications of PDEs, including fluid dynamics, quantum mechanics, heat conduction, and weather prediction [23–27]. The CINT method combines ANNs with Galerkin methods in order to solve a wide class of IBVPs that include parabolic and hyperbolic type problems with Dirichlet, Neumann, and/or Robin boundary conditions. These PDE problems frequently arise in areas such as fluid mechanics, thermodynamics, and optimal control. This section presents a brief overview of spectral methods for IBVPs and introduces key notation. Throughout this paper, bold lower-case symbols represent vectors, and bold upper-case letters represent matrices.

Let $\mathcal{D}(\cdot)$ and $\mathcal{B}(\cdot)$ represent two spatial differential operators, and let the solution of the IBVP be denoted by $u : \Omega \times [t_0, t_f) \to \mathbb{R}$. The CINT method requires that $\mathcal{D}(\cdot)$ be the Riemann integrable and that $\mathcal{B}(\cdot)$ be linear. Consider the PDE:

$$\frac{\partial^k u(\mathbf{x},t)}{\partial t^k} = \mathcal{D}[u(\mathbf{x},t)], \quad \mathbf{x} \in \Omega, \quad t \in [t_0, t_f) \tag{1}$$

subject to linear boundary conditions:

$$\mathcal{B}[u(\mathbf{x},t)] = f(\mathbf{x},t), \quad \mathbf{x} \in \partial\Omega, \quad t \in [t_0, t_f) \tag{2}$$

and the initial (or terminal) condition(s):

$$\frac{\partial^\ell u}{\partial t^\ell}(\mathbf{x},t_0) = g_\ell(\mathbf{x}), \quad \ell = 0, \ldots, k-1, \quad \mathbf{x} \in \Omega, \quad t \in [t_0, t_f). \tag{3}$$

In spectral methods, the PDE solution u is approximated by a finite sum of linearly independent basis functions. Let $\{\psi_1(\mathbf{x}), \ldots, \psi_Q(\mathbf{x})\}$ denote a set of linearly independent basis functions that belong to a Hilbert space with corresponding output weights $\{\alpha_1(t), \ldots, \alpha_Q(t)\}$. Fourier and Chebyshev polynomials are commonly used bases thanks to the availability of the fast Fourier transformation (FFT) [18]. Thus, it is assumed that the approximate solution to (1)–(3) takes the form

$$\hat{u}(\mathbf{x},t) = \sum_q^Q \psi_q(\mathbf{x})\alpha_q(t). \tag{4}$$

Now, let the inner product of two functions $p(\mathbf{x}), q(\mathbf{x}) \in L^2$ be denoted by $\langle p(\mathbf{x}), q(\mathbf{x})\rangle$ and defined as

$$\langle p(\mathbf{x}), q(\mathbf{x})\rangle \triangleq \int_\Omega p(\mathbf{x})\overline{q(\mathbf{x})}\, d\mathbf{x}. \tag{5}$$

Then, the approximate solution (4) is substituted into (1), and the inner-product (5) can be used to obtain the system of ODEs:

$$\mathbf{A}\frac{\partial^k \boldsymbol{\alpha}(t)}{\partial t^k} = \mathbf{b}[\boldsymbol{\alpha}(t)], \tag{6}$$

where the matrix $\mathbf{A} \in \mathbb{R}^{Q\times Q}$ and vector $\mathbf{b}[\boldsymbol{\alpha}(t)] \in \mathbb{R}^Q$ are defined as

$$\mathbf{A}_{(i,j)} \triangleq \langle \psi_j(\mathbf{x}), \psi_i(\mathbf{x})\rangle = \int_\Omega \psi_j(\mathbf{x})\overline{\psi_i(\mathbf{x})}\, d\mathbf{x}, \tag{7}$$

$$\mathbf{b}_{(i)} \triangleq \langle \mathcal{D}[\hat{u}(t,\mathbf{x})], \psi_i(\mathbf{x})\rangle = \int_\Omega \mathcal{D}[\hat{u}(\mathbf{x},t)]\overline{\psi_i(\mathbf{x})}\, d\mathbf{x}. \tag{8}$$

The ODE initial conditions are given by

$$\mathbf{A}\frac{\partial^\ell \boldsymbol{\alpha}}{\partial t^\ell}(t_0) = \mathbf{q}, \tag{9}$$

where

$$\mathbf{q}_{(i)} \triangleq \langle g_\ell(\mathbf{x}), \psi_i(\mathbf{x})\rangle = \int_\Omega g_\ell(\mathbf{x})\overline{\psi_i(\mathbf{x})}\, d\mathbf{x}. \tag{10}$$

The boundary conditions in (2) are enforced by performing integration by parts on the right-hand side of (6), as shown in [28].

In practice, computing the right-hand side of (6) can be very expensive [29]. In particular, if $\mathcal{D}[\hat{u}(\mathbf{x},t)]$ contains time-varying coefficients or nonlinearities, a convolution of sums must be computed. To simplify the enforcement of the boundary condition and to avoid computing convoluted sums, a pseudo-spectral method is often used. In pseudo-spectral methods, the PDE solution is approximated at a set of collocation points in space and time, such that $\hat{u}_{i,j} \approx u(\mathbf{x}_i, t_j)$. Then, at each time step, t_j, the output weights $\boldsymbol{\alpha}(t_j)$ are obtained by transforming the approximate solution $\hat{u}_{i,j}$ to the spectral domain. The spatial derivatives of $\hat{u}_{i,j}$ are then found by evaluating the partial derivatives of (4) at the collocation points, and computing the right-hand side of (1) at every time step. The values of the approximate solution at collocation points on the boundary are then adjusted to satisfy the boundary conditions in (2). This transformation can be performed in $O(N \log N)$ time, using the FFT. However, when N is large, the integration step size Δt is severely restricted [18]. Thus, another approach to simplifying the basis is to choose functions

that satisfy the boundary conditions at every time step. For example, a Fourier series can be used for the special case in which Ω is a rectangular domain and u is periodic on the boundary $\partial\Omega$.

3. Constrained integration (CINT) method

This section presents the CINT method and its relationship with the classical Galerkin method is reviewed in Section 2. The primary differences between CINT and Galerkin methods are how the boundary conditions (2) are enforced, and how the integrals in (6)–(10) are approximated. The linear combination of basis functions (4) is similar in structure to a feed-forward ANN with a single hidden layer. Using this paradigm, the boundary condition (2) is enforced in the CINT method using a modification of the CPROP training approach for preserving prior knowledge in ANNs [1,2]. CPROP consists of constraining incremental backpropogation training algorithms to satisfy equality constraints by partitioning the ANN such that one set of weights is adjusted to preserve prior knowledge, and another set of weights is adjusted to assimilate new data via backpropogation.

