



# A novel method for enforcing exactly dirichlet, Neumann and robin conditions on general quadrilateral domains with curved boundaries for physics informed machine learning

Suchuan Dong<sup>\*</sup>, Yuchuan Zhang<sup>id</sup>

Center for Computational and Applied Mathematics, Department of Mathematics, Purdue University, USA

## ARTICLE INFO

### Keywords:

Exact boundary condition enforcement  
Physics informed machine learning  
Scientific machine learning  
Transfinite interpolation  
Theory of functional connections  
Complex geometry

## ABSTRACT

We present a systematic method for exactly enforcing Dirichlet, Neumann, and Robin type conditions on general quadrilateral domains with arbitrary curved boundaries. Our method is built upon exact mappings between general quadrilateral domains and the standard domain, and employs a combination of TFC (theory of functional connections) constrained expressions and transfinite interpolations. When Neumann or Robin boundaries are present, especially when two Neumann (or Robin) boundaries meet at a vertex, it is critical to enforce exactly the induced compatibility constraints at the intersection, in order to enforce exactly the imposed conditions on the joining boundaries. We analyze in detail and present constructions for handling the imposed boundary conditions and the induced compatibility constraints for two types of situations: (i) when Neumann (or Robin) boundary only intersects with Dirichlet boundaries, and (ii) when two Neumann (or Robin) boundaries intersect with each other. We describe a four-step procedure to systematically formulate the general form of functions that exactly satisfy the imposed Dirichlet, Neumann, or Robin conditions on general quadrilateral domains. The method developed herein has been implemented together with the extreme learning machine (ELM) technique we have developed recently for scientific machine learning. Ample numerical experiments are presented with several linear/nonlinear stationary/dynamic problems on a variety of two-dimensional domains with complex boundary geometries. Simulation results demonstrate that the proposed method has enforced the Dirichlet, Neumann, and Robin conditions on curved domain boundaries exactly, with the numerical boundary-condition errors at the machine accuracy.

## 1. Introduction

Artificial neural networks (NN) have garnered remarkable success in diverse fields of science and engineering [1,2]. This has catalyzed the development and adoption of neural network-based techniques for scientific computing. As universal function approximators, neural networks are natural for the ansatz space for solving ordinary or partial differential equations (ODE/PDE). This underpins their use in mathematical modeling and scientific computing, and has fueled the advancement of scientific machine learning [3].

The use of NNs for computing ODEs/PDEs dates back to the 1990s (see [4–9]). Leveraging the universal approximation property [10–12], NN-based methods typically transform the PDE problem into an optimization problem, thanks to the residual

<sup>\*</sup> Corresponding author.

E-mail address: [sdong@purdue.edu](mailto:sdong@purdue.edu) (S. Dong).

<https://doi.org/10.1016/j.cma.2026.119302>

Received 23 March 2026; Received in revised form 4 August 2026; Accepted 7 August 2026

0045-7825/© 2026 Elsevier B.V. All rights are reserved, including those for text and data mining, AI training, and similar technologies.

minimization theorem as elaborated in [13]. The PDE and its boundary and initial conditions (BC/IC) are encoded into a cost/loss function by penalizing their residual norms on a set of sampling points [14]. This general residual minimization technique [13] is presently often known as the physics-informed approach for solving PDEs. The optimization, usually by gradient descent-type algorithms, constitutes the core computations in NN-based PDE solvers, commonly known as NN training. After training, the NN parameters effectively encode the PDE solution.

Prominent advancements of this area in recent years include the development of physics-informed neural network (PINN) method [15] and sister approaches such as the deep Galerkin method (DGM) [16] and deep Ritz method [17], as well as related methods such as the weak adversarial network [18], Galerkin neural network [19], deep Nitsche method [20], deep mixed residual method [21], along with many variant techniques (see e.g. [22–38], among others). These methods have been applied across a wide range of fields; see comprehensive reviews of this area in [3,39–41].

Enforcing boundary conditions is one of the central technical issues in physics informed machine learning. Unlike classical numerical methods, standard NN ansätze are non-interpolatory and do not automatically satisfy prescribed data on the boundary. As a result, a substantial literature has developed on how to enforce Dirichlet, Neumann, Robin, and periodic BCs in PINN and related neural PDE solvers. Existing approaches can be classified into four broad categories: (i) soft or penalty-based enforcement, (ii) exact or hard enforcement through trial function design, (iii) weak enforcement in variational formulations, and (iv) multiplier-based constrained formulations.

The original PINN adopts soft enforcement, in which the PDE residual is minimized together with BC residual terms sampled on the boundary [15]. This strategy is simple and broadly applicable, but BC satisfaction is only approximate and training can be highly sensitive to the relative weights of the PDE and BC loss terms. This difficulty is analyzed from the perspective of gradient-flow pathologies in PINNs [42]; see also [43] for a recent comparative study of BC-enforcement strategies on three-dimensional geometries.

A second major line of work seeks exact or hard enforcement, in which the neural approximation is constructed to satisfy the BCs identically. This idea goes back to the early works of Lagaris, McFall and collaborators, who introduced neural trial functions of the form  $u_\theta = g + \phi N_\theta$ , with  $g$  satisfying the boundary data and  $\phi$  vanishing on the constrained boundary [9,44,45]. In more recent literature, it has been shown that boundary and initial conditions can be embedded directly into the ansatz, thereby eliminating BC penalty terms from the loss in forward and inverse settings [46–51].

Among exact-enforcement methods, the geometry-aware technique of Sukumar and Srivastava is particularly influential [52]. Using approximate distance functions (ADF) and the theory of R-functions, they constructed admissible neural trial spaces that can enforce Dirichlet, Neumann, and Robin conditions exactly. A comparative study of several approaches has been conducted in [53], which concludes that exact output modification is generally superior to penalty-only training for Dirichlet conditions. A number of follow-on works extend or specialize this perspective; see e.g. [54] for exact Dirichlet enforcement in solid mechanics and [55] for hybrid hard/soft Fourier-based treatment in advection–diffusion problems.

Although the distance-function framework of [52] is elegant, this comes with certain limitations. This method is especially effective for Dirichlet conditions. Its extension to Neumann and Robin conditions, however, is more delicate because derivative boundary operators require higher regularity of the trial space and of the underlying geometric representation. In particular, follow-on work has noted that exact enforcement using approximate distance functions becomes more challenging for higher-order PDEs, and recent studies have further pointed out that strong Neumann/Robin constructions may become unstable when boundary segments are only piecewise  $C^1$  rather than globally  $C^1$  [56–58]. A further subtlety concerns vertices or corner points where two boundary segments meet. The true Euclidean distance function is generally not differentiable at such points, which is precisely why the method relies on approximate distance functions constructed through R-functions and related smooth implicit representations. Accordingly, differentiability of the resulting trial function at a vertex is not automatic in a classical geometric sense; rather, it depends on the regularity built into the approximate distance construction. For Dirichlet conditions this is often sufficient, since exact trace satisfaction is the primary requirement. For Neumann/Robin conditions, however, the issue is more fundamental: at a corner where two Neumann or Robin boundaries meet the outward normal is typically not uniquely defined. So the boundary operator itself becomes ambiguous unless additional compatibility constraints are introduced. Thus, while the method is a powerful strategy its application to Neumann and Robin conditions on non-smooth geometries requires additional