A Constrained Backpropagation Approach for the Adaptive Solution of Partial Differential Equations

Keith Rudd, Gianluca Di Muro, and Silvia Ferrari, *Senior Member, IEEE*

***Abstract*— This paper presents a constrained backpropagation (CPROP) methodology for solving nonlinear elliptic and parabolic partial differential equations (PDEs) adaptively, subject to changes in the PDE parameters or external forcing. Unlike existing methods based on penalty functions or Lagrange multipliers, CPROP solves the constrained optimization problem associated with training a neural network to approximate the PDE solution by means of direct elimination. As a result, CPROP reduces the dimensionality of the optimization problem, while satisfying the equality constraints associated with the boundary and initial conditions exactly, at every iteration of the algorithm. The effectiveness of this method is demonstrated through several examples, including nonlinear elliptic and parabolic PDEs with changing parameters and nonhomogeneous terms.**

***Index Terms*— Adaptive algorithm, artificial neural networks (ANNs), partial differential equations (PDEs), scientific computing.**

I. Introduction

ARTIFICIAL neural networks (ANNs) are commonly implemented to obtain functional representations of partial differential equation (PDE) solutions that are amenable to mathematical analysis, and efficient processing by data assimilation and estimation algorithms. Examples range from solving the Poisson equation [1], to solving the Hamilton–Jacobi–Bellman equation for finite-horizon optimal control problems [2]. In many applications, however, the PDEs describe dynamic processes that are subjected to change. Therefore, while a given PDE problem may capture a dynamic process on short time scales, the process, and thus the PDE, may be undergo changes in the parameters and external forcing. An adaptive method can respond to these changes by adapting the solution incrementally to satisfy the PDE problem with respect to changing parameters and/or a changing nonhomogeneous term.

Given a PDE with fixed parameters and fixed initial and boundary conditions (I/BCs), finite difference methods (FDMs) and finite element methods (FEMs) determine an approximate solution at a discrete number of points, producing a lookup table that can be interpolated when the solution is needed elsewhere in the domain [3], [4]. One disadvantage of these methods is that, to obtain satisfactory solution accuracy, it may be necessary to deal with fine meshes that significantly increase the size of the lookup table and memory required [5], [6]. Another disadvantage is that a new numerical solution must be determined from scratch, and the previous one discarded, every time a change occurs in the PDE problem.

ANNs provide an ideal representation tool for adaptive PDE solutions because they are characterized by adjustable parameters that can be modified by incremental training algorithms [7], and because of their ability to approximate nonlinear functions on a compact space. Another advantage of ANN solutions over solutions obtained by FDM or FEM is that ANN solutions are in a closed analytic form that is infinitely differentiable. Thus, an ANN solution can be represented by a small number of parameters, reducing the amount of memory required compared with FDM and FEM [8]. Furthermore, because an ANN solution is valid over the entire domain, it eliminates the need for interpolation [9].

One approach for solving PDEs numerically using ANNs is to use an FDM or FEM solution to train a neural network using a backpropagation algorithm, such as Levenberg–Marquardt (LM) [10]. Methods have also been proposed to determine the PDE solution in one step, by training an ANN to minimize an error function formulated in terms of the differential operator. One of the main difficulties that arise in ANN-based method lies in satisfying the BCs and ICs, which amounts to a set of equality constraints on a continuous domain. One possibility is to use a problem-specific ansatz that has been tailored to automatically satisfy the BCs, and includes an ANN that is trained to minimize the PDE error. Although this approach has been shown effective in solving boundary value problems (BVPs) with a high degree of accuracy [1], [11], [12], it has yet to be demonstrated on initial BVPs (IBVPs). Another disadvantage is that because the ansatz is problem specific, this approach may not be applicable to all PDE problems, and cannot be used to obtain an adaptive PDE solution.

Another approach for incorporating the I/BCs in the ANN solution is to use them for formulating a penalty function, thereby converting the constrained optimization problem into an unconstrained optimization problem [13]. As for all penalty function methods, this method can display slow convergence, and poor solution accuracy in the equality constraints. Improving accuracy typically requires using many more nodes in the ANN hidden layer, and a dense set of collocation points along the boundary of the domain. In addition to making the approach computationally expensive, these steps involve user intervention, and, therefore, do not allow for an adaptive solution of the PDE problem.

Manuscript received January 2, 2013; revised April 18, 2013; accepted July 26, 2013. Date of publication November 8, 2013; date of current version February 14, 2014. This work was supported by the National Science Foundation under ECCS under Grant 0823945.

The authors are with the Department of Mechanical Engineering and Materials Science, Duke University, Durham, NC 27707 USA (e-mail: keith.rudd@duke.edu; gianluca.dimuro@duke.edu; sferrari@duke.edu).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TNNLS.2013.2277601

2162-237X © 2013 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission.
See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.

A well-known result from constrained optimization theory is that if the equality constraints satisfy the implicit function theorem, they can be at once satisfied exactly, and used to reduce the dimensionality of the optimization problem, through the method of direct elimination [14], [15]. Thus, whenever applicable, direct elimination is to be preferred over the penalty function method or the method of Lagrange multipliers. This is because the latter relies on augmenting the objective function by a function of the constraints and, thus, increases the dimensionality of the unconstrained optimization by introducing additional variables (e.g., Lagrange multipliers).

It was recently shown in [16] that the method of direct elimination can be used to train ANNs in the presence of equality constraints through a method known as constrained backpropagation (CPROP). CPROP preserves a set of input–output and gradient information during incremental training sessions by embedding this information into a set of equality constraints that are formulated in terms of the neural weights by means of algebraic training [17]. In [18], CPROP has been used to eliminate interference and preserve prior knowledge in fully connected sigmoidal neural networks, and to adapt an ANN-based nonlinear controller online [16]. In [19], CPROP was demonstrated on benchmark problems in function approximation and system identification, and for the solution of ordinary differential equations.

This paper shows that CPROP offers a natural paradigm for solving PDEs via ANNs, because the ANN can be adapted to minimize the error defined by the differential operator, while satisfying the equality constraints provided by the I/BCs. Furthermore, because it allows for the equality constraints to be satisfied during incremental training sessions, CPROP can be used to solve PDEs adaptively. It is also shown that the adaptive CPROP solutions bring about a significant reduction in computation time compared with existing methods [20].

This paper is organized as follows. The CPROP approach is reviewed in Section II. The adaptive PDE solution problem is formulated in Section III for elliptic and parabolic IBVPs, and the novel CPROP method of solution is presented in Section IV. The computational complexity of the CPROP PDE solution algorithm is analyzed in Section V. In Section VI, the method is demonstrated through several example problems including the Laplace equation, the heat/diffusion equation, and the Boussinesq equation.

II. Background on CPROP

Classical backpropagation algorithms solve an unconstrained optimization problem involving the minimization of a scalar objective function with respect to all of the network weights. CPROP, instead, solves a constrained optimization problem involving the minimization of a scalar objective function, subject to a set of equality constraints. In particular, CPROP algorithms are based on the finding that, through algebraic training [17], the method of direct elimination can be used to train a nonlinear ANNs in the presence of equality constraints that are automatically satisfied during repeated incremental training sessions. Algebraic training consists of

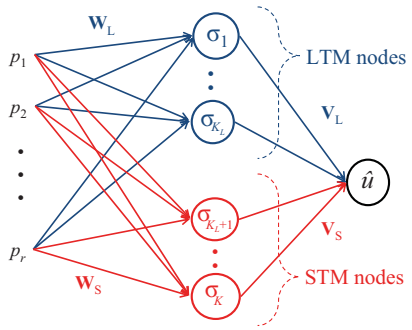


Fig. 1. Partitioning of ANN nodes and weights into LTM (L) and STM (S) sets.

deriving a set of nonlinear equations from the training data and the neural network input–output and derivative equations. Then, by letting the number of nodes be equal to the number of samples in the training set, it can be shown that an exact solution to the nonlinear algebraic training equations can be obtained by solving linear systems of equations.

As shown in Section III, CPROP can be used to approximate and adapt the solution of a PDE, $u = h(\mathbf{p})$, where $\mathbf{p} \in \mathbb{R}^{r\times 1}$, and $h : \mathbb{R}^r \to \mathbb{R}$ is a smooth scalar function of r independent variables, based on a (changing) differential operator, while satisfying a set of equality constraints obtained from the PDE I/BCs at all times. Consider a feedforward one-hidden-layer sigmoidal ANN that is characterized by universal function approximation abilities [21]–[23]. The hidden layer can be represented by an operator with repeated sigmoidal functions, $\mathbf{\Phi}(\mathbf{n}) := [\sigma(n_1)\cdots\sigma(n_K)]^T$, where n_i is the ith component of the input-to-node vector $\mathbf{n} \in \mathbb{R}^{K\times 1}$, and $\sigma(n_i) := (e^{n_i}-1)/(e^{n_i}+1)$. Then, the neural network input–output equation is as follows:

$$\hat{u} = \mathbf{\Phi}(\mathbf{p}^T\mathbf{W}^T + \mathbf{b}^T)\mathbf{V}^T \approx h(\mathbf{p}) \tag{1}$$

where $\mathbf{b} \in \mathbb{R}^{K\times 1}$, $\mathbf{W} \in \mathbb{R}^{K\times r}$, and $\mathbf{V} \in \mathbb{R}^{1\times K}$, are the adjustable bias, input, and output weights, respectively. The neural network derivative equations can be obtained by differentiating $\hat{u}$ with respect to $\mathbf{p}$.

In CPROP, an objective function to be minimized is obtained from a training set of input–output information, denoted by $\mathcal{T}_S = \{\mathbf{p}_k, u_k\}_{k=1,\ldots,N_S}$, that may be obtained in batch mode or incrementally over time, and thus is referred to as short-term memory (STM) [16]. In this paper, the STM training set is obtained from the PDE differential operator that may change as a result changing PDE parameters and/or nonhomogeneous term (forcing). Another training set, referred to as long-term memory (LTM), consists of input–output and derivative information to be preserved at all times, and is denoted by $\mathcal{T}_L = \{\mathbf{p}_\ell, h^n(\mathbf{p}_\ell)\}_{\ell=1,\ldots,N_L}$, where $h^n(\cdot)$ denotes the nth-order derivative of h with respect to $\mathbf{p}$, and $n = 1,\ldots,N_D$. The LTM is used to formulate the CPROP equality constraints via algebraic training, and, in this paper, is obtained from the PDE I/BCs.

The CPROP constraints are derived by partitioning the K nodes into an LTM set of K_L nodes, and an STM set of K_S nodes, as shown in Fig. 1, such that $K_L = N_L$ and $K_L + K_S = K$. Then, letting the subscript L denotes the weights associated with LTM-node connections, and letting

the subscript S denotes the weights associated with STM-node connections, the neural network input–output equation (1) can be partitioned as follows:

$$\hat{u}(\mathbf{p}) = \mathbf{\Phi}\left(\mathbf{p}^T\mathbf{W}_L^T + \mathbf{b}_L^T\right)\mathbf{V}_L^T + \mathbf{\Phi}\left(\mathbf{p}^T\mathbf{W}_S^T + \mathbf{b}_S^T\right)\mathbf{V}_S^T \quad (2)$$

and the LTM and STM weights can be reorganized into two vectors $\mathbf{w}_L \in \mathbb{R}^{M_L}$ and $\mathbf{w}_S \in \mathbb{R}^{M_S}$. Then, from the input–output equation (2) and its derivatives, the LTM data $\mathcal{T}_L$ can be embedded into an implicit function $\mathbf{c} : \mathbb{R}^M \to \mathbb{R}^\nu$, where $\nu = N_L + N_D$, such that the LTM data are preserved provided the equality constraint

$$\mathbf{c}(\mathbf{w}_L, \mathbf{w}_S) = \mathbf{0} \quad (3)$$

is satisfied. Furthermore, a training method that preserves $\mathcal{T}_L$ while minimizing an objective function $e : \mathbb{R}^M \to \mathbb{R}$ obtained from $\mathcal{T}_S$ can be formulated as a constrained optimization problem

$$\begin{aligned} &\min\ e(\mathbf{w}_L, \mathbf{w}_S) \\ &\text{s.t. } \mathbf{c}(\mathbf{w}_L, \mathbf{w}_S) = \mathbf{0} \end{aligned} \quad (4)$$

in the neural network weights $\mathbf{w}_L, \mathbf{w}_S$.