Similar to spectral methods, in the CINT method the solution to (1) is approximated by a feedforward ANN with a single hidden layer. As in [1,2], the ANN is partitioned into two sets of basis functions, where one is used to satisfy the boundary condition (2), and the other set is used to satisfy the PDE (1) as shown in Fig. 1. In this paper, the basis functions used to preserve (2) are radial basis functions (RBFs) denoted by $\{\tilde{\sigma}_1(\mathbf{x}),\ldots,\tilde{\sigma}_{\tilde{Q}}(\mathbf{x})\}$. Gaussian RBFs are used for problems in which the PDE solution u is specified at the Dirichlet boundary:

$$\tilde{\sigma}_i(\mathbf{x}) = \exp(-\gamma\|\mathbf{x}-\mathbf{x}_i\|^2), \tag{11}$$

where the shape parameter, γ, is a positive constant, and $\mathbf{x}_i$ is a collocation point on the boundary and the center of the RBF. For PDE problems with a specified flux or the Neumann condition, the basis functions, $\tilde{\sigma}_i(\mathbf{x})$, are chosen to be sigmoidal functions of the form:

$$\tilde{\sigma}_i(\mathbf{x}) = \exp[-\gamma\|\mathbf{x}-\mathbf{x}_i\|^2]\left\{\frac{\exp[(\mathbf{x}-\mathbf{x}_i)^T\hat{\mathbf{n}}_i]-1}{\exp[(\mathbf{x}-\mathbf{x}_i)^T\hat{\mathbf{n}}_i]+1}\right\}, \tag{12}$$

where $\hat{\mathbf{n}}_i$ is the unit vector normal to the boundary at $\mathbf{x}_i$. The transfer functions in (11) were chosen to enforce the Dirichlet condition because they have local support and contribute little to the derivatives of the solutions at the boundary, $\partial\Omega$, since the derivative of (11) is near zero. Similarly, the transfer functions (12) were chosen to satisfy the Neumann condition because (12) is near zero at its center along the boundary, $\partial\Omega$, and has a derivative normal to the boundary that is equal to one.

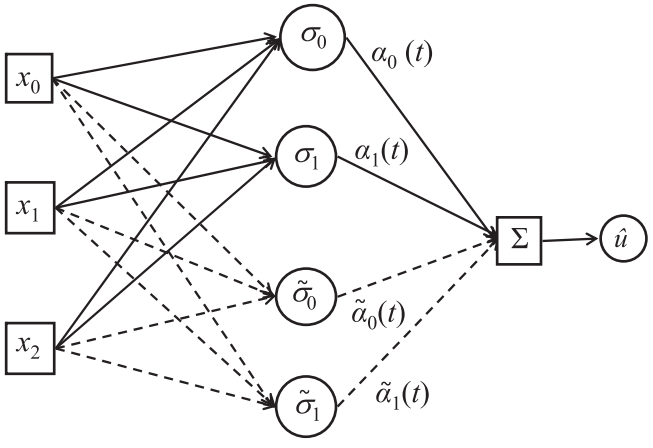

Fig. 1. Simple example of neural network used to approximate CINT solution by partitioning basis/transfer functions into a set $\{\sigma_i\}$ used to satisfy the differential equation, and a set $\{\tilde{\sigma}_j\}$ used to satisfy the boundary conditions.

The CINT transfer functions used to satisfy (1) are denoted by $\{\sigma_1(\mathbf{x}),\ldots,\sigma_Q(\mathbf{x})\}$. Polynomials and Fourier functions were found to work well for the IBVPs solved in Section 4. The ANN representation of the approximate solution of (1) can be written as

$$\begin{aligned}\hat{u}(\mathbf{x},t) &= \sum_{\tilde{q}=1}^{\tilde{Q}} \tilde{\sigma}_{\tilde{q}}(\mathbf{x})\tilde{\alpha}_{\tilde{q}}(t) + \sum_{q=1}^{Q}\sigma_q(\mathbf{x})\alpha_q(t)\\ &= \tilde{\boldsymbol{\sigma}}^T(\mathbf{x})\tilde{\boldsymbol{\alpha}}(t)+\boldsymbol{\sigma}^T(\mathbf{x})\boldsymbol{\alpha}(t)\end{aligned} \tag{13}$$

and substituted into (2). Then the resulting equation is evaluated at a set of training or collocation points on the boundary, $\mathcal{T}_B = \{\mathbf{x}_k|\mathbf{x}_k\in\partial\Omega\}$. The boundary condition is approximately satisfied at these points provided

$$\tilde{\boldsymbol{\alpha}}(t) \approx \tilde{\mathbf{B}}^{+}\left[\mathbf{f}(t)-\mathbf{B}\boldsymbol{\alpha}(t)\right], \tag{14}$$

where the superscript "+" denotes the right pseudo-inverse and,

$$\tilde{\mathbf{B}}_{(i,j)} = \mathcal{B}[\tilde{\sigma}_j(\mathbf{x})]\big|_{\mathbf{x}=\mathbf{x}_i}, \tag{15}$$

$$\mathbf{B}_{(i,j)} = \mathcal{B}[\sigma_j(\mathbf{x})]\big|_{\mathbf{x}=\mathbf{x}_i}, \tag{16}$$

$$\mathbf{f}_{(i)}(t) = f(\mathbf{x}_i,t), \tag{17}$$

for all $\mathbf{x}_i\in\mathcal{T}_B$. Using the right hand side of (14) as ANN output weights in (13) yields an approximate solution that satisfies (2) within a desired tolerance at every time step:

$$\begin{aligned}\hat{u}(\mathbf{x},t) &= \left[\boldsymbol{\sigma}^T(\mathbf{x})-\tilde{\boldsymbol{\sigma}}^T(\mathbf{x})\tilde{\mathbf{B}}^{+}\mathbf{B}\right]\boldsymbol{\alpha}(t)+\tilde{\boldsymbol{\sigma}}^T(\mathbf{x})\tilde{\mathbf{B}}^{+}\mathbf{f}(t)\\ &= \sum_{q=1}^{Q}\psi_q(\mathbf{x})\alpha_q(t)+h(\mathbf{x},t).\end{aligned} \tag{18}$$

Rather than directly computing the inner product in (6), the constrained approximate solution (18) is substituted into the PDE in (1), and evaluated at a set of training or collocation points within the domain, $\mathcal{T}_I=\{\mathbf{x}_i \mid \mathbf{x}_i\in\Omega\}$, producing the following system of ODEs:

$$\mathbf{M}\frac{\partial^k\boldsymbol{\alpha}}{\partial t^k}(t) = \boldsymbol{\xi}[\boldsymbol{\alpha}(t)] \tag{19}$$

$$\mathbf{M}\frac{\partial^\ell\boldsymbol{\alpha}}{\partial t^\ell}(0) = \mathbf{p}_\ell \tag{20}$$

where

$$\mathbf{M}_{(i,j)} \triangleq \psi_j(\mathbf{x}_i), \tag{21}$$

$$\boldsymbol{\xi}_{(i)}[\boldsymbol{\alpha}(t)] \triangleq \left\{\mathcal{D}[\hat{u}(\mathbf{x},t)]-\frac{\partial^k h(\mathbf{x},t)}{\partial t^k}\right\}_{x=x_i} \tag{22}$$

$$\mathbf{p}_{\ell_{(i)}} \triangleq g_\ell(\mathbf{x}_i)-\frac{\partial^\ell h(\mathbf{x}_i,t)}{\partial t^\ell}\bigg|_{t_0} \tag{23}$$

as proven in Appendix A.