Now, if (3) satisfies the implicit function theorem, it uniquely implies the function

$$\mathbf{w}_L = \mathcal{C}(\mathbf{w}_S) \quad (5)$$

and the method of direct elimination can be applied by rewriting the objective function as

$$E(\mathbf{w}_S) = e(\mathcal{C}(\mathbf{w}_S), \mathbf{w}_S) \quad (6)$$

such that the value of $\mathbf{w}_S$ can be determined independently of $\mathbf{w}_L$. In this case, the solution of (4) is an extremum of (6) that obeys $\partial E/\partial \mathbf{w}_{S_{(j)}} = \mathbf{0}$ for $j = 1, \ldots, M_S$. Throughout this paper, the jth element of a vector is denoted by a subscript (j). For matrices, a single subscript (l) denotes the lth column of the matrix, and a subscript (i, j) denotes the element in the ith row and jth column of the matrix.

Once the optimal value of $\mathbf{w}_S$ is determined, the optimal value of $\mathbf{w}_L$ can be obtained from $\mathbf{w}_S$ using (5). Furthermore, the extremum of (6) can be obtained numerically using the adjoined error gradient

$$\frac{\partial E}{\partial \mathbf{w}_{S_{(i)}}} = \frac{\partial e}{\partial \mathbf{w}_{S_{(i)}}} + \frac{\partial e}{\partial \mathcal{C}}\frac{\partial \mathcal{C}}{\partial \mathbf{w}_{S_{(i)}}} \quad (7)$$

obtained from (6) using the chain rule [24]. Then, the objective function in (6) can be rewritten as follows:

$$E(\mathbf{w}_S) = \frac{1}{2}\boldsymbol{\epsilon}^T\boldsymbol{\epsilon} \quad (8)$$

where $\boldsymbol{\epsilon}_{(j)}$ is the error associated with the jth sample in $\mathcal{T}_S$.

In this paper, LM is the training algorithm of choice because of its excellent convergence and stability properties [10], [25]. In the LM algorithm, the update to the weights, $\Delta\mathbf{w}_S$, is found by solving a nonlinear system of equations

$$\left(\mathbf{J}^T\mathbf{J} + \eta\mathbf{I}\right)\Delta\mathbf{w}_S = -\mathbf{J}^T\boldsymbol{\epsilon} = -\nabla_{\mathbf{w}_S}E \quad (9)$$

where $\mathbf{I}$ is the identity matrix, η is a positive constant known as learning rate, and $\mathbf{J}$ is the Jacobian matrix. The CPROP LM training algorithm is obtained by deriving the adjoined Jacobian

$$\begin{aligned} \mathbf{J}_{(m,n)}(\mathbf{w}_S) = \frac{\partial \boldsymbol{\epsilon}_{(m)}[\mathcal{C}(\mathbf{w}_S), \mathbf{w}_S]}{\partial \mathbf{w}_{S_{(n)}}} &= \frac{\partial \boldsymbol{\epsilon}_{(m)}(\mathbf{w}_L, \mathbf{w}_S)}{\partial \mathbf{w}_{S_{(n)}}} \\ &+ \frac{\partial \boldsymbol{\epsilon}_{(m)}[\mathcal{C}(\mathbf{w}_S), \mathbf{w}_S]}{\partial \mathcal{C}}\frac{\partial \mathcal{C}}{\partial \mathbf{w}_{S_{(n)}}} \end{aligned} \quad (10)$$

and by updating the STM weights according to (9) and (10). The following sections illustrate how the CPROP LM training approach can be applied to the solution of elliptic and parabolic IBVPs, by deriving (5), the objective function (8), and the adjoined Jacobian (10).

III. CPROP Solution of Elliptic and Parabolic IBVPs

The CPROP approach offers a natural paradigm for solving PDEs via ANNs, because ANNs can be trained to minimize the error defined by the differential operator, while satisfying the I/BCs equality constraints. Satisfying I/BCs typically involves a smooth function approximation problem that can be represented by the equality constraints (3), obtained through algebraic training [17]. Therefore, CPROP can be used to guarantee that the I/BCs equality constraints are satisfied during repeated incremental sessions by introducing a sequence of objective functions e_n, for $n = 1, 2, \ldots$, such that the PDE solution is obtained and, then, adapted subject to changes in the PDE parameters or external forcing.

The CPROP PDE solution approach is illustrated for linear and nonlinear elliptic BVPs and parabolic IBVPs of second order that arise in many areas of science and engineering, and can greatly benefit from an adaptive solution method. As will be shown in a separate paper, the method is also applicable to PDEs of higher and lower orders, and to other PDE problems. A second-order PDE obeys the form

$$\begin{aligned} a(\xi, \eta, u)\frac{\partial^2 u}{\partial \xi^2} + b(\xi, \eta, u)\frac{\partial^2 u}{\partial \xi \partial \eta} + c(\xi, \eta, u)\frac{\partial^2 u}{\partial \eta^2} \\ + d(\xi, \eta, u)\frac{\partial u}{\partial \xi} + e(\xi, \eta, u)\frac{\partial u}{\partial \eta} + h(\xi, \eta, u)u = F. \end{aligned} \quad (11)$$

The above PDE is said to be elliptic if $b^2 - 4ac < 0$, parabolic if $b^2 - 4ac = 0$, and hyperbolic if $b^2 - 4ac > 0$ [26]. Elliptic and parabolic PDEs can be written in compact form

$$\mathcal{L}_n\left[u(\mathbf{p})\right] = f_n(\mathbf{p}) \quad (12)$$

where $\mathbf{p} \in \mathcal{I} \subset \mathbb{R}^r$, $\mathcal{L}_n$ is the differential operator, and $f_n : \mathbb{R}^r \to \mathbb{R}$ is a forcing function or source term.

In many science and engineering applications, the PDE differential operator and/or forcing function in (12) may be subject to change. In this case, a solution of (12) may be required for a sequence of PDEs, represented by the sequence of functions $\{(\mathcal{L}_n, f_n) : n = 1, 2, \ldots\}$, where each pair of functions $(\mathcal{L}_n, f_n)$ defines one elliptic or parabolic PDE. The I/BCs associated with the nth PDE, $(\mathcal{L}_n, f_n)$, may also be subject to change. Let every PDE problem be indexed by n, and assume that the nth PDE problem holds for a period ΔT that is much greater than the time required to obtain the ANN solution. Then, the sequence of PDE problems can be solved

incrementally by adapting the ANN solution (2) through CPROP, such that the $(n+1)$th solution is obtained after the ANN has converged to the solution of the nth PDE problem.

At every iteration of the CPROP training algorithm, the adjoined Jacobian is used to satisfy the equality constraints expressed by a training set $\mathcal{T}_L$, while minimizing an objective function defined by the differential operator. For elliptic equations, $\mathcal{T}_L$ is obtained from the BCs, and for parabolic equations, it is obtained from the ICs. The objective function (8) is obtained using a training set of points taken from the interior of the domain, denoted by $\mathcal{T}_S = \{\mathbf{p}_k | \mathbf{p}_k \in \mathcal{I} \setminus \partial\mathcal{I}\}_{k=1,\ldots,q}$. Let $\hat{u}(\mathbf{p})$ represent the PDE solution approximated by the ANN. Then, the objective function (8) can be written in terms of the differential operator error as follows:

$$\boldsymbol{\epsilon}_{(k)} \equiv \left\{ f_n(\mathbf{p}_k) - \mathcal{L}_n\left[\hat{u}(\mathbf{p}_k)\right] \right\}, \quad \mathbf{p}_k \in \mathcal{T}_S \tag{13}$$

where $\boldsymbol{\epsilon}_{(k)}$ denotes the kth element of the error $\boldsymbol{\epsilon}$ in (8).

A. Elliptic BVPs

In this section, let $\mathcal{L}_n$ in (12) denote an elliptic operator defined over a compact set $\mathcal{I} \in \mathbb{R}^r$, with BCs

$$\mathcal{B}\left[u(\mathbf{p})\right] = h(\mathbf{p}) \qquad \forall \mathbf{p} \in \partial\mathcal{I} \tag{14}$$

where $\mathcal{B}$ is a linear differential operator of order less than $\mathcal{L}_n$, and $h : \mathbb{R}^r \to \mathbb{R}$ is a continuous and known function. Satisfying the BCs in (14) requires solving a smooth function approximation problem that can be formulated as an equality constraint (3), using an LTM training set in the form

$$\mathcal{T}_L = \{\mathbf{p}_\ell, h(\mathbf{p}_\ell)\}_{\ell=1,\ldots,N_L}. \tag{15}$$

Because we seek an approximate ANN solution to the elliptic BVP problem (12), (14), the number of LTM nodes is $K_L = N_L$, based on the algebraic training approach in [17]. The number of STM nodes, K_S, is determined heuristically, based on the size and complexity of $\mathcal{T}_S$.

The CPROP objective function (4) is obtained by applying the differential operator $\mathcal{L}_n$ to the approximate ANN solution (2), and by substituting the result in (13) as follows. Consider the partial derivative

$$\chi(\mathbf{p}) = \frac{\partial^\gamma u(\mathbf{p})}{\partial \mathbf{p}_{(1)}^{m_1} \ldots \partial \mathbf{p}_{(r)}^{m_r}} \tag{16}$$

where $\gamma = m_1 + \cdots + m_r$. Let $\boldsymbol{\omega}_{S_j}$ represents a diagonal matrix of the jth column of $\mathbf{W}_S$, and let

$$\boldsymbol{\Lambda}_S = \prod_{j=1}^{r} \boldsymbol{\omega}_{S_j}^{m_j}. \tag{17}$$

Similarly, $\boldsymbol{\Lambda}_L$ is a product of diagonal matrices taken from columns of $\mathbf{W}_S$. Then, the ANN input–output equation (2) is differentiated with respect to $\mathbf{p}$

$$\begin{aligned} \frac{\partial^\gamma \hat{u}(\mathbf{p})}{\partial \mathbf{p}_{(1)}^{m_1} \ldots \partial \mathbf{p}_{(r)}^{m_r}} \equiv \hat{\chi}(\mathbf{p}) &= \boldsymbol{\Phi}^\gamma \left(\mathbf{p}^T \mathbf{W}_L^T + \mathbf{b}_L^T\right) \boldsymbol{\Lambda}_L \mathbf{V}_L^T \\ &+ \boldsymbol{\Phi}^\gamma \left(\mathbf{p}^T \mathbf{W}_S^T + \mathbf{b}_S^T\right) \boldsymbol{\Lambda}_S \mathbf{V}_S^T \end{aligned} \tag{18}$$

where $\boldsymbol{\Phi}^\gamma(\cdot)$ denotes the γth derivative of the sigmoidal operator. Thus, all the partial derivatives in $\mathcal{L}_n[\hat{u}(\mathbf{p})]$ can be obtained in the closed form from (2)–(17), and substituted in (13) to determine the ANN objective function (8).

B. Parabolic IBVPs

In this section, let $\mathcal{L}_n$ in (12) denote a parabolic operator, and let the parabolic PDE solution be a function of $\mathbf{p} \in \mathcal{I}$, where $\mathcal{I} = \mathcal{H} \times [t_0, t_f) \subset \mathbb{R}^r$, and $\mathcal{H}$ is a compact set. We consider the case of Dirichlet BCs

$$u(\mathbf{p}) = h(\mathbf{p}) \qquad \forall \mathbf{p} \in \partial\mathcal{H} \times [t_0, t_f) \tag{19}$$

where $h : \mathbb{R}^r \to \mathbb{R}$. IBVPs differ from BVPs in that they also have an IC associated with one of the variables in $\mathbf{p}$, denoted here by $\mathbf{p}_{(r)}$, where $t_0 \le \mathbf{p}_{(r)} \le t_f$. Then, in addition to (19), the PDE solution must also satisfy the IC

$$u(\mathbf{p}_{(1)}, \ldots, \mathbf{p}_{(r-1)}, t_0) = d(\mathbf{p}_{(1)}, \ldots, \mathbf{p}_{(r-1)}) \tag{20}$$

where $d : \mathbb{R}^{r-1} \to \mathbb{R}$ is a known function.

From the ANN input–output equation (2), the approximate ANN solution can be written as

$$\begin{aligned} \hat{u}(\mathbf{p}) = \tilde{h}(\mathbf{p}) + q(\mathbf{p}) \Big[&\boldsymbol{\Phi}\left(\mathbf{p}^T \mathbf{W}_L^T + \mathbf{b}_L^T\right) \mathbf{V}_L^T \\ &+ \boldsymbol{\Phi}\left(\mathbf{p}^T \mathbf{W}_S^T + \mathbf{b}_S^T\right) \mathbf{V}_S^T \Big] \end{aligned} \tag{21}$$

where $\tilde{h} : \mathbb{R}^r \to \mathbb{R}$ is a differentiable function, that satisfies the BCs in (19) for all $\mathbf{p}$ in $\mathcal{I}$. The function $q : \mathbb{R}^r \to \mathbb{R}$ also is differentiable, and it is zero for $\mathbf{p} \in \partial\mathcal{H}$, and nonzero for $\mathbf{p} \in \{\mathcal{H} \setminus \partial\mathcal{H}\}$. When the BCs do not change over time, using (21) has the advantage that the BCs are automatically satisfied, leaving the IC as the only equality constraint.

With the above simplification, the ANN equality constraint (3) is obtained from the training set

$$\begin{aligned} \mathcal{T}_L &= \{\mathbf{p}_\ell, z(\mathbf{p}_\ell)\}_{\ell=1,\ldots,N_L} \\ &= \left\{ [\mathbf{p}_{(1)}, \ldots, \mathbf{p}_{(r-1)}, t_0]_\ell^T, \left. \frac{u(\mathbf{p}) - \tilde{h}(\mathbf{p})}{q(\mathbf{p})} \right|_{\mathbf{p}=\mathbf{p}_\ell} \right\}_{\ell=1,\ldots,N_L} \end{aligned}$$

for $\mathbf{p}_1, \ldots, \mathbf{p}_{r-1} \in \mathcal{H}$. The ANN objective function (8) to be minimized is obtained from the parabolic differential operator $\mathcal{L}_n$, as shown in Section III-A. For the parabolic IBVP problem described in this subsection, the partial derivatives, however, differ from (18) because $\mathcal{L}_n$ is applied to the ANN approximate solution in (21). The derivatives of (21) consist of products of $q(\mathbf{p})$ and its derivatives, and the derivatives of the ANN output, $\hat{\chi}(\mathbf{p})$, shown in (18). In particular, the first-order derivatives of (21) are given by

$$\begin{aligned} \frac{\partial \hat{u}(\mathbf{p})}{\partial \mathbf{p}_{(j)}} = &\frac{\partial \tilde{h}(\mathbf{p})}{\partial \mathbf{p}_{(j)}} + \frac{\partial q(\mathbf{p})}{\partial \mathbf{p}_{(j)}} \\ &\times \left[\boldsymbol{\Phi}\left(\mathbf{p}^T \mathbf{W}_L^T + \mathbf{b}_L^T\right) \mathbf{V}_L^T + \boldsymbol{\Phi}\left(\mathbf{p}^T \mathbf{W}_S^T + \mathbf{b}_S^T\right) \mathbf{V}_S^T \right] \\ &+ q(\mathbf{p}) \Big[\boldsymbol{\Phi}^1\left(\mathbf{p}^T \mathbf{W}_L^T + \mathbf{b}_L^T\right) \boldsymbol{\omega}_{L_j} \mathbf{V}_L^T \\ &\quad + \boldsymbol{\Phi}^1\left(\mathbf{p}^T \mathbf{W}_S^T + \mathbf{b}_S^T\right) \boldsymbol{\omega}_{S_j} \mathbf{V}_S^T \Big] \end{aligned} \tag{22}$$

and the second-order derivatives of (21) are given by (62), in Appendix VII.

IV. Derivation of Adjoined Jacobian for Elliptic and Parabolic PDEs

This section derives the CPROP adjoined Jacobian (10) from the PDE objective functions derived in the previous section. As shown in Section II, the adjoined Jacobian accounts for the ANN equality constraints (5) and, thus, can be used to minimize the constrained objective function numerically. Although the adjoined Jacobian depends on the form of the differential operator $\mathcal{L}_n$, for all elliptic and parabolic PDEs it can be expressed in terms of the partial derivative $\hat{\chi}$, defined in (16), as shown in this section. Then, because the constraint (5) and the Jacobian (10) are functions of the PDE parameters and external forcing, the ANN solution can be adapted incrementally over time.

Because the forcing function $f_n(\cdot)$ in (13) is independent of $\mathbf{w}_S$, it follows that $\partial \boldsymbol{\epsilon}_{(k)}/\partial \mathbf{w}_S = -\partial \mathcal{L}_n/\partial \mathbf{w}_S|_{\mathbf{p}_k}$ for any k, where $k = 1, \ldots, q$. Let M denotes the number of partial derivatives of $\hat{u}$ in $\mathcal{L}_n$. Then, the equality constraint $\mathcal{L}_n = F(\hat{\chi}_1, \ldots, \hat{\chi}_M)$, the adjoined error gradient (7), and the Jacobian (10) can be obtained from the gradient

$$\frac{\partial \mathcal{L}_n[\hat{u}(\mathbf{p})]}{\partial \mathbf{w}_{S_{(k)}}} = \sum_i \frac{\partial F}{\partial \hat{\chi}_i} \frac{\partial \hat{\chi}_i}{\partial \mathbf{w}_{S_{(k)}}}. \tag{23}$$

For every ith derivative, let

$$\frac{\partial F}{\partial \hat{\chi}_i} \frac{\partial \hat{\chi}_i}{\partial \mathbf{w}_{S_{(k)}}} = \frac{\partial F}{\partial \hat{\chi}_i} \left[g_1(\mathbf{p}) + g_2(\mathbf{p}) \right] \tag{24}$$

where

$$\begin{aligned} g_1(\mathbf{p}) &= \frac{\partial}{\partial \mathbf{w}_{S_{(k)}}} \left[\boldsymbol{\Phi}^{\gamma} (\mathbf{p}^T \mathbf{W}_L^T + \mathbf{b}_L^T) \boldsymbol{\Lambda}_L \mathbf{V}_L^T \right] \\ g_2(\mathbf{p}) &= \frac{\partial}{\partial \mathbf{w}_{S_{(k)}}} \left[\boldsymbol{\Phi}^{\gamma} (\mathbf{p}^T \mathbf{W}_S^T + \mathbf{b}_S^T) \boldsymbol{\Lambda}_S \mathbf{V}_S^T \right] \end{aligned} \tag{25}$$

and $\partial[\cdot]/\partial \mathbf{w}_{S_{(k)}}$ denotes the kth element of the gradient vector $\partial[\cdot]/\partial \mathbf{w}_S$. Because $\mathbf{w}_S$ is obtained by regrouping the elements of $\mathbf{W}_S$, $\mathbf{b}_S$, and $\mathbf{V}_S$, the partial derivatives with respect to these weights are derived separately as follows.

As a first step, consider the input weights, where $\mathbf{w}_{S_{(k)}}$ corresponds to the input weight $\mathbf{W}_{S_{(i,j)}}$, and let

$$\alpha_{ij} \equiv m_j \left(\mathbf{W}_{S_{(i,j)}} \right)^{m_j - 1} \prod_{k \neq j}^{r} \left(\mathbf{W}_{S_{(i,k)}} \right)^{m_k}. \tag{26}$$

Then, for any input weight $\mathbf{W}_{S_{(i,j)}}$, the term $g_2(\mathbf{p})$ in (24) can be written as

$$\begin{aligned} g_2(\mathbf{p}) &= \frac{\partial}{\partial \mathbf{W}_{S_{(i,j)}}} \left[\boldsymbol{\Phi}^{\gamma} (\mathbf{p}^T \mathbf{W}_S^T + \mathbf{b}_S^T) \boldsymbol{\Lambda}_S \mathbf{V}_S^T \right] \\ &= \Big[\alpha_{ij} \boldsymbol{\Phi}_{(i)}^{\gamma} (\mathbf{p}^T \mathbf{W}_S^T + \mathbf{b}_S^T) \\ &\quad + \boldsymbol{\Lambda}_{S_{(i,i)}} \boldsymbol{\Phi}_{(i)}^{\gamma+1} (\mathbf{p}^T \mathbf{W}_S^T + \mathbf{b}_S^T) \mathbf{p}_{(j)} \Big] \mathbf{V}_{S_{(i)}} \end{aligned} \tag{27}$$

and for any input bias $\mathbf{b}_{S_{(i)}}$, or output weight $\mathbf{V}_{S_{(i)}}$, $g_2(\mathbf{p})$ can be written as

$$\begin{aligned} g_2(\mathbf{p}) &= \frac{\partial}{\partial \mathbf{b}_{S_{(i)}}} \left[\boldsymbol{\Phi}^{\gamma} (\mathbf{p}^T \mathbf{W}_S^T + \mathbf{b}_S^T) \boldsymbol{\Lambda}_S \mathbf{V}_S^T \right] \\ &= \boldsymbol{\Lambda}_{S_{(i,i)}} \boldsymbol{\Phi}_{(i)}^{\gamma+1} (\mathbf{p}^T \mathbf{W}_S^T + \mathbf{b}_S^T) \mathbf{V}_{S_{(i)}} \end{aligned} \tag{28}$$

or

$$\begin{aligned} g_2(\mathbf{p}) &= \frac{\partial}{\partial \mathbf{V}_{S_{(i)}}} \left[\boldsymbol{\Phi}^{\gamma} (\mathbf{p}^T \mathbf{W}_S^T + \mathbf{b}_S^T) \boldsymbol{\Lambda}_S \mathbf{V}_S^T \right] \\ &= \boldsymbol{\Lambda}_{S_{(i,i)}} \boldsymbol{\Phi}_{(i)}^{\gamma} (\mathbf{p}^T \mathbf{W}_S^T + \mathbf{b}_S^T) \end{aligned} \tag{29}$$

respectively. These equations provide the first term of the adjoined Jacobian (10), and correspond to the (unconstrained) partial derivatives used in classical backpropagation. The second term in the adjoined Jacobian (10) is given by $(\partial F/\partial \hat{\chi}_i) g_1(\mathbf{p})$ in (24). This term is referred to as constrained derivative because it takes into account the ANN constraint (5), and is derived in the following sections.

A. Constrained Derivative in Elliptic BVPs

The equality constraint for an elliptic PDE is obtained from a training set $\mathcal{T}_L$ determined from the BCs. When the BCs in (14) are imposed on the ANN approximate solution (2), they can be written as

$$\begin{aligned} \mathcal{B}[\hat{u}(\mathbf{p})] = \mathcal{B} \left[\boldsymbol{\Phi} \left(\mathbf{p}^T \mathbf{W}_L^T + \mathbf{b}_L^T \right) \right] \mathbf{V}_L^T \\ + \mathcal{B} \left[\boldsymbol{\Phi} \left(\mathbf{p}^T \mathbf{W}_S^T + \mathbf{b}_S^T \right) \right] \mathbf{V}_S^T \end{aligned} \tag{30}$$

for a linear operator $\mathcal{B}$. According to the algebraic training approach in [17], (30) is evaluated at the collocation points in the training set $\mathcal{T}_L$ and arranged into a linear system of equations. It follows that an ANN that satisfies $\mathcal{T}_L$ at all times can be obtained provided training satisfies the following equality constraint:

$$\mathbf{h} = \boldsymbol{\Psi} \mathbf{V}_L^T + \boldsymbol{\Omega} \mathbf{V}_S^T \tag{31}$$

where

$$\mathbf{h}_{(\ell)} \equiv h(\mathbf{p}_\ell) \tag{32}$$

$$\boldsymbol{\Psi}_{(\ell,k)} \equiv \mathcal{B} \left[\boldsymbol{\Phi}_{(k)} \left(\mathbf{p}_\ell^T \mathbf{W}_L^T + \mathbf{b}_L^T \right) \right] \tag{33}$$

$$\boldsymbol{\Omega}_{(\ell,k)} \equiv \mathcal{B} \left[\boldsymbol{\Phi}_{(k)} \left(\mathbf{p}_\ell^T \mathbf{W}_S^T + \mathbf{b}_S^T \right) \right] \tag{34}$$

for all $\mathbf{p}_\ell \in \mathcal{T}_L$. Then, an explicit equality constraint in the form (5) can be obtained from (31) as follows:

$$\mathbf{V}_L^T = \boldsymbol{\Psi}^{-1} [\mathbf{h} - \boldsymbol{\Omega} \mathbf{V}_S^T] \tag{35}$$

where $\boldsymbol{\Psi}$ is assumed to be an invertible matrix that can be constructed using the method in [17].