We are now ready to state the main theoretical result for the CINT method:

Theorem 1. *If $\mathcal{D}[\hat{u}(\mathbf{x},t)]$ is Lebesgue integrable, then the coefficients, $\boldsymbol{\alpha}$, obtained by solving the Galerkin systems (6)–(10) approach those obtained by solving (19) in the limit as $N\to\infty$.*

Proof. Let Ω be partitioned into N Lebesgue-measurable subdomains, $S_n\subset\Omega$, such that $\mu(S_n\cap S_k)=0$, $\forall n\neq k$, and $\mu(S_n)=\mu(S_k)=\delta\mathbf{x}$, $\forall n,k\in\{1,\ldots,N\}$, where $\mu(S_n)$ is the Lebesgue measure of S_n. Choose $\mathcal{T}_I$ such that $\forall\mathbf{x}_n\in\mathcal{T}_I$, $\mathbf{x}_n\in S_n$. Then, the least squares solution to (19) is equal to the solution found by solving the equivalent system:

$$\mathbf{M}^*\mathbf{M}\boldsymbol{\alpha}\delta\mathbf{x} = \mathbf{M}^*\boldsymbol{\xi}\delta\mathbf{x} \tag{24}$$

where the superscript "$*$" indicates the conjugate transpose. For any element of the matrix $\mathbf{M}^*\mathbf{M}$, the following holds:

$$(\mathbf{M}^*\mathbf{M})_{(i,j)}\delta\mathbf{x} = \sum_{n=1}^{N} \psi_j(\mathbf{x}_n)\overline{\psi_i(\mathbf{x}_n)}\delta\mathbf{x} = \sum_{n=1}^{N} \psi_j(\mathbf{x}_n)\overline{\psi_i(\mathbf{x}_n)}\mu(S_n). \tag{25}$$

The above equation is a Lebesgue sum, and as $N\to\infty$, (25) converges to the limit

$$\lim_{N\to\infty}(\mathbf{M}^*\mathbf{M})_{(i,j)}\delta\mathbf{x} = \int_{\Omega}\psi_j(\mathbf{x})\overline{\psi_i(\mathbf{x})}\,d\mathbf{x} = \langle\psi_j(\mathbf{x}),\psi_i(\mathbf{x})\rangle = \mathbf{A}_{i,j}. \tag{26}$$

Similarly, for any element of the matrix in the right-hand side of (24), the following holds:

$$(\mathbf{M}^*\boldsymbol{\xi})_{(i)}\delta\mathbf{x} = \sum_{n=1}^{N}\left(\mathcal{D}[\hat{u}(\mathbf{x},t)] - \frac{\partial^k h(\mathbf{x},t)}{\partial t^k}\right)\Big|_{\mathbf{x}_n}\overline{\psi_i(\mathbf{x}_n)}\delta\mathbf{x} = \sum_{n=1}^{N}\left(\mathcal{D}[\hat{u}(\mathbf{x},t)] - \frac{\partial^k h(\mathbf{x},t)}{\partial t^k}\right)\Big|_{\mathbf{x}_n}\overline{\psi_i(\mathbf{x}_n)}\mu(S_n). \tag{27}$$

Because (27) is a Lebesgue sum as $N\to\infty$, it converges to the limit

$$\lim_{N\to\infty}(\mathbf{M}^*\boldsymbol{\xi})_{(i)}\delta\mathbf{x} = \int_{\Omega}\left(\mathcal{D}[\hat{u}(\mathbf{x},t)] - \frac{\partial^k h(\mathbf{x},t)}{\partial t^k}\right)\overline{\psi_i(\mathbf{x})}\,d\mathbf{x} = \left\langle\mathcal{D}[\hat{u}(\mathbf{x},t)] - \frac{\partial^k h(\mathbf{x},t)}{\partial t^k},\psi_i(\mathbf{x})\right\rangle = \mathbf{b}_{(i)}. \quad \square \tag{28}$$

In pseudo-spectral methods, the solution is transformed from a pointwise approximate solution, $\hat{u}_{i,j}$, to a *spectral space* where the solution is represented by the output weights, $\alpha_n(t)$. This transformation is typically performed by means of the FFT. In this space, spatial derivatives are computed by taking derivatives of the basis functions. An inverse transformation is then performed and the pointwise representation of the derivatives is incorporated into the temporal integration scheme to explicitly solve for $\hat{u}_{i,j}$ at the next time step.

As in pseudo-spectral schemes, in the CINT method the approximate solution is transformed between the explicit pointwise approximate solution $\hat{u}_{i,j}$ and the spectral representation of the solution at each time step of the temporal integration. However, in the CINT method the temporal integration is not performed on the pointwise solution, $\hat{u}_{jk}$, but on the output weights, $\alpha_n(t)$, of the transfer functions. At each time step, the output weights, $\alpha_n(t)$, are used to reconstruct $\hat{u}_{i,j}$ via (18). The right-hand side of (1) is then evaluated and the result is multiplied by $\mathbf{M}^+$, which gives an approximation of the temporal derivatives of $\boldsymbol{\alpha}(t)$. Finally, these output weights are integrated to find the value of the network output weights at the next time step. This comparison between pseudo-spectral and CINT methods is schematized in Fig. 2.

In the following section, the CINT method is demonstrated on three IBVPs. In the first two problems, the CINT method is applied to a linear, hyperbolic PDE, known as the advection equation, in two spatial dimensions. In the third IBVP, the CINT method is used to solve a linear, parabolic PDE, known as the heat/diffusion equation, in two spatial dimensions. In the first two problems, the CINT method outperforms the FE method both with respect to computational time and accuracy. For the parabolic heat/diffusion equation, the CINT and FE methods have similar performances.

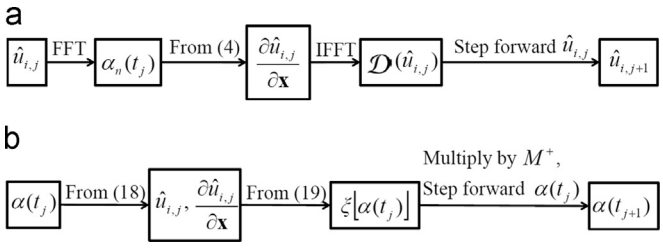

Fig. 2. Diagrams of (a) pseudo-spectral and (b) CINT methods.

4. Simulation and results

The CINT method presented in the previous section is demonstrated on three IBVPs with known analytical solutions. The first problem consists of a 2D wave equation in a circular domain with Dirichlet boundary conditions. The second problem is a 2D wave equation on a square domain and Neumann boundary conditions. The wave equation is a linear hyperbolic PDE that is chosen because of its broad applicability to areas ranging from acoustics [30] to electromagnetics [31]. The third problem is a 2D heat/diffusion equation on a semi-circular domain with Dirichlet boundary conditions. The heat/diffusion equation is a parabolic PDE that has been used to model physical phenomena such as particle and thermal diffusion, as well as processes in finance. These PDEs and their domains were chosen because they each have an analytical solution that can be used to compare the results obtained by CINT and FE methods.

The FE method is chosen for comparison because it is most commonly used to solve IBVPs with complex geometries [20] and to solve the wave and heat/diffusion equations [32–36]. In this paper, the FE method is implemented through the MATLAB[4] PDE Toolbox [34]. Both CINT and FE methods are implemented using the MATLAB© *ODE15s* ODE solver [37] for integrating (19) and the analogous ODE that arises in the FE method. As the speed of the algorithm is determined by the computational complexity and the allowable integration step size, the mean step size, Δt, observed in each method is also reported. Because *ODE15s* uses an adaptive time step, the mean observed step size provides an approximation to the allowable step size for the PDE solvers. The approximation error obtained for the hyperbolic IBVPs is measured using the root mean square (RMS) error, defined as

$$\mathrm{RMS}(t) \triangleq \sqrt{\frac{\sum_{m=1}^{M}[u(\mathbf{x}_m,t)-\hat{u}(\mathbf{x}_m,t)]^2}{M}} \tag{29}$$

for M points in Ω, where $u(\mathbf{x}_m,t)$ is the analytical solution. Because for the parabolic IBVP, $u\to 0$ as $t\to\infty$, a more meaningful measure of solution accuracy is the relative error norm (REN), defined as

$$\mathrm{REN}(t) \triangleq \frac{\sqrt{\sum_{m=1}^{M}[u(\mathbf{x}_m,t)-\hat{u}(\mathbf{x}_m,t)]^2}}{\sqrt{\sum_{p=1}^{P}u^2(\mathbf{x}_p,t)}} \tag{30}$$

The performance comparison in Table 1 shows the cumulative RMS errors for the two IBVPs in Sections 4.1 and 4.2, and the cumulative REN for the IBVP in Section 4.3. The mean time step, $\overline{\Delta t}$, is computed, and τ is the computational time required to solve the problem. It can be seen that in both hyperbolic problems, the CINT method computes the numerical solution significantly faster

Table 1
Performance comparison between CINT and FE methods.

IBVP	Method	RMS/REN	Mean $\overline{\Delta t}$	Time τ (s)
(31)	CINT	0.0018	0.006	6.9
(31)	FE	0.0064	0.003	49.2
(39)	CINT	0.013	0.0046	2.18
(39)	FE	0.11	0.001	38.65
(45)	CINT	0.0018	0.147	3.1
(45)	FE	0.0034	0.128	1.7

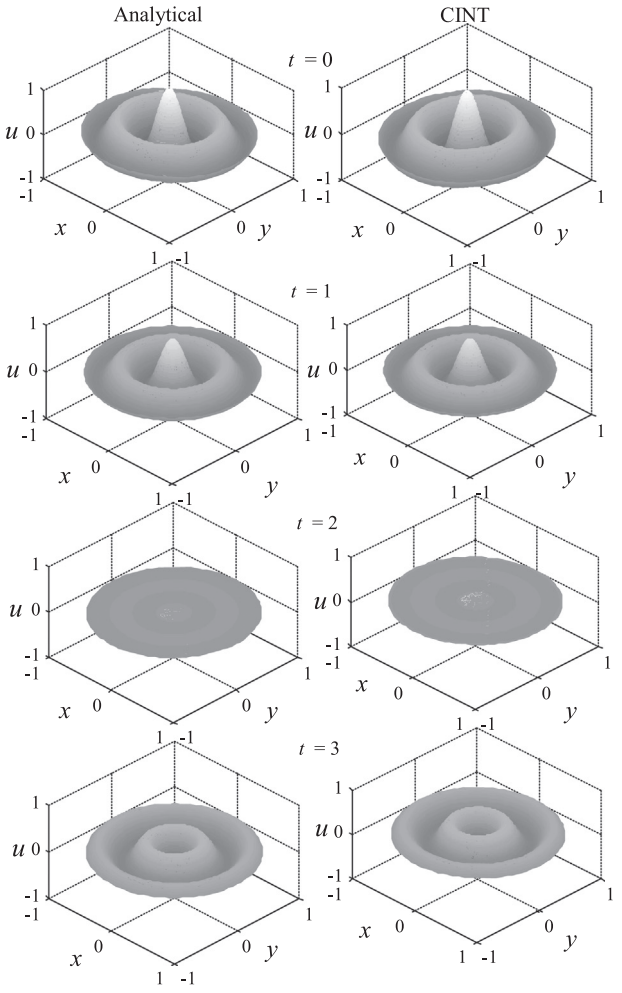

Fig. 3. Analytical (left) and numerical (right) solutions to (31)–(35).

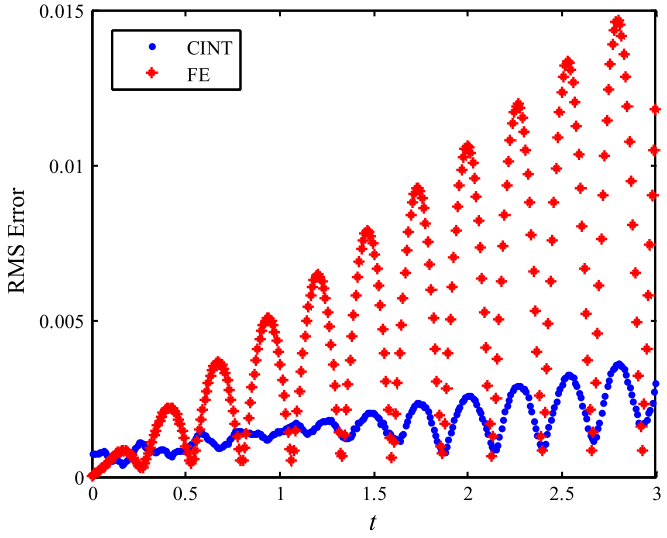

Fig. 4. RMS error in the approximate solutions to (31)–(35) returned by the FE and CINT methods versus t.

than the FE method. In the heat/diffusion equation, the FE method was found to be faster, but in all IBVPs, the solution obtained by CINT was considerably more accurate than the solution obtained by FE method. Examples of CINT ANN parameter values are shown in Appendix B.

4.1. Wave equation with Dirichlet boundary conditions in two dimensions

This section presents the results obtained by solving the two-dimensional wave equation on the circular domain $\Omega=\{(x,y)\mid x^2+y^2\le 1\}$, and the time interval $[0,3]$ s. The wave equation describes the evolution of u, such that

$$\frac{\partial^2 u}{\partial t^2}=c^2\left(\frac{\partial^2 u}{\partial x^2}+\frac{\partial^2 u}{\partial y^2}\right) \tag{31}$$

where the wave speed, c, is chosen to be equal to 2. The above PDE is subject to the Dirichlet boundary conditions:

$$u(x,y,t)=0,\quad \forall (x,y)\in\partial\Omega \tag{32}$$

and the initial conditions:

$$u(x,y,0)=J_0\left(\lambda_4\sqrt{x^2+y^2}\right),\quad \forall (x,y)\in\Omega \tag{33}$$

$$\frac{\partial u}{\partial t}(x,y,0)=0,\quad \forall (x,y)\in\Omega \tag{34}$$

where $J_0(\cdot)$ represents a Bessel function of the first kind, defined as

$$J_0(r)=\sum_{m=0}^{\infty}\frac{(-1)^m}{(m!)^2}\left(\frac{r}{2}\right)^{2m} \tag{35}$$

and λ_4 represents the 4th zero of $J_0(\cdot)$. This IBVP can be solved using the method of separation of variables [38], and has the analytical solution:

$$u(x,y,t)=J_0\left(\lambda_4\sqrt{x^2+y^2}\right)\cos(c\lambda_4 t) \tag{36}$$

The RBFs used to approximate the CINT solution are given by (11), with $\gamma=10$ and $\tilde{Q}=40$. The 40 RBFs are centered at 40 points that are uniformly distributed along the boundary of the domain, $\partial\Omega$. The polynomial basis

$$\sigma_{j,m}(x,y)=x^{j-m}y^m \tag{37}$$

is used for the transfer functions $\sigma_{j,m}(\mathbf{x})$, where $j=0,1,\ldots,14$, and $m=0,1,\ldots,j$. Thus, the ANN used to approximate the solution to the IBVP (31)–(35) is given by

$$\hat{u}(x,y,t)=\sum_{j=0}^{14}\sum_{m=0}^{m=j}x^{j-m}y^m\alpha_{jm}(t)+\sum_{q=1}^{40}e^{-10[(x-x_q)^2+(y-y_q)^2]}\tilde{\alpha}_q(t). \tag{38}$$

Both the analytical and numerical solutions of the IBVP (31)–(35) are shown in Fig. 3. The RMS errors obtained by the FE and CINT methods are plotted in Fig. 4. It can be seen that, initially, the error is slightly larger for the CINT solution. But, over time, the CINT error grows much more slowly than the FE error. Furthermore, the CINT method obtained the numerical solution in approximately 6.9 s, while the MATLAB FE solver required approximately 49.2 s. Thus, the CINT method reduced the computation time by 85%. The cumulative RMS error was 0.0064 for the FE method, and 0.0018 for the CINT method. Thus, the CINT method reduced the RMS by approximately 70%.