According to the CPROP training approach reviewed in Section II, the objective function (8) is minimized with respect to $\mathbf{w}_S$, while $\mathbf{W}_L$ is held constant. Thus, the matrix $\boldsymbol{\Psi}$ remains known and constant at all times. With the constraint now defined, the function $g_1(\mathbf{p})$ in (24) is

$$g_1(\mathbf{p}) = \boldsymbol{\Phi}^{\gamma} \left(\mathbf{p}^T \mathbf{W}_L^T + \mathbf{b}_L^T \right) \boldsymbol{\Lambda}_L \frac{\partial \mathbf{V}_L^T}{\partial \mathbf{w}_{S_{(k)}}}. \tag{36}$$

Then, the derivative of the explicit constraint equation (35) with respect to any STM input weight $\mathbf{w}_{S_{(k)}} = \mathbf{W}_{S_{(i,j)}}$ is given by

$$\frac{\partial \mathbf{V}_L^T}{\partial \mathbf{W}_{S_{(i,j)}}} = -\boldsymbol{\Psi}^{-1} \mathbf{y} \mathbf{V}_{S_{(i)}}. \tag{37}$$

For any point $\mathbf{p}_\ell \in \partial\mathcal{I}$ at which $\mathcal{B}$ defines the Dirichlet BCs, the ℓth element of the vector $\mathbf{y}$ in (37) is given by

$$\mathbf{y}_{(\ell)} = \mathbf{p}_{\ell_{(j)}} \mathbf{\Phi}'_{(i)} \left(\mathbf{p}_\ell^T \mathbf{W}_S^T + \mathbf{b}_S^T \right). \tag{38}$$

While for any point $\mathbf{p}_\ell \in \partial\mathcal{I}$ at which $\mathcal{B}$ defines BCs on the derivatives of u

$$\mathbf{y}_{(\ell)} = \alpha_{ij} \mathbf{\Phi}^\gamma_{(i)} \left(\mathbf{p}_\ell^T \mathbf{W}_S^T + \mathbf{b}_S^T \right) + \mathbf{\Lambda}_{S_{(i,i)}} \mathbf{p}_{\ell_{(j)}} \mathbf{\Phi}^{\gamma+1}_{(i)} \left(\mathbf{p}_\ell^T \mathbf{W}_S^T + \mathbf{b}_S^T \right). \tag{39}$$

Similarly, the derivative of the constraint (35) with respect to the input bias is given by

$$\frac{\partial \mathbf{V}_L^T}{\partial \mathbf{b}_{S_{(i)}}} = -\mathbf{\Psi}^{-1} \mathbf{y} \mathbf{V}_{S_{(i)}} \tag{40}$$

where for any point $\mathbf{p}_\ell \in \partial\mathcal{I}$ at which $\mathcal{B}$ defines Dirichlet BCs

$$\mathbf{y}_{(\ell)} = \mathbf{\Phi}'_{(i)} \left(\mathbf{p}_\ell^T \mathbf{W}_S^T + \mathbf{b}_S^T \right) \tag{41}$$

and for any point $\mathbf{p}_\ell \in \partial\mathcal{I}$ at which $\mathcal{B}$ defines BCs on the derivatives of u

$$\mathbf{y}_{(\ell)} = \mathbf{\Lambda}_{S_{(i,i)}} \mathbf{\Phi}^{\gamma+1}_{(i)} \left(\mathbf{p}_\ell^T \mathbf{W}_S^T + \mathbf{b}_S^T \right). \tag{42}$$

Finally, the derivative of (35) with respect to the output weights is

$$\frac{\partial \mathbf{V}_L^T}{\partial \mathbf{V}_{S_{(i)}}} = -\mathbf{\Psi}^{-1} \mathbf{y} \tag{43}$$

where

$$\mathbf{y}_{(\ell)} = \mathbf{\Phi}_{(i)} \left(\mathbf{p}_\ell^T \mathbf{W}_S^T + \mathbf{b}_S^T \right) \tag{44}$$

for points with Dirichlet BCs, and

$$\mathbf{y}_{(\ell)} = \mathbf{\Lambda}_{S_{(i,i)}} \mathbf{\Phi}^\gamma_{(i)} \left(\mathbf{p}_\ell^T \mathbf{W}_S^T + \mathbf{b}_S^T \right). \tag{45}$$

for points with BCs on the derivatives of u.

Equations (27)–(29) and (37)–(43) complete the derivation of the equality constraints and corresponding adjoined Jacobian for elliptic BVPs. The following section derives the equality constraints and adjoined Jacobian for parabolic IBVPs.

B. Constrained Derivative in Parabolic IBVPs

The equality constraint for a parabolic PDE is obtained by evaluating the ANN input–output equation in (21) at all points in the training set $\mathcal{T}_L$, defined in (22). The resulting set of algebraic equations is then organized into a linear system of equations that can be solved to obtain the explicit equality constraint

$$\mathbf{V}_L^T = \mathbf{\Psi}^{-1} \left(\mathbf{z} - \mathbf{\Omega} \mathbf{V}_S^T \right) \tag{46}$$

where $\mathbf{\Psi}$ and $\mathbf{\Omega}$ are defined as in Section IV-A, $\mathbf{z}_{(j)} \equiv z(\mathbf{p}_j)$, and $z(\cdot)$ is defined in (22). It can be seen that if the problem is shifted so that $t_0 = 0$, the term $\mathbf{p}_k^T \mathbf{W}_L^T$ in (33) is independent of the weights in the rth column of $\mathbf{W}_L$, and, thus, so is the equality constraint (46) representing the PDE ICs. This equality constraint is also independent of the rth column of $\mathbf{W}_S$, and, thus, the corresponding derivatives needed to train these weights can be computed by means of classical backpropagation.

TABLE I
COMPUTATIONAL COMPLEXITY OF ANN PDE SOLUTION METHODS

	CPROP	FDM w/ ANN	Penalty Function
$\mathbf{J}$	$O(N_S N_L K_S K_L)/O(N_S N_L K_S)$	$O(NK)$	$O(NK)$
$\mathbf{J}^T\mathbf{J}$	$O(N_S K_S^2)$	$O(NK^2)$	$O(NK^2)$
LM update	$O(K_S^3)$	$O(K^3)$	$O(K^3)$

Because the equality constraint (46) is in the same form as the elliptic constraint (35), the derivatives in (37)–(43) are also used to compute the adjoined Jacobian for the parabolic IBVP. Given the explicit equality constraint equations, the adjoined Jacobian, and objective function derived in Sections III and IV, the CPROP LM algorithm described in Section II is used to determine the ANN weights incrementally, such that the chosen PDE is solved within a user-defined tolerance e_{tol}.

V. COMPUTATIONAL COMPLEXITY ANALYSIS

This section analyzes how the computational time required by CPROP grows with respect to the number of ANN weights, which is proportional to the number of nodes in the hidden layer, and with respect to the number of collocation points. Thus, in this section, the order of the computation required by the adjoined Jacobian is derived for the case of linear elliptic/parabolic PDEs, and a comparison is made between the order of operations of the constrained and unconstrained training algorithms, as shown in Table I.

Consider the computation of the derivatives of the differential operator in (24), required to obtain the adjoined gradient for elliptic BVPs. Let N_S represents the number of training pairs in $\mathcal{T}_S$, and N_L represents the number of training pairs in $\mathcal{T}_L$, and let $N = N_L + N_S$. The term $\mathbf{\Phi}^\gamma(\mathbf{p}^T \mathbf{W}_L^T + \mathbf{b}_L^T)$ is a row vector of length K_L that is independent of $\mathbf{w}_S$. The derivative of $\mathbf{V}_L^T$ is given by (37) and (43), which multiply the matrix $\mathbf{\Psi}^{-1} \in \mathbb{R}^{K_L \times N_L}$ by an $N_L \times 1$ vector. Then, the most computationally expensive operation in (24) is a matrix–vector multiplication that is $O(K_L N_L)$. Performing this multiplication for N_S collocation points and K_S weights leads to a computational complexity $O(K_L N_L K_S N_S)$ for the derivatives in (24). In the case of a parabolic IBVP, the complexity of (24) can be reduced compared with the elliptic case because $\mathbf{W}_L$ is held constant, and thus $\mathbf{\Phi}^\gamma(\mathbf{p}^T \mathbf{W}_L^T + \mathbf{b}_L^T)\mathbf{\Psi}^{-1}$ can be computed for all N_S points and stored prior to training. Then, the matrix–vector multiplication in (24) is reduced to a vector–vector multiplication with a total number of operations $O(N_L N_S K_S)$.

The function $g_2(\mathbf{p})$ in (24) does not contain the constraint, and thus is of the same order as unconstrained LM, and can be obtained from (27)–(29). The most computationally expensive operation in (27)–(29) is evaluating $\mathbf{\Phi}^\gamma$ at $N_S \times K_S$ points, an operation that requires $O(N_S K_S)$ computations. When the differential operator is linear, $\partial \mathcal{L}_n[\hat{u}(\mathbf{p})]/\partial \hat{\chi}$ in (24) is a constant. Thus, computing the Jacobian is $O(N_L N_S K_S)$ for elliptic problems, and $O(N_L N_S K_L K_S)$ for parabolic problems.

One approach that has been used extensively in the literature is to use a numerical solution method, such as FDM, to obtain a discrete solution in the form of a lookup table, and then to use this solution to train an ANN [27], [28].

Explicit FDM schemes are known to be $O(N)$, although, in practice, more computations may be required to obtain an accurate solution and avoid instabilities. Hence, the most computationally expensive step in training an ANN via FDM is determining the (unconstrained) Jacobian, which requires evaluating $\mathbf{\Phi}$ at NK points and, thus, is $O(NK)$. It can be easily shown that this is also the order of the computation required by the Jacobian in penalty function methods, which include all the collocation points in $\mathcal{T}_L$.

Suppose the LM algorithm is used to train an ANN, either using classical unconstrained backpropagation or CPROP. LM requires computing $\mathbf{J}^T\mathbf{J}$, and solving the linear system of equations in (9) to update the weights (LM update). In the case of CPROP, computing $\mathbf{J}^T\mathbf{J}$ is $O(K_S^2 N_S)$, whereas in the case of unconstrained ANN training (using the FDM solution or penalty function method), computing $\mathbf{J}^T\mathbf{J}$ is $O(NS^2)$. The order of the computations required to solve the system of equations in (9) for CPROP is $O(K_S^3)$, whereas for the FDM solution and the penalty function method, it is $O(K^3)$. As can be expected, the most expensive step in the CPROP method is computing the Jacobian, with complexity $O(N_S N_L K_S)$ for the elliptic case, whereas the most expensive step in the other methods is $O(NK^2)$. Because, typically, $N > K$ to avoid over-fitting, it can be concluded that the computational complexity of CPROP is comparable both with the process of training an ANN using an FDM solution, and to the method of solving the PDE via ANNs using penalty function methods.

VI. Numerical Simulations and Results

This section demonstrates the effectiveness of the CPROP methodology through several examples of elliptic and parabolic PDEs. Unlike other methods of solution, such as the MATLAB PDE toolbox [29] used here for comparison, the CPROP methodology does not pose any restrictions on the class of the PDE, or on the form of the I/BCs and their domain. In each example, the CPROP solution is compared either with the analytical solution, or the best available numerical solution when an analytical solution is not known.

A. Adaptive CPROP Solution of Elliptic BVP

Consider the elliptic equation

$$\nabla^2 u(\mathbf{p}) + \alpha_n e^{u(\mathbf{p})} = \alpha_n \left[1 + \mathbf{p}_{(1)}^2 + \mathbf{p}_{(2)}^2 + \frac{4}{(1+\mathbf{p}_{(1)}^2+\mathbf{p}_{(2)}^2)^2} \right] \quad \mathbf{p} \in \mathcal{I} = [-1,1]\times[-1,1] \tag{47}$$

with the BC

$$u(\mathbf{p}) = \log(\mathbf{p}_{(1)}^2 + \mathbf{p}_{(2)}^2 + 1) \qquad \forall \mathbf{p} \in \partial\mathcal{I}. \tag{48}$$

The above PDE can be used to capture many dynamic processes in fluid mechanics, electrostatics, and thermodynamics, such as steady incompressible irrotational fluid flow in two dimensions, and heat/diffusion processes in steady state. The adaptive CPROP solution approach is illustrated by changing the parameter α_n, representing the relative importance of the nonlinear term versus the forcing function. A sequence of six PDEs problems in (47) and (48) is obtained by letting $n = 0, \ldots, 5$, and $\alpha_n = 0.2n$. For $\alpha_n = 0$, the PDE in (47) reduces to Laplace's equation.

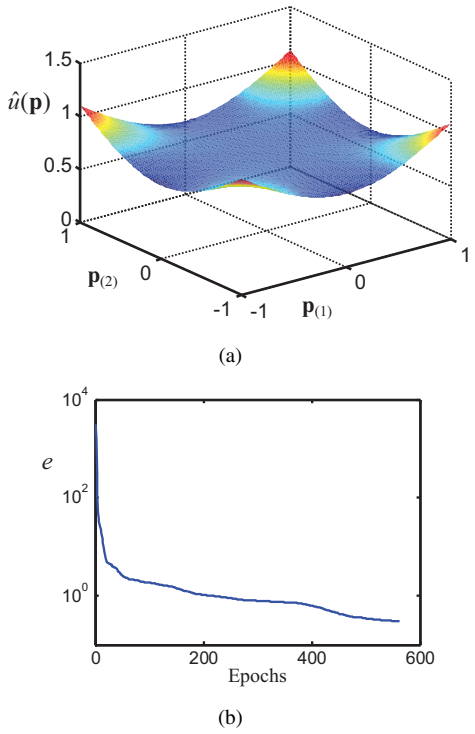
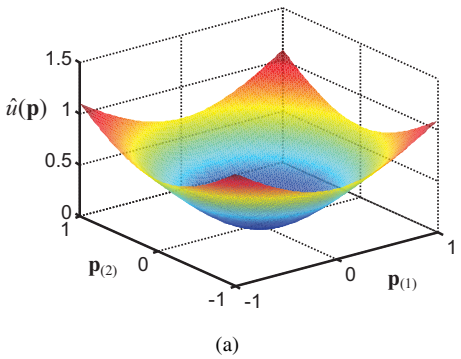


Fig. 2. (a) CPROP solution of the elliptic PDE (47) when $n = 0$, and (b) corresponding training blue.

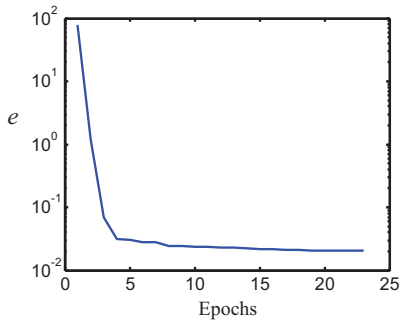
Using the CPROP methodology presented in Section IV, the ANN in (2) is trained to solve these six PDEs adaptively. When the objective function decreases below e_{tol}, the CPROP algorithm ceases training the ANN. Then, when the value of α_n is modified, the change is reflected in the training sets and, subsequently, in the objective function. As a result, the objective function exceeds e_{tol}, and the CPROP algorithm resumes training the ANN incrementally, starting with the weights obtained during the last training session.

The input data in $\mathcal{T}_L$ consist of 180 equally spaced collocation points in $\partial\mathcal{I}$. The input data in $\mathcal{T}_S$ consist of a 35×35 grid of points in the interior of $\mathcal{I}$. The corresponding output data for the two training sets are computed as explained in Section IV. The ANN is partitioned into 40 LTM nodes and 20 STM nodes (defined in Fig. 1). The training set $\mathcal{T}_S$ is used to formulate the objective function (13) in terms of the differential operator in (47). At $n = 0$, the weights are initialized randomly. No training of $\mathbf{W}_L$ and $\mathbf{b}_L$ is required and, instead, it is sufficient to initialize the input weights with uniformly distributed values in the interval $(-5, 5)$, similarly to [30] and [31]. The CPROP adaptive solution is obtained for $n = 0, \ldots, 5$, and is plotted in Figs. 2 and 3 for $n = 0$ and $n = 5$, respectively, along with the objective function e, defined in (8).

For $n = 0$, the CPROP solution is evaluated using the MATLAB PDE toolbox solution [29], and for $n = 5$, it is evaluated using the analytical solution, $u(\mathbf{p}) = \log(\mathbf{p}_{(1)}^2 + \mathbf{p}_{(2)}^2 + 1)$ [12]. For $n = 5$, a quantitative comparison between the CPROP ANN solution, $\hat{u}$, and the analytical solution, u,



(a)



(b)

Fig. 3. (a) CPROP solution of the elliptic PDE (47) when $n = 5$, and (b) corresponding training blue.

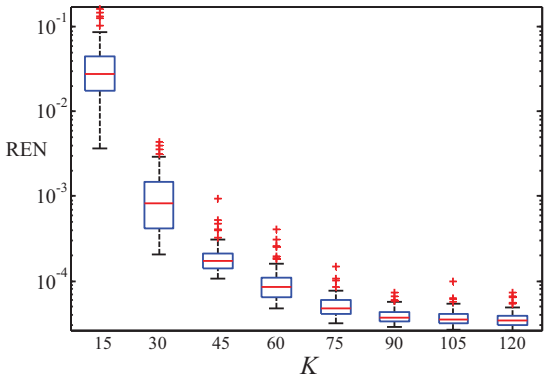


Fig. 4. Box plot of REN between the CPROP and the analytical solution of the elliptic PDE (47) when $n = 5$, for different ANN sizes (K).

is obtained by computing the relative error norm (REN)

$$\text{REN} = \frac{\sum_i [u(\mathbf{p}_i) - \hat{u}(\mathbf{p}_i)]^2}{\sum_i u^2(\mathbf{p}_i)} \tag{49}$$

using a validation set of points in the domain that was not used for training. Because the REN varies with network size and with different random initializations of the weights, a box plot of REN is obtained using eight network sizes (K) shown in Fig. 4. It is found that the CPROP solution achieves the same order of accuracy as the MATLAB finite element solution. Each box plot in Fig. 4 shows the distribution of REN resulting from 100 CPROP solutions with random initializations, for the chosen numbers of nodes. Similar results were obtained by comparing the CPROP solution and the MATLAB solution for $n = 0$, but are omitted for brevity.

For $n = 1, \ldots, 4$, analytical solutions are not available, and the MATLAB PDE toolbox is not applicable because

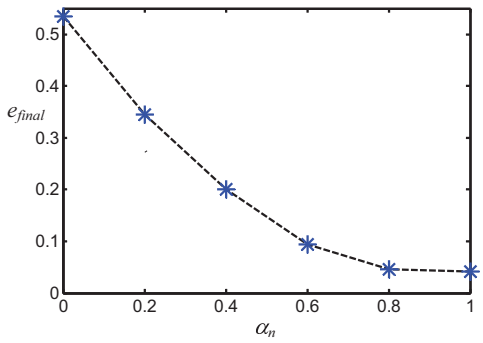


Fig. 5. Final training error of elliptic PDEs CPROP solutions.

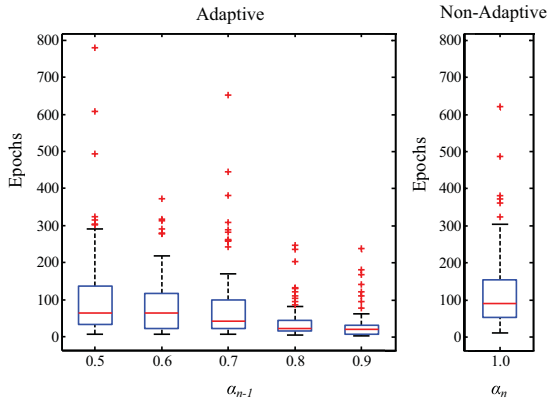


Fig. 6. Box plot of CPROP training epochs needed to solve the elliptic PDE (47) adaptively from the α_{n-1} solution, and nonadaptively.

of the presence of the nonlinearity in the elliptic PDE (47). Therefore, the adaptive CPROP solution is evaluated by monitoring the objective function e, defined in terms of the differential operator. The simulation results in Fig. 5 show that the value of e at the end of each training session, denoted by e_{final}, decreases exponentially with n. Because the order of magnitude of the PDE solution does not vary with n, this result shows that the REN also decreases exponentially with n and, thus, the accuracy of the CPROP solution improves from one PDE problem to the next. This is because the adaptive CPROP approach benefits from its knowledge of the previous solution through incremental training. Furthermore, as shown in Fig. 6, with every new PDE problem, fewer iterations are required to converge a satisfactory CPROP solution (with $e < e_{\text{tol}}$). The results in Fig. 6 were obtained by solving the elliptic PDE in (47) adaptively 100 times using CPROP. In every case, the PDE problem was also solved using CPROP with random initial weights (nonadaptively). As shown in Fig. 6, the number of epochs required by the adaptive CPROP solution is far less than that required by the nonadaptive CPROP solution.

B. Adaptive CPROP Solution of Parabolic IBVPs

This section presents the results obtained for a 2-D linear unsteady heat/diffusion equation without convection or source/sink terms, which is one of the most basic parabolic equations. The same equation is then solved in 3-D space. Both types of PDE problems are solved adaptively, subject to a changing coefficient that represents the diffusivity of the material. Then, the CPROP methodology is demonstrated on

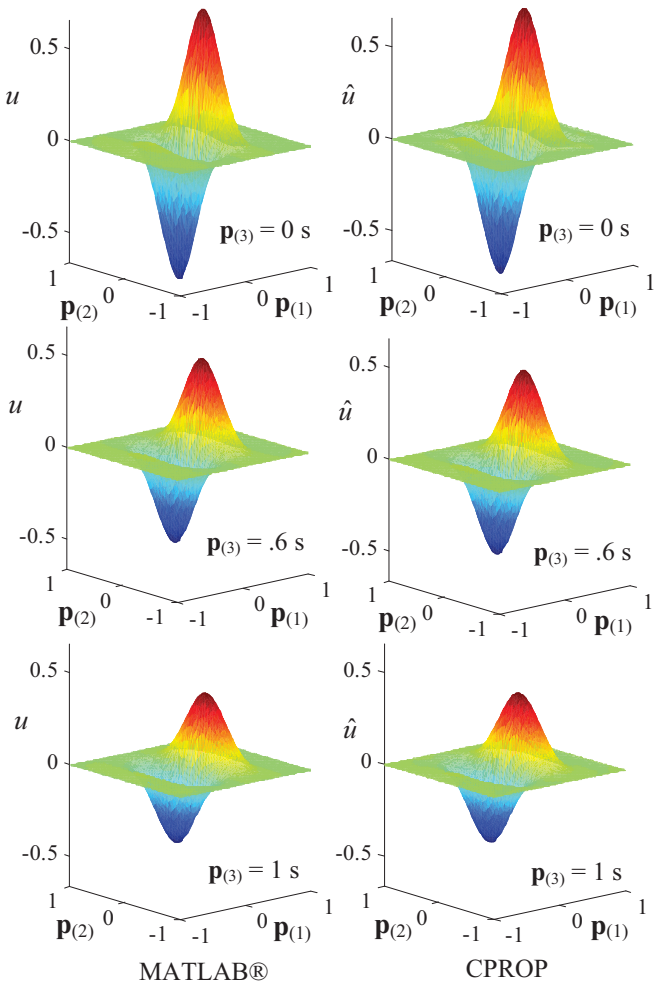


Fig. 7. Solutions of 2-D heat/diffusion equation obtained using MATLAB and CPROP for $n = 0$, $\mathbf{p}_{(3)} = 0$ s, $\mathbf{p}_{(3)} = 0.6$ s, and $\mathbf{p}_{(3)} = 1$ s.

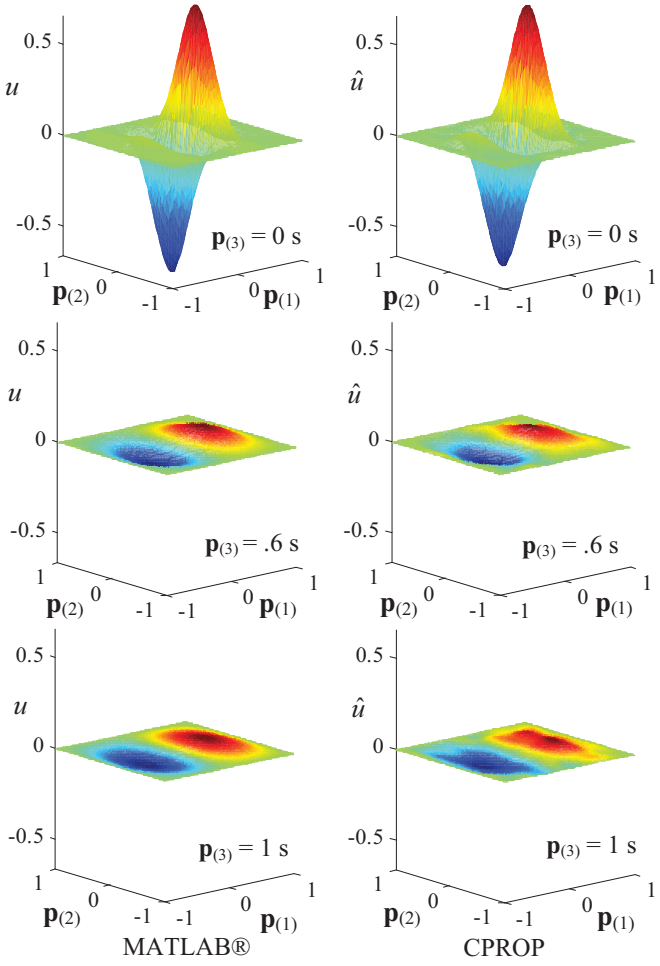


Fig. 8. Adaptive CPROP solution of the 2-D heat/diffusion (50) for $n = 1$ is compared with the (nonadaptive) solution obtained using MATLAB at times $\mathbf{p}_{(3)} = 0$ s, $\mathbf{p}_{(3)} = 0.6$ s, and $\mathbf{p}_{(3)} = 1$ s.

a nonlinear diffusion PDE problem, commonly known as the Boussinesq equation, which is chosen to show the applicability of the method to nonlinear parabolic IBVPs.