4.2. Wave equation with Neumann boundary conditions in two dimensions

This section presents the results obtained for the 2D wave equation on a square domain with Neumann boundary conditions, $\Omega=\{(x,y)|(x,y)\in[-1,1]\times[-1,1]\}$, and a time interval $[0,3]$ s. The

wave equation describes the dynamics of u, such that

$$\frac{\partial^2 u}{\partial t^2} = c^2\left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}\right) \tag{39}$$

where the wave speed, c, is chosen to be equal to 3 (units depend on the specific application). The wave equation (39) is subject to the boundary conditions:

$$\nabla u(x,y,t)\cdot \hat{\mathbf{n}} = 0, \quad (x,y)\in \partial\Omega \tag{40}$$

where $\hat{\mathbf{n}}$ is the outward unit normal vector, and to the initial conditions:

$$u(x,y,0) = \cos(2\pi x)\cos(3\pi y), \quad \forall (x,y)\in\Omega, \tag{41}$$

$$\frac{\partial u}{\partial t}(x,y,0) = 0, \quad \forall (x,y)\in\Omega \tag{42}$$

The analytical solutions of the IBVP in (39)–(42) can be obtained using the method of separation of variables [38], and is given by

$$u(x,y,t) = \cos(2\pi x)\cos(3\pi y)\cos\left(c\pi\sqrt{2^2+3^2}t\right) \tag{43}$$

The RBFs used to approximate the CINT solution are given by (12), with γ=90 and $\tilde{Q}=70$. The 70 RBFs are centered at points distributed uniformly along the boundary. As in Section 4.1, the polynomial basis

$$\sigma_{mn}(\mathbf{x}) = x^m y^n \tag{44}$$

is used for the transfer functions, where $m,n=0,1,\ldots,14$. The analytical and numerical solutions of the IBVP (39)–(42) are shown in Fig. 5.

The CINT and FE RMS errors plotted in Fig. 6 show that, initially, the error is slightly larger for the CINT solution. But, over time, the CINT error grows significantly slower than the FE error. Also, the CINT method required approximately 2.18 s to obtain a solution, while the MATLAB FE solver required approximately 38.65 s. Therefore, the CINT method reduced the computation time by 94%. The cumulative RMS error for the FE was 0.11, while the cumulative RMS error for the CINT method was 0.013. Thus, CINT displayed an error reduction of approximately 88% compared to FE.

4.3. Heat/diffusion equation in two dimensions

This section presents the results obtained by solving the heat/diffusion equation in two spatial dimensions, on a semicircular domain, $\Omega = \left\{(x,y) \mid x\in[-1,1], y\in\left[0,\sqrt{(1-x^2)}\right]\right\}$, and the time interval $[0,5]$ s. The parabolic PDE

$$\frac{\partial u}{\partial t} = k\left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}\right), \quad \mathbf{x}\in\Omega \tag{45}$$

describes dissipation/diffusion of the solution u, where the diffusivity, or rate of diffusion, k, is chosen to be equal to 0.002. The parabolic PDE is subject to the boundary conditions:

$$u(\mathbf{x},t) = 0, \quad \mathbf{x}\in\partial\Omega \tag{46}$$

and the initial conditions:

$$u(\mathbf{x},t_0) = J_3\left(\lambda_3\sqrt{x^2+y^2}\right)\sin[3\arctan(y/x)] \tag{47}$$

where $J_3(\cdot)$ is a Bessel function of the first kind, defined as

$$J_3(r) = \sum_{m=0}^{\infty}\frac{(-1)^m}{m!(m+3)!}\left(\frac{r}{2}\right)^{2m+3} \tag{48}$$

and λ_3 is the 3rd zero of the above Bessel function. The IBVP in (45)–(48) has the analytical solution:

$$u(\mathbf{x},t) = J_3\left(\lambda_3\sqrt{x^2+y^2}\right)\sin[3\arctan(y/x)]e^{-k\lambda_3^2 t} \tag{49}$$

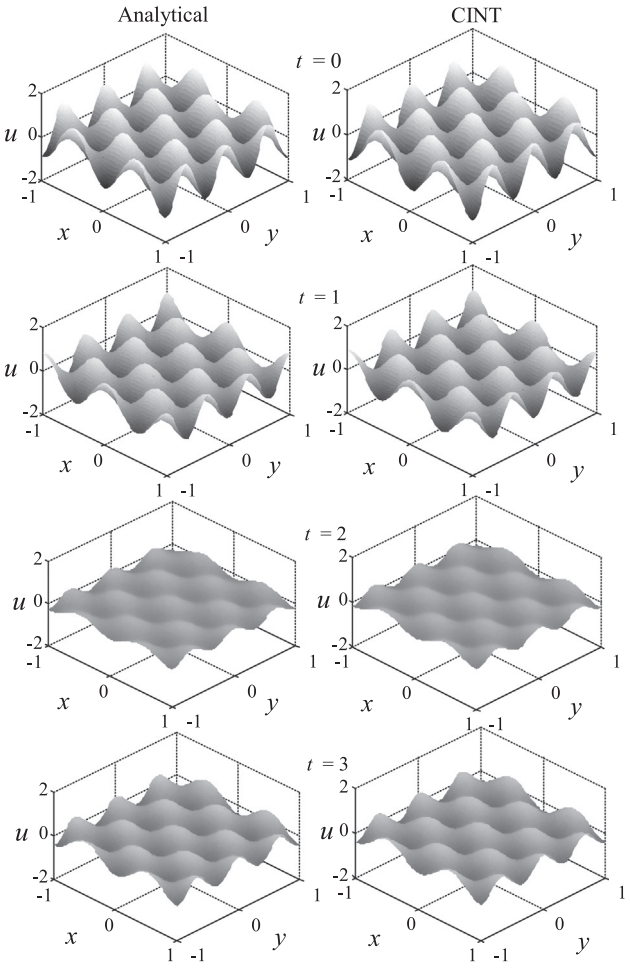

Fig. 5. Analytical (left) and numerical (right) solutions to (39)–(42).

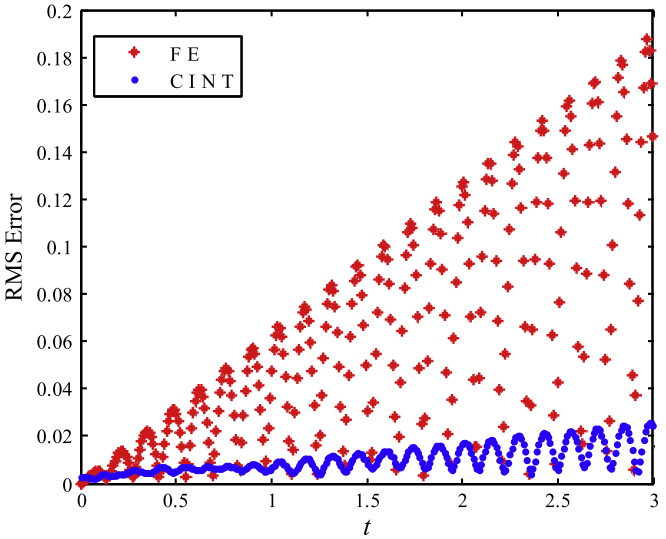

Fig. 6. RMS error in the solutions to (39)–(42) returned by the FE and CINT methods over time.

which may be obtained using the method of separation of variables [38].

The RBFs used to approximate the CINT solution are given by (11), with γ=20 and $\tilde{Q}=60$. The 60 RBFs are centered at 60 points

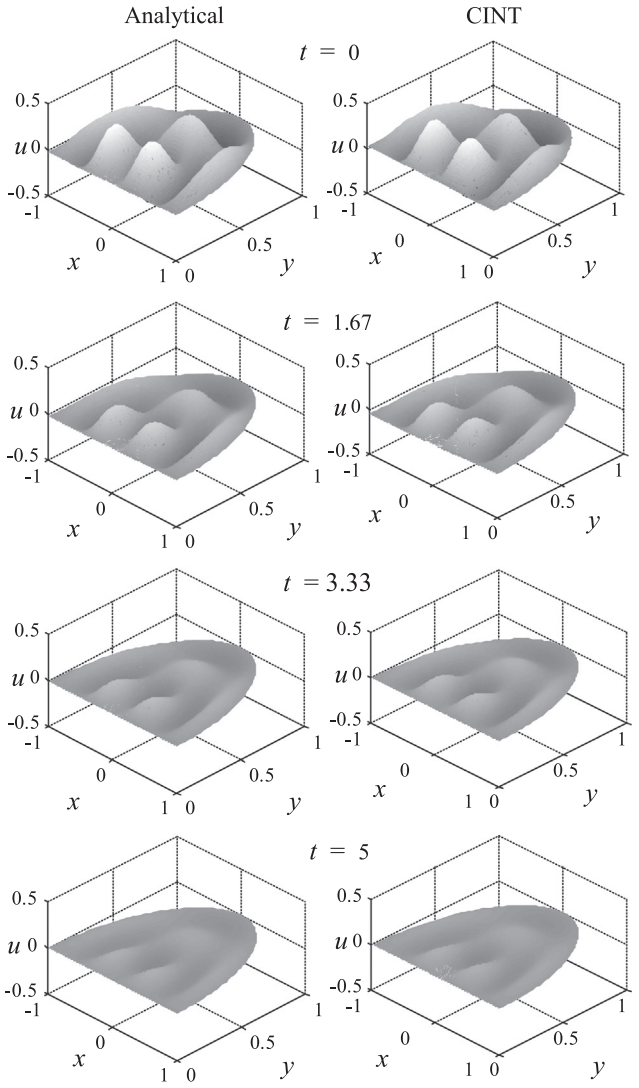

Fig. 7. Analytical (left) and numerical (right) solutions to (45)–(48).

distributed uniformly over the boundary $\partial\Omega$, and a truncated Fourier basis was used for the transfer functions:

$$\sigma_j(\mathbf{x}) \in \{1, \sin(\pi x), \cos(\pi x), \ldots, \sin(5\pi x), \cos(5\pi x)\} \otimes \{1, \sin(\pi y), \cos(\pi y), \ldots, \sin(5\pi y), \cos(5\pi y)\}, \quad (50)$$

where $\otimes$ denotes the tensor product.

It can be seen that the error plot for the third example (45), shown in Fig. 8, is substantially different from the errors observed in Figs. 4–6. The difference is due to the dissimilarities between the heat equation, a parabolic equation with a solution that decays to zero, and the wave equation, a hyperbolic equation with an oscillating solution. Because the heat equation (45) with zero boundary condition converges to zero, both CINT and FE methods produce an error that decays very rapidly and are dominated by errors at $t=0$. Thus, the REN (30) provides a more meaningful performance measure of solution accuracy over time.

The analytical and numerical solutions of (45)–(48) are shown in Fig. 7. The FE method required approximately 0.6 s, while CINT required 1.7 s. The REN error comparison plotted in Fig. 8 shows that, although initially more accurate, over time the FE error becomes significantly larger than the CINT error, ultimately reaching an error that is one order of magnitude higher than the CINT error by the final time.

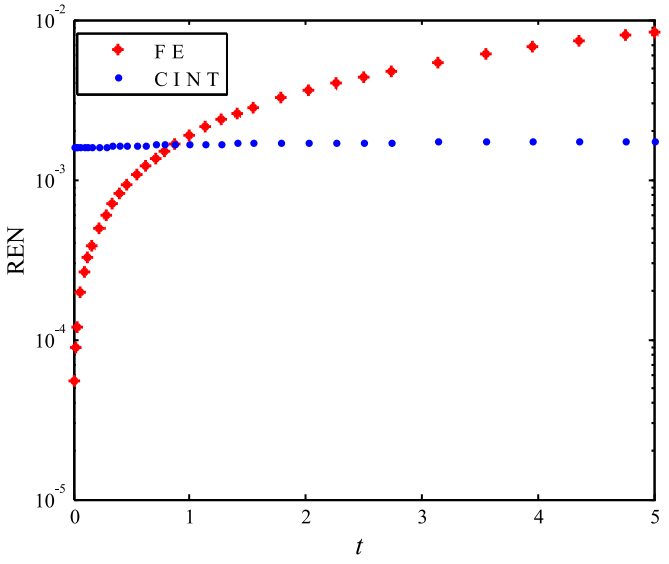

Fig. 8. REN observed in the FE and CINT numerical solutions to (45)–(48).

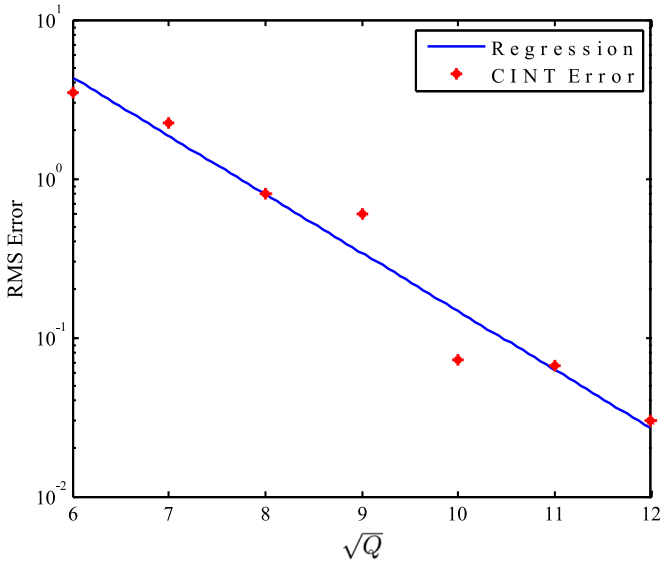

Fig. 9. Exponential convergence of CINT method for the approximate solution of (18). (For interpretation of the references to color in this figure caption, the reader is referred to the web version of this paper.)

4.4. CINT performance analysis

The rate of convergence of the CINT method is determined by the number of basis functions chosen to satisfy a user-defined error tolerance. Assuming that the right pseudo-inverse $\mathbf{M}^+$ has been pre-computed and stored, each step of the ODE solver requires $O(NQ)$ computations. It has been shown in [39] that, given an analytical solution u, classical spectral methods converge exponentially in Q, i.e.,

$$||u(\mathbf{x},t) - \hat{u}(\mathbf{x},t)|| \leq c_1 e^{-c_2 Q} \quad (51)$$

where c_1 and c_2 are positive constants. Exponential convergence was also observed in the CINT method. Fig. 9 shows how the RMS error decreases with the highest degree of polynomial used to solve the IBVP given in Section 4.2 (or $\sqrt{Q}$ as the PDE is solved in two dimensions). The CINT errors computed in simulations are

plotted by red stars, and the exponential regression is plotted in a solid line representing an $RMS \approx 692 \exp(-0.85\sqrt{Q})$.