1) Linear 2-D Heat/Diffusion Equation: The unsteady linear 2-D heat/diffusion equation is

$$\frac{\partial u(\mathbf{p})}{\partial \mathbf{p}_{(3)}} = k_n \left[\frac{\partial^2 u(\mathbf{p})}{\partial \mathbf{p}_{(1)}^2} + \frac{\partial^2 u(\mathbf{p})}{\partial \mathbf{p}_{(2)}^2} \right] \tag{50}$$

where $u(\mathbf{p})$ represents the temperature in the heat equation, or the density in the diffusion equation. The coefficient k_n, which is typically held constant, represents the diffusivity of the material, and determines the rate at which heat or mass is diffused through the system. The domain of the PDE is $(\mathbf{p}_{(1)}, \mathbf{p}_{(2)}) \in \mathcal{H} = [-1, 1] \times [-1, 1]$, and $\mathbf{p}_{(3)} \geq 0$, where $\mathbf{p}_{(3)}$ represents time. The PDE in (50) has Dirichlet BCs

$$u(\mathbf{p}) = 0 \qquad \forall (\mathbf{p}_{(1)}, \mathbf{p}_{(2)}) \in \partial\mathcal{H} \tag{51}$$

and the IC

$$u(\mathbf{p}_{(1)}, \mathbf{p}_{(2)}, 0) = e^{-7(\mathbf{p}_{(1)}^2 + \mathbf{p}_{(2)}^2)} \sin(2\pi\, \mathbf{p}_{(1)}), \quad \forall (\mathbf{p}_{(1)}, \mathbf{p}_{(2)}) \in \mathcal{H} \tag{52}$$

which must be satisfied by the solution everywhere in $\mathcal{H}$, at time $\mathbf{p}_{(3)} = 0$.

The ANN solution takes the form (21), with a user-defined function $q(\mathbf{p}) \equiv (\mathbf{p}_{(1)}^2 - 1)(\mathbf{p}_{(2)}^2 - 1)$. The ANN in (21) is chosen to have 50 LTM nodes and 30 STM nodes (defined in Fig. 1). The input data in $\mathcal{T}_L$ consist of a 30×30 grid of equally spaced points in $\mathcal{H}$, which are used together with the ICs (52) to formulate the equality constraint in (46). The input data in $\mathcal{T}_S$ consist of a $15 \times 15 \times 15$ lattice of points in $\mathcal{H} \times (0, 1]$.

To illustrate the adaptive CPROP solution approach, two PDE problems in (50)–(52) were considered by letting $n = 0, 1$, with $k_0 = 0.01$, and $k_1 = 0.1$. The results in Fig. 7 show sample snapshots of the PDE solutions obtained using MATLAB and CPROP for $n = 0$, at sample moments in time. These results are representative of 20 CPROP solutions obtained using random weight initializations and $K = 120$, all resulting in an REN between the MATLAB and CPROP solutions of $O(10^{-2})$. The results also show that the REN can be further reduced to $O(10^{-3})$ by increasing the number of nodes to $K = 140$. Because benchmark finite element solutions to (50)–(52) can be shown to have an REN of $O(10^{-3})$, however, it is not possible to investigate RENs below this order of magnitude in the absence of an analytical solution.

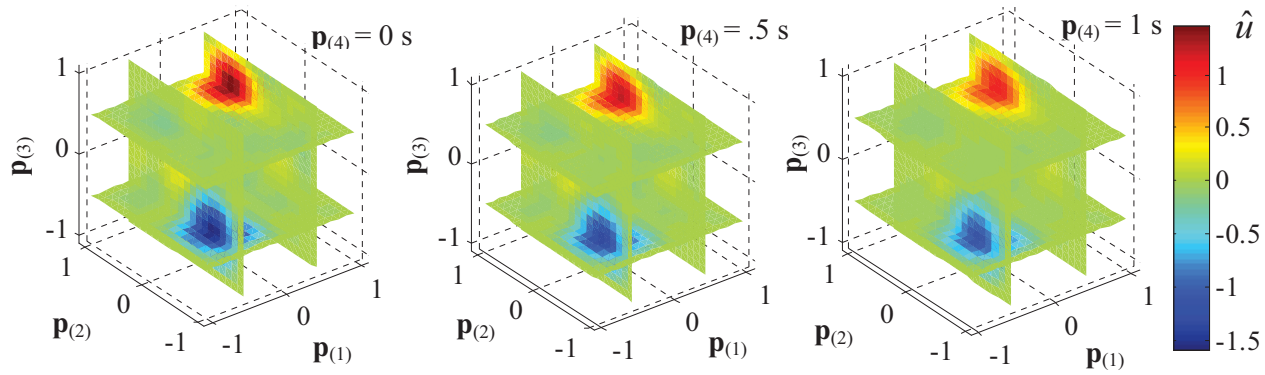


Fig. 9. CPROP solution of the 3-D heat/diffusion equation (53) for $n = 0$.

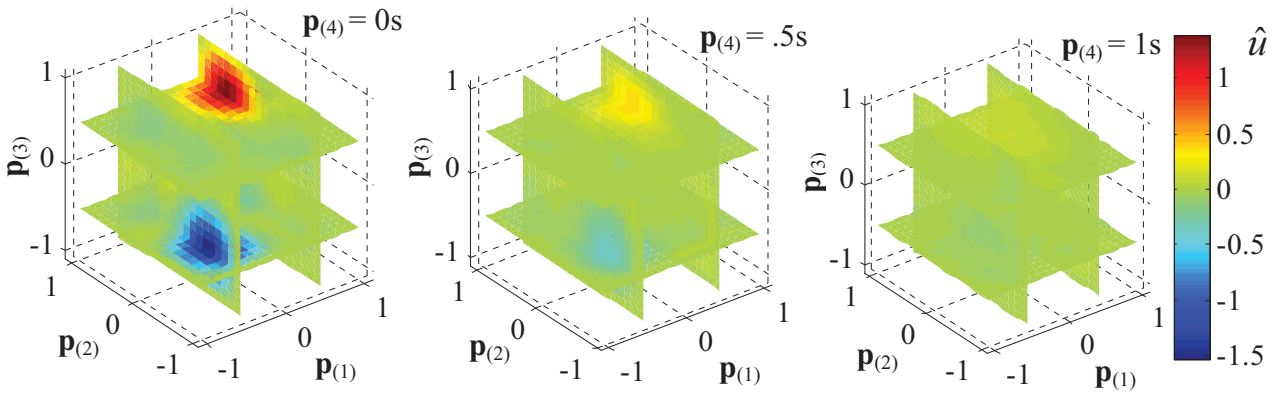


Fig. 10. Adaptive CPROP solution of the 3-D heat/diffusion equation (53) for $n = 1$.

The adaptive CPROP solution obtained for $n = 1$ is plotted and compared with the (nonadaptive) MATLAB solution in Fig. 8. The adaptive CPROP solution was found to rapidly converge to the steady-state zero solution. This type of flat function is one of the hardest to approximate via ANNs because all of the sigmoidal nonlinearities must cancel each other out everywhere in $\mathcal{H} \times (0, 1]$ in order to produce a zero output (temperature). The CPROP solutions plotted in Fig. 8 (for $n = 1$) are representative of 20 adaptive CPROP solutions, all resulting in an REN of $O(10^{-2})$. As in the previous examples, the REN could be further reduced by increasing the number of nodes, and/or decreasing e_{tol}, and by increasing the input data grid size.

2) Linear 3-D Heat/Diffusion Equation: This subsection presents the CPROP results obtained for the parabolic unsteady linear 3-D heat/diffusion equation

$$\frac{\partial u(\mathbf{u})}{\partial \mathbf{p}_{(4)}} = k_n \left[\frac{\partial^2 u(\mathbf{p})}{\partial \mathbf{p}_{(1)}^2} + \frac{\partial^2 u(\mathbf{p})}{\partial \mathbf{p}_{(2)}^2} + \frac{\partial^2 u(\mathbf{p})}{\partial \mathbf{p}_{(3)}^2} \right] \tag{53}$$

where $(\mathbf{p}_{(1)}, \mathbf{p}_{(2)}, \mathbf{p}_{(3)}) \in \mathcal{I} = [-1, 1] \times [-1, 1] \times [-1, 1]$, and time is denoted by $\mathbf{p}_{(4)} \geq 0$. The 3-D heat/diffusion equation in (53) is subjected to the BCs

$$u(\mathbf{p}) = 0 \qquad \forall (\mathbf{p}_{(1)}, \mathbf{p}_{(2)}, \mathbf{p}_{(3)}) \in \partial\mathcal{I} \tag{54}$$

and to the ICs

$$u(\mathbf{p}_{(1)}, \mathbf{p}_{(2)}, \mathbf{p}_{(3)}, 0) = 2\left(e^{-10||\mathbf{p}-\mathbf{p}_0||^2} - e^{-10||\mathbf{p}+\mathbf{p}_0||^2}\right) \quad \forall (\mathbf{p}_{(1)}, \mathbf{p}_{(2)}, \mathbf{p}_{(3)}) \in \mathcal{I} \tag{55}$$

where $\mathbf{p}_0 = [0.5\ 0.5\ 0.5\ 0]^T$ is a known constant vector.

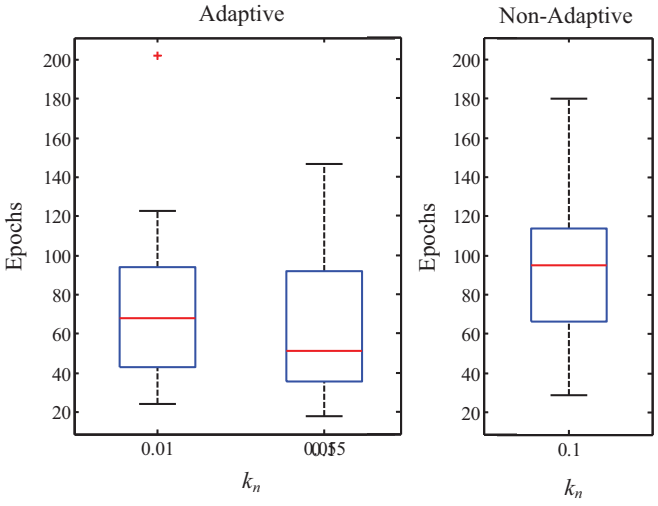


Fig. 11. Box plot of training epochs needed to solve the 3-D heat/diffusion equation (50) adaptively and nonadaptively, using CPROP.

The PDE problem in (53)–(55) is chosen to demonstrate the CPROP method's ability to cope with several variables, and to adapt a 4-D PDE solution to changing parameters, by letting $n = 0, 1$, where $k_0 = 0.01$ and $k_1 = 0.1$. The ANN solution takes the form (21), with a user-defined function $q(\mathbf{p}) = (\mathbf{p}_{(1)}^2 - 1)(\mathbf{p}_{(2)}^2 - 1)(\mathbf{p}_{(3)}^2 - 1)$. The ANN architecture consists of 30 LTM nodes and 60 STM nodes. The input data in $\mathcal{T}_L$ consist of a $25 \times 25 \times 25$ lattice of spatial points in the interior of $\mathcal{I}$, whereas the input data in $\mathcal{T}_S$ consist of an $8 \times 8 \times 8 \times 8$ lattice of points in $\mathcal{H} \times (0, 1]$.

The CPROP solution obtained using random weight initialization, for $n = 0$, is plotted in Fig. 9. The ANN solution in Fig. 9 is then adapted by the CPROP algorithm for $n = 1$, obtaining the solution plotted in Fig. 10. As for the previous examples, the CPROP solution was found to converge to a satisfactory objective function ($e < e_{\text{tol}}$), and to the known steady-state solution, $u = 0$. To illustrate the computational savings brought about by the adaptive CPROP solution, the 3-D heat/diffusion PDE problem (53)–(55) with $n = 1$ was also solved 20 times using random initial weights (nonadaptively). As shown in Fig. 11, it was found that the adaptive CPROP solution significantly decreases the number of epochs required by benefiting from its knowledge of the previous solution ($n = 0$) through incremental training.