5. Summary and conclusions

Numerous authors have proposed using ANNs to obtain a functional representations of numerical PDE solutions and, thus, circumvent the need for interpolation or numerical differentiation. However, traditional training methods have shown to exhibit slow convergence and poor accuracy compared to other numerical methods of solutions such as FE and FD. This paper presents a novel CINT method that transforms the PDE into an ODE, similar to Galerkin methods, and uses a novel CPROP training algorithm to satisfy the boundary conditions. As a result, the CINT method achieves very high convergence rates and accuracy, while also being applicable to non-rectangular domains.

The numerical results presented in this paper show that, for hyperbolic PDEs, the CINT method outperformed MATLAB© FE solver in both speed and accuracy. For the wave equation with Dirichlet boundary conditions, the CINT method reduced the computation time by 85%, and the approximation error by 70%. For the wave equation with Neumann boundary conditions, the CINT method reduced the computation time by 94%, and the approximation error by 88%. Furthermore, the CINT method was shown to exhibit exponential convergence rates, similar to Galerkin methods.

Similar to classical Galerkin methods, the present version of the CINT method is not expected to perform well for problems with discontinuities or sharp gradients, such as supersonic fluid flow PDE problems involving shock waves. Because these PDE solutions are not smooth, convergence is no longer expected to be exponential and may require many more basis/transfer functions, potentially becoming computationally too intensive. Therefore, these problems will be the topic of future research.

Acknowledgmens

This work was supported by the National Science Foundation under ECCS grant No. 0823945.

Appendix A. Derivation of CINT approximate solution

The first step in obtaining the CINT approximate solution $\hat{u}$ in (18) is to apply the boundary condition (2) to the ansatz (13), which gives

$$f(\mathbf{x},t) = \mathcal{B}\left[\sum_{\tilde{q}=1}^{\tilde{Q}} \tilde{\sigma}_{\tilde{q}}(\mathbf{x})\tilde{\alpha}_{\tilde{q}}(t) + \sum_{q=1}^{Q} \sigma_q(\mathbf{x})\alpha_q(t)\right]$$
$$= \sum_{\tilde{q}=1}^{\tilde{Q}} \mathcal{B}[\tilde{\sigma}_{\tilde{q}}(\mathbf{x})]\tilde{\alpha}_{\tilde{q}}(t) + \sum_{q=1}^{Q} \mathcal{B}[\sigma_q(\mathbf{x})]\alpha_q(t) \quad (A.1)$$

The above equation is then evaluated at a set of training or collocation points on the boundary, $\mathcal{T}_B = \{\mathbf{x}_k | \mathbf{x}_k \in \partial\Omega\}$, and organized into a set of linear equations:

$$\mathbf{f}(t) = \tilde{\mathbf{B}}\tilde{\boldsymbol{\alpha}}(t) + \mathbf{B}\boldsymbol{\alpha}(t) \quad (A.2)$$

where, as defined in Section 3,

$$\tilde{\mathbf{B}}_{(k,j)} = \mathcal{B}[\tilde{\sigma}_j(\mathbf{x})]\big|_{\mathbf{x}=\mathbf{x}_k} \quad (A.3)$$

$$\mathbf{B}_{(k,j)} = \mathcal{B}[\sigma_j(\mathbf{x})]\big|_{\mathbf{x}=\mathbf{x}_k} \quad (A.4)$$

$$\mathbf{f}_{(k)}(t) = f(\mathbf{x}_k, t) \quad (A.5)$$

for all $\mathbf{x}_k \in \mathcal{T}_B$.

If the number of RBFs is less than the number of collocation points on the boundary, $\tilde{Q} < |\mathcal{T}_B|$, then (A.2) is an over-determined set of linear equations. In this case, a least-squares approximation of $\tilde{\boldsymbol{\alpha}}$ can be obtained by means of the right pseudo-inverse of $\tilde{\mathbf{B}}$:

$$\tilde{\mathbf{B}}^{+} = (\tilde{\mathbf{B}}^{*}\tilde{\mathbf{B}})^{-1}\tilde{\mathbf{B}}^{*} \quad (A.6)$$

In practice, the Moore–Penrose pseudo-inverse computed using singular value decomposition (SVD) has been found to work well. This method of computing the pseudo-inverse was chosen for its ability to handle ill-conditioned matrices by setting a tolerance for nonzero singular values. For the examples in this paper, the tolerance was set to 10^{-4}. For more information on using the SVD to compute the pseudo-inverse of a matrix the reader is referred to [40]. Now, taking the right pseudo-inverse of (A.2) provides the output weights in the ANN solution:

$$\tilde{\boldsymbol{\alpha}}(t) \approx \tilde{\mathbf{B}}^{+}\left[\mathbf{f}(t) - \mathbf{B}\boldsymbol{\alpha}(t)\right] \quad (A.7)$$

and the result is the approximate solution (18), which satisfies the boundary condition (2) to within a desired tolerance at every integration time step.

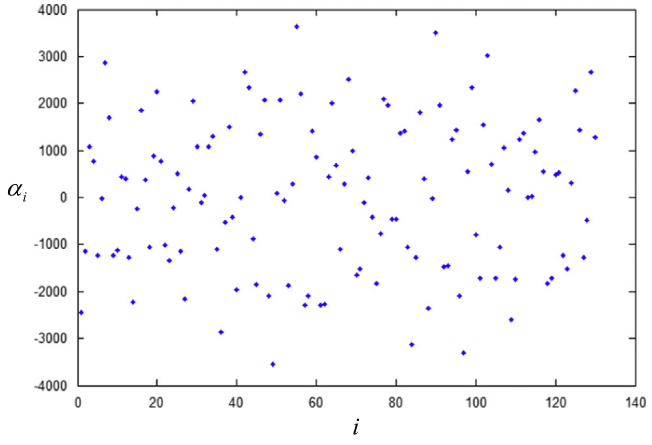

Fig. B1. ANN output weights for CINT solution to (31)–(35).

Table B1
Parameters used in the ANN solution in the CINT method.

Symbol	Description	Example 1	Example 2	Example 3
γ	Adjusts the support of RBFs (11) and (12)	10	90	20
Q	Number of transfer functions in approximate solution (13)	120	225	121
$\tilde{Q}$	Number of RBFs in approximate solution (13)	40	70	60

Appendix B. Neural network parameters

The ANN parameters used in the three numerical examples presented in Section 4, along with a brief description, are shown in Table B1. The values of the output weights for the solution of the wave equation with Dirichlet boundary conditions (Section 4.1) are plotted in Fig. B1.