3) Nonlinear 3-D Heat/Diffusion, or Boussinesq, Equation: The Boussinesq equation is a model of heat/diffusion process with nonlinear diffusive properties that is used extensively in numerical groundwater flow simulations [32], and can be written as

$$S_n \frac{\partial u}{\partial \mathbf{p}_{(3)}} = \frac{\partial}{\partial \mathbf{p}_{(1)}}\left[K_n u \frac{\partial u}{\partial \mathbf{p}_{(1)}}\right] + \frac{\partial}{\partial \mathbf{p}_{(2)}}\left[K_n u \frac{\partial u}{\partial \mathbf{p}_{(2)}}\right] \quad (56)$$

where u is the elevation of the water table above a horizontal base, the spatial coordinates are $(\mathbf{p}_{(1)}, \mathbf{p}_{(2)}) \in \mathcal{H} = [-1, 1] \times [-1, 1]$, and $\mathbf{p}_{(3)} \geq 0$ is time. S_n is the specific yield, or the amount of water released per volume of porous medium when changing from a saturated state to an unsaturated state high above the water table, and K_n is the hydraulic conductivity.

To demonstrate the ability of CPROP for solving nonlinear parabolic IBVPs adaptively, the specific yield and hydraulic connectivity are modeled here as nonlinear functions of the PDE variables $\mathbf{p}_{(1)}$ and $\mathbf{p}_{(2)}$, as follows:

$$S_n \equiv 0.2\left(1 + \beta_n \frac{e^{10\mathbf{p}_{(1)}+2\mathbf{p}_{(2)}} - 1}{e^{10\mathbf{p}_{(1)}+2\mathbf{p}_{(2)}} + 1}\right) \quad (57)$$

$$K_n \equiv 0.0002\left(1 - \beta_n \frac{e^{10\mathbf{p}_{(1)}+2\mathbf{p}_{(2)}} - 1}{e^{10\mathbf{p}_{(1)}+2\mathbf{p}_{(2)}} + 1}\right) \quad (58)$$

also rendering the PDE problem considerably more challenging. The Boussinesq PDE in (56) has the ICs

$$\begin{aligned} u(\mathbf{p}_{(1)}, \mathbf{p}_{(2)}, 0) = {} & 10 + 9\sin(\pi(\mathbf{p}_{(1)}\mathbf{p}_{(2)} + \mathbf{p}_{(1)}^2)) \\ & \times \cos(2\pi(\mathbf{p}_{(2)}^2 + .1))(\mathbf{p}_{(1)}^2 - 1)(\mathbf{p}_{(2)}^2 - 1) \\ & \times \exp(-\sin^2(2\pi\,\mathbf{p}_{(2)})) \quad \forall (\mathbf{p}_{(1)}, \mathbf{p}_{(2)}) \in \mathcal{H} \end{aligned} \quad (59)$$

and the Dirichlet BCs

$$u(\mathbf{p}) = 10 \qquad \forall (\mathbf{p}_{(1)}, \mathbf{p}_{(2)}) \in \partial\mathcal{H}. \quad (60)$$

The MATLAB PDE toolbox is only capable of solving linear parabolic PDEs. Therefore, in this section, the CPROP solution is compared with the numerical solution obtained using FDM. The implemented FDM scheme discretizes the PDE domain by letting

$$u_{i,j}^m = u(-1 + i\,\Delta\mathbf{p}_{(1)}, -1 + j\,\Delta\mathbf{p}_{(2)}, m\,\Delta\mathbf{p}_3) \quad (61)$$

denote a pointwise solution, and using a forward stepping temporal difference to approximate $\partial u/\partial \mathbf{p}_{(3)}$, with central differencing for spatial derivatives. The FDM stencil is shown in Appendix VII. The PDE domain, $\mathcal{H}$, is discretized using small increments, i.e., $\Delta\mathbf{p}_{(3)} = 5e^{-4}$ and $\Delta\mathbf{p}_{(1)} = \Delta\mathbf{p}_{(2)} = 0.02$ to avoid stability issues exhibited by strictly explicit FDM schemes applied to nonlinear PDEs.

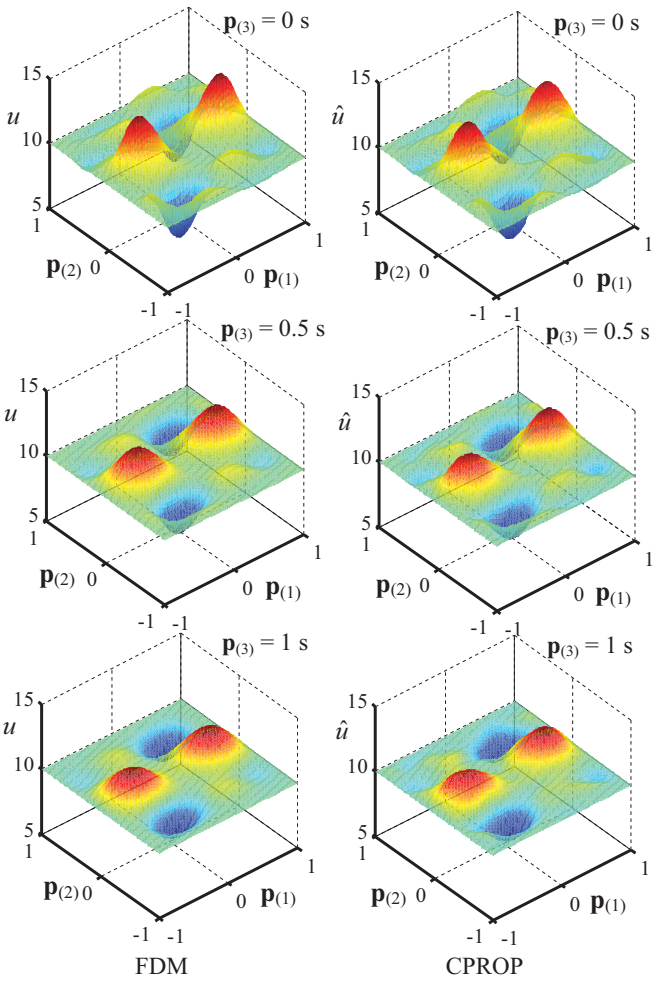


Fig. 12. Solution of Boussinesq PDE (56) obtained by CPROP when $n = 0$ is compared with FDM solution at $\mathbf{p}_{(3)} = 0$ s, $\mathbf{p}_{(3)} = 0.5$ s, and $\mathbf{p}_{(3)} = 1$ s.

The CPROP solution of the Boussinesq PDE problem (56)–(60) is obtained using input data in $\mathcal{T}_L$ that consist of a 40×40 grid in $\mathcal{H}$, and input data in $\mathcal{T}_S$ that consist of a $20 \times 20 \times 20$ lattice in $\mathcal{H} \times (0, 1]$, with 110 LTM nodes and 30 STM nodes. Because the ICs in (59) are more challenging to approximate than the ICs of earlier examples, a larger number of LTM nodes, and a larger input data set $\mathcal{T}_L$ were required to achieve the desired accuracy (e_{tol}). To illustrate the adaptive CPROP solution, the parameter β_n in (57) and (58) is varied from $\beta_0 = 0$ ($n = 0$) to $\beta_1 = 0.5$ ($n = 1$), obtaining two Boussinesq PDE problems.

The results in Fig. 12 show sample snapshots of the PDE solutions obtained using FDM and CPROP for $n = 0$, at sample moments in time. These results are representative of 20 CPROP solutions obtained using random weight initializations and $K = 140$, all resulting in an REN between the FDM and CPROP solutions of $O(10^{-2})$. The REN could be further reduced by increasing the number of nodes and/or decreasing e_{tol}, and by increasing the input data grid size. The adaptive

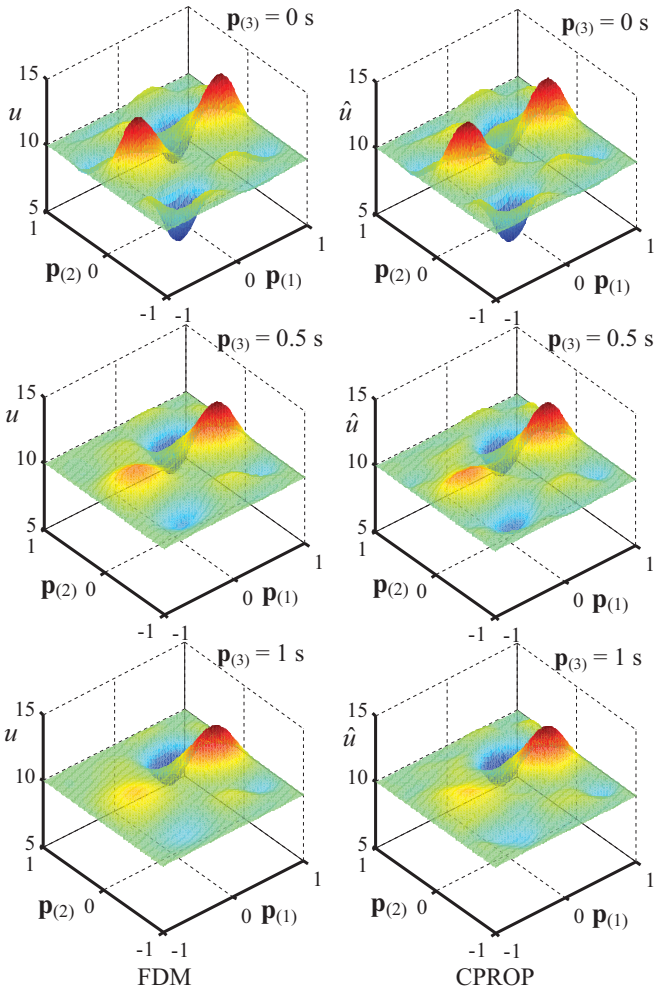


Fig. 13. Adaptive solution of Boussinesq PDE (56) obtained by CPROP when $n = 1$ is compared with (nonadaptive) FDM solution at $\mathbf{p}_{(3)} = 0$ s, $\mathbf{p}_{(3)} = 0.5$ s, and $\mathbf{p}_{(3)} = 1$ s.

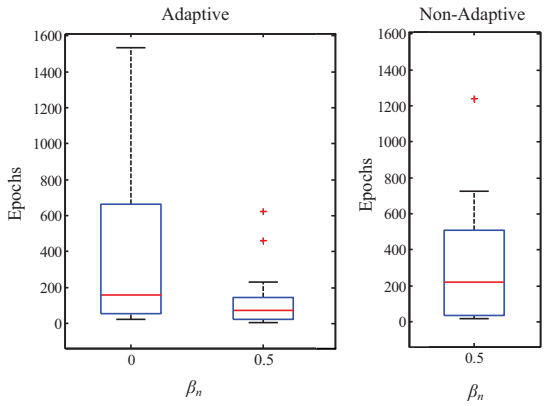


Fig. 14. Box plot of number of training epochs needed to solve the Boussinesq PDE (56) adaptively and nonadaptively using CPROP.

CPROP solution obtained for $n = 1$ is plotted and compared with the (nonadaptive) FDM solution in Fig. 13.

The Boussinesq PDE problem (56)–(60) with $n = 1$ was also solved 20 times nonadaptively (with random weight initializations) using CPROP. As shown in Fig. 14, it was found that the adaptive CPROP solution significantly decreases the number of epochs required by benefiting from its knowledge of the previous solution ($n = 0$). Furthermore, by comparing Figs. 14 with 11, it can be seen that, for this nonlinear 3-D heat/diffusion PDE problem, the computational savings were even more significant than for the linear 3-D heat/diffusion PDE problem in (53)–(55).