References

[1] S. Ferrari, M. Jensenius, A constrained optimization approach to preserving prior knowledge during incremental training, IEEE Trans. Neural Netw. 19 (6) (2008) 996–1009.
[2] K. Rudd, G.D. Muro, S. Ferrari, A constrained optimization approach for the adaptive solution of partial differential equations, IEEE Trans. Neural Netw. Learn. Syst. 25 (3) (2014) 571–584.
[3] MATLAB Neural Network Toolbox, User's Guide, The MathWorks, 2005.
[4] I. Lagaris, A. Likas, D. Papageorgio, Neural-network methods for boundary value problems with irregular boundaries, IEEE Trans. Neural Netw. 11 (5) (2000) 1041–1049.
[5] Y. Shirvany, M. Hayati, R. Moradian, Multilayer perceptron neural networks with novel unsupervised training method for numerical solution of the partial differential equations, Appl. Soft Comput. 9 (1) (2009) 20–29.
[6] M.W.M.G. Dissanayake, N. Phan-Thien, Neural-network-based approximations for solving partial differential equations, Commun. Numer. Methods Eng. 10 (1994) 195–201.
[7] R. Fletcher, S. Leyffer, Nonlinear programming without a penalty function, Math. Program. 91 (2) (2002) 239–269.
[8] I. Lagaris, A. Likas, D.I. Fotiadis, Artificial neural networks for solving ordinary and partial differential equations, IEEE Trans. Neural Netw. 9 (5) (1998) 987–1000.
[9] R. Shekari Beidokhti a Malek, Solving initial-boundary value problems for systems of partial differential equations using neural networks and optimization techniques, J. Frankl. Inst. 346 (2009) 898–913.
[10] K.S. McFall, J.R. Mahan, Artificial neural network method for solution of boundary value problems with exact satisfaction of arbitrary boundary conditions, IEEE Trans. Neural Netw. 20 (8) (2009) 1221–1233.
[11] M. Hüsken, C. Goerick, A. Vogel, Fast adaptation of the solution of differential equations to changing constraints, in: ICSC Symposium on Neural Computation, 2000.
[12] A.J. Meade, A. Fernandez, Solution of nonlinear ordinary differential equations by feedforward neural networks, Math. Comput. Model. 20 (9) (1994) 19–44.
[13] B. Luitel, G. Venayagamoorthy, Particle swarm optimization with quantum infusion for the design of digital filters, in: Swarm Intelligence Symposium, 2008, SIS 2008, IEEE, Hoboken, NJ, 2008, pp. 1–8.
[14] R. Xu, J. Xu, D. Wunsch, Clustering with differential evolution particle swarm optimization, in: IEEE Congress on Evolutionary Computation (CEC), IEEE, Hoboken, NJ, 2010, pp. 1–8.
[15] T. Cheng, F.L. Lewis, M. Abu-Khalaf, Fixed-final-time-constrained optimal control of nonlinear systems using neural network HJB approach, IEEE Trans. Neural Netw. 18 (6) (2007) 1725–1737.
[16] Z. Chen, S. Jagannathan, Generalized Hamilton–Jacobi–Bellman formulation-based neural network control of affine nonlinear discrete-time systems, Neural Netw. 19 (1) (2008) 90–106.
[17] E. Javidmanesh, Z. Afsharnezhad, S. Effati, Bifurcation analysis of a cellular nonlinear network model via neural network approach, Neural Comput. Appl. (2013) 1–6.
[18] D. Kopriva, Implementing Spectral Methods for Partial Differential Equations: Algorithms for Scientists and Engineers, Scientific Computation, Springer, New York, NY, 2009.
[19] S.A. Orszag, Spectral methods for problems in complex geometries, J. Comput. Phys. 37 (1) (1980) 70–92.
[20] A. Bueno-Orovio, V. Perez-Garcia, F.H. Fenton, Spectral methods for partial differential equations in irregular domains: the spectral smoothed boundary method, SIAM J. Sci. Comput. 28 (3) (2006) 886–915.
[21] S. Ferrari, M. Jensenius, A constrained optimization approach to preserving prior knowledge during incremental training, IEEE Trans. Neural Netw. 19 (6) (2008) 996–1009.
[22] G.D. Muro, S. Ferrari, A constrained-optimization approach to training neural networks for smooth function approximation and system identification, in: IJCNN'08, 2008, pp. 2353–2359.
[23] C. Canuto, Spectral methods in fluid dynamics, Springer Series in Computational Physics, Springer-Verlag, New York, NY, 1988.
[24] D. Gottlieb, S. Orszag, Numerical analysis of spectral methods: theory and applications, in: CBMS-NSF Regional Conference Series in Applied Mathematics, Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, 1977.
[25] R. Peyret, Spectral methods for incompressible viscous flow, in: Applied Mathematical Sciences, vol. 148, Springer, New York, NY, 2002.
[26] I. Iliev, E. Khristov, K. Kirčev, Spectral methods in soliton equations, Monographs and Surveys in Pure and Applied Mathematics Series, Longman Scientific & Technical, Essex, England, 1994.
[27] B. Guo, Spectral Methods and Their Applications, World Scientific Publishing Company, Singapore, 1998.
[28] C. Canuto, Y. Hussaini, A. Quarteroni, Spectral Methods: Fundamentals in Single Domains, Scientific Computation, Deutsches MAB-Nationalkomitee beim Bundesministerium für Umwelt, Naturschutz und Reaktorsicherheit, 2006.
[29] L. Howle, A comparison of the reduced Galerkin and pseudo-spectral methods for simulation of steady Rayleigh–Bénard convection, Int. J. Heat Mass Transf. 39 (12) (1996) 2401–2407.
[30] A. Pierce, Acoustics: An Introduction to its Physical Principles and Applications, Acoustical Society of America, American Institute of Physics, Woodbury, NY, 1989.
[31] R. Shevgaonkar, Electromagnetic Waves, Electrical & Electronic Engineering Series, McGraw-Hill Education (India) Pvt Limited, New Delhi, DELHI, India, 2005.
[32] M. Ainsworth, P. Monk, W. Muniz, Dispersive and dissipative properties of discontinuous Galerkin finite element methods for the second-order wave equation, J. Sci. Comput. 27 (1–3) (2006) 5–40.
[33] N.N. Abboud, P.M. Pinsky, Finite element dispersion analysis for the three-dimensional second-order scalar wave equation, Int. J. Numer. Methods Eng. 35 (6) (1992) 1183–1218.
[34] Partial Differential Equation Toolbox, User's Guide, MathWorks, Inc., 2013.
[35] T. Hughes, The Finite Element Method: Linear Static and Dynamic Finite Element Analysis, Dover Publications, Mineola, NY, Incorporated, 2012.
[36] R. Verfürth, A posteriori error estimates for finite element discretizations of the heat equation, CALCOLO 40 (3) (2003) 195–212.
[37] L. Shampine, M. Reichelt, The matlab ode suite, SIAM J. Sci. Comput. 18 (1) (1997) 1–22.
[38] R. Haberman, Applied Partial Differential Equations: With Fourier Series and Boundary Value Problems, Pearson Education, Limited, Upper Saddle River, NJ, 2012, URL: ⟨http://books.google.com/books?id=hGNwLgEACAAJ⟩.
[39] J. Boyd, Large-degree asymptotics and exponential asymptotics for Fourier, Chebyshev and Hermite coefficients and Fourier transforms, J. Eng. Math. 63 (2–4) (2009) 355–399.
[40] L. Trefethen, D. Bau, Numerical Linear Algebra, Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, 1997. URL: ⟨http://books.google.com/books?id=5Y1TPgAACAAJ⟩.

Keith Rudd was born in Salt Lake City, UT, USA. He received the B.S. degree in mathematics from Brigham Young University, Provo, UT, USA, in 2007, the M.S. degree in applied mathematics from Northwestern University, Evanston, IL, USA, in 2008, and the Master of Engineering Management degree and the Ph.D. degree from Duke University, Durham, NC, USA, in 2010 and 2013, respectively.

Silvia Ferrari is Paul Ruffin Scarborough Associate Professor of Engineering and Computer Science at Duke University, where she directs the Laboratory for Intelligent Systems and Controls (LISC) and the NSF IGERT on Wireless Intelligent Sensor Networks (WISeNet). Her principal research interests include robust adaptive control of aircraft, learning and approximate dynamic programming, and optimal control of mobile sensor networks. She received the B.S. degree from Embry-Riddle Aeronautical University and the M.A. and Ph.D. degrees from Princeton University. She is a senior member of the IEEE, and a member of ASME, SPIE, and AIAA. She is the recipient of the ONR young investigator award (2004), the NSF CAREER award (2005), and the Presidential Early Career Award for Scientists and Engineers (PECASE) award (2006).