VII. Conclusion

ANNs have been used in a number of applications to provide functional representations of PDE solutions that are amenable to mathematical analysis, and to more efficient processing by data assimilation and estimation algorithms. In many of these applications, however, the PDE parameters and/or external forcing may be subject to change. CPROP offers a natural paradigm for solving PDEs adaptively over time, because the ANN solution can be adapted to minimize the error defined by the differential operator, while satisfying I/BCs through direct elimination. In this paper, the effectiveness of the CPROP solution method is demonstrated through several examples of linear and nonlinear elliptic and parabolic PDEs, subject to I/BCs. The numerical results show that the CPROP methodology can be used to solve elliptic BVPs and parabolic IBVPs adaptively, with excellent accuracy. Furthermore, CPROP eliminates the need for user intervention, as required by the FDM-based method, and requires less weights, less collocation points, and less training epochs than the penalty function method, because it reduces the dimensionality of the optimization problem using direct elimination. For both elliptic and parabolic equations, CPROP brings about a significant reduction in the number of iterations required for solving the PDE adaptively, and is characterized by a computational complexity and a solution accuracy that compare favorably with the existing methods of solution. Finally, as will be shown in a separate paper, the method can be extended to irregular domains, and to other classes of PDEs, including hyperbolic equations.

Appendix A
IBVP Partial Derivative

The second partial derivative of the ansatz to the parabolic IBVP is given by

$$\begin{aligned}\frac{\partial^2 \hat{u}(\mathbf{p})}{\partial \mathbf{p}_{(j)} \partial \mathbf{p}_{(k)}} &= \frac{\partial^2 \tilde{h}(\mathbf{p})}{\partial \mathbf{p}_{(j)} \partial \mathbf{p}_{(k)}} + \frac{\partial^2 q(\mathbf{p})}{\partial \mathbf{p}_{(j)} \partial \mathbf{p}_{(k)}} \\ &\quad \times [\boldsymbol{\Phi}(\mathbf{p}^T \mathbf{W}_L^T + \mathbf{b}_L^T)\mathbf{V}_L^T + \boldsymbol{\Phi}(\mathbf{p}^T \mathbf{W}_S^T + \mathbf{b}_S^T)\mathbf{V}_S^T] \\ &\quad + \frac{\partial q(\mathbf{p})}{\partial \mathbf{p}_{(j)}} [\boldsymbol{\Phi}^1(\mathbf{p}^T \mathbf{W}_L^T + \mathbf{b}_L^T)\boldsymbol{\omega}_{L_k} \mathbf{V}_L^T \\ &\qquad + \boldsymbol{\Phi}^1(\mathbf{p}^T \mathbf{W}_S^T + \mathbf{b}_S^T)\boldsymbol{\omega}_{S_k} \mathbf{V}_S^T] \\ &\quad + \frac{\partial q(\mathbf{p})}{\partial \mathbf{p}_{(k)}} [\boldsymbol{\Phi}^1(\mathbf{p}^T \mathbf{W}_L^T + \mathbf{b}_L^T)\boldsymbol{\omega}_{L_j} \mathbf{V}_L^T \\ &\qquad + \boldsymbol{\Phi}^1(\mathbf{p}^T \mathbf{W}_S^T + \mathbf{b}_S^T)\boldsymbol{\omega}_{S_j} \mathbf{V}_S^T] \\ &\quad + q(\mathbf{p}) [\boldsymbol{\Phi}^2(\mathbf{p}^T \mathbf{W}_L^T + \mathbf{b}_L^T \boldsymbol{\omega}_{L_j} \boldsymbol{\omega}_{L_k} \mathbf{V}_L^T \\ &\qquad + \boldsymbol{\Phi}^2(\mathbf{p}^T \mathbf{W}_S^T + \mathbf{b}_S^T)\boldsymbol{\omega}_{S_j} \boldsymbol{\omega}_{S_k} \mathbf{V}_S^T]. \end{aligned} \quad (62)$$

APPENDIX B
FDM STENCIL

The FDM stencil used to solve the Boussinesq equation is given by

$$S^m_{n_{i,j}} \frac{\left(u^{(m+1)}_{i,j} - u^{(m)}_{i,j}\right)}{\Delta \mathbf{p}_{(3)}} = \left(\frac{\partial K_n}{\partial \mathbf{p}_{(1)}}\right)^m_{i,j} u^m_{i,j} \frac{\left(u^{(m)}_{i+1,j} - u^{(m)}_{i-1,j}\right)}{2\Delta \mathbf{p}_{(1)}} + \left(\frac{\partial K_n}{\partial \mathbf{p}_{(2)}}\right)^m_{i,j} u^m_{i,j} \frac{\left(u^{(m)}_{i,j+1} - u^{(m)}_{i,j+1}\right)}{2\Delta \mathbf{p}_{(2)}} + K^m_{n_{i,j}} \left[\left(\frac{u^m_{i+1,j} - u^m_{i-1,j}}{2\Delta \mathbf{p}_{(1)}}\right)^2 + \left(\frac{u^m_{i,j+1} - u^m_{i,j-1}}{2\Delta \mathbf{p}_{(2)}}\right)^2\right] + K^m_{n_{i,j}} u^m_{i,j} \left(\frac{u^m_{i+1,j} - 2u^m_{i,j} + u^m_{i-1,j}}{\Delta \mathbf{p}^2_{(1)}} + \frac{u^m_{i,j+1} - 2u^m_{i,j} + u^m_{i,j-1}}{\Delta \mathbf{p}^2_{(2)}}\right). \quad (63)$$

REFERENCES

[1] I. Lagaris, A. Likas, and D. I. Fotiadis, "Artificial neural networks for solving ordinary and partial differential equations," *IEEE Trans. Neural Netw.*, vol. 9, no. 5, pp. 987–1000, Sep. 1998.

[2] T. Cheng, F. L. Lewis, and M. Abu-Khalaf, "Fixed-final-time-constrained optimal control of nonlinear systems using neural network HJB approach," *IEEE Trans. Neural Netw.*, vol. 18, no. 6, pp. 1725–1737, Nov. 2007.

[3] G. D. Smith, *Numerical Solution of Partial Differential Equations: Finite Difference Methods*. Oxford, U.K.: Clarendon Press, 1978.

[4] T. J. R. Hughes, *The Finite Element Method*. Upper Saddle River, NJ, USA: Prentice-Hall, 1987.

[5] W. Press, S. Flannery, S. Teukolsky, and W. Vetterling, *Numerical Recipes: The Art of Scientific Computing*. Cambridge, U.K.: Cambridge Univ. Press, 1986.

[6] C. Saloma, "Computational complexity and the observation of physical signals," *J. Appl. Phys.*, vol. 74, no. 9, pp. 5314-1–5314-6, 1993.

[7] *MATLAB Neural Network Toolbox, User's Guide*. Natick, MA, USA: MathWorks, 2005.

[8] I. Lagaris, A. Likas, and D. Papageorgio, "Neural-network methods for boundary value problems with irregular boundaries," *IEEE Trans. Neural Netw.*, vol. 11, no. 5, pp. 1041–1049, Sep. 2000.

[9] Y. Shirvany, M. Hayati, and R. Moradian, "Multilayer perceptron neural networks with novel unsupervised training method for numerical solution of the partial differential equations," *Appl. Soft Comput.*, vol. 9, no. 1, pp. 20–29, 2009.

[10] K. Levenberg, "A method for the solution of certain non-linear problems in least squares," *Quart. J. Appl. Math.*, vol. 2, no. 2, pp. 164–168, 1944.

[11] R. Shekari Beidokhti, and A. Malek, "Solving initial-boundary value problems for systems of partial differential equations using neural networks and optimization techniques," *J. Franklin Inst.*, vol. 346, pp. 1–11, Aug. 2009.

[12] K. S. McFall and J. R. Mahan, "Artificial neural network method for solution of boundary value problems with exact satisfaction of arbitrary boundary conditions," *IEEE Trans. Neural Netw.*, vol. 20, no. 8, pp. 1221–1233, Aug. 2009.

[13] M. W. M. G. Dissanayake and N. Phan-Thien, "Neural-network-based aproximations for solving partial differential equations," *Commun. Numer. Methods Eng.*, vol. 10, no. 3, pp. 195–201, 1994.

[14] R. F. Stengel, *Optimal Control and Estimation*. New York, NY, USA: Dover Publications, 1986.

[15] D. P. Bertsekas, *Constrained Optimization and Lagrange Multiplier Methods*. Belmont, MA, USA: Athena Scientific, 1996.

[16] S. Ferrari and M. Jensenius, "A constrained optimization approach to preserving prior knowledge during incremental training," *IEEE Trans. Neural Netw.*, vol. 19, no. 6, pp. 996–1009, Jun. 2008.

[17] S. Ferrari and R. Stengel, "Smooth function approximation using neural networks," *IEEE Trans. Neural Netw.*, vol. 16, no. 1, pp. 24–38, Jan. 2005.

[18] G. H. Bower, *The Psychology of Learning and Motivation: Advances in Research and Theory*. San Diego, CA, USA: Academic Press, 1989.

[19] G. D. Muro and S. Ferrari, "A constrained-optimization approach to training neural networks for smooth function approximation and system identification," in *Proc. Int. Joint Conf. Neural Netw.*, 2008, pp. 2353–2359.

[20] M. Husken, C. Goerick, and A. Vogel, "Fast adaptation of the solution of differential equations to changing constraints," in *Proc. 2nd ICSC Symp. Neural Comput.*, 2000, pp. 181–187.

[21] J. Attali and G. Pages, "Approximations of functions by a multilayer perceptron: A new approach," *Neural Netw.*, vol. 10, no. 6, pp. 1069–1081, 1997.

[22] X. Li, "Simultaneous approximations of multivariate functions and their derivatives by neural networks with one hidden layer," *Neurocomputing*, vol. 12, no. 4, pp. 327–343, 1996.

[23] K. Hornik, M. Stinchcombe, and H. White, "Universal approximation of an unknown mapping and its derivatives using multilayer feedforward networks," *Neural Netw.*, vol. 3, no. 5, pp. 551–560, 1990.

[24] P. Werbos, "Backwards differentiation in AD and neural nets: Past links and new opportunities," in *Automatic Differentiation: Applications, Theory, and Implementations*, vol. 50. M. BAcker, G. Corliss, U. Naumann, P. Hovland, and B. Norris, Eds. New York, NY, USA: Springer-Verlag, Berlin Heidelberg, 2006, pp. 15–34.

[25] D. W. Marquardt, "An algorithm for least-squares estimation of nonlinear parameters," *J. Soc. Ind. Appl. Math.*, vol. 11, no. 2, pp. 431–441, 1963.

[26] P. O'Neil, *Advanced Engineering Mathematics*. Stamford, CT, USA: Cengage Learning, 2012.

[27] L. Rogers and F. Dowla, "Optimization of groundwater remediation using artificial neural networks with parallel solute transport modeling," *Water Resour. Res.*, vol. 30, no. 2, pp. 457–481, 1994.

[28] S. W. Liu, J. H. Huang, J. C. Sung, and C. C. Lee, "Detection of cracks using neural networks and computational mechanics," *Comput. Methods Appl. Mech. Eng.*, vol. 191, no. 25, pp. 2831–2845, 2000.

[29] *MATLAB Partial Differential Equation Toolbox, User's Guide*. Natick, MA, USA: MathWorks, 2005.

[30] B. Igelnik and Y.-H. Pao, "Stochastic choice of basis functions in adaptive function approximation and the functional-link net," *IEEE Trans. Neural Netw.*, vol. 6, no. 6, pp. 1320–1329, Nov. 1995.

[31] W. Liu, J. Sarangapani, G. Venayagamoorthy, D. Wunsch, and D. Cartes, "Neural network based decentralized excitation control of large scale power systems," in *Proc. IJCNN*, 2006, pp. 1975–1981.

[32] J. W. Delleur, *Groundwater Engineering*. Boca Raton, FL, USA: CRC Press, 2007.

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 math from Northwestern University, Evanston, IL, USA, in 2008, and the Master of Engineering Management degree from Duke University, Durham, NC, USA, in 2010, where he is currently pursuing the Ph.D. degree.

Gianluca Di Muro received the Laurea degree in aerospace engineering from La Sapienza University of Rome, Rome, Italy, in 2003, and the M.S. degree in mechanical engineering from Duke University, Durham, NC, USA, in 2009.

Silvia Ferrari (S'01–M'02–SM'08) received the B.S. degree from Embry-Riddle Aeronautical University, Daytona Beach, FL, USA, and the M.A. and Ph.D. degrees from Princeton University, Princeton, NJ, USA.

She is the Paul Ruffin Scarborough Associate Professor of engineering with Duke University, Durham, NC, USA, where she directs the Laboratory for Intelligent Systems and Controls. Her current research interests include robust adaptive control of aircraft, learning and approximate dynamic programming, and optimal control of mobile sensor networks.

Dr. Ferrari is a member of ASME, SPIE, and AIAA. She is a recipient of the ONR Young Investigator Award in 2004, the National Science Foundation CAREER Award in 2005, and the Presidential Early Career Award for Scientists and Engineers in 2006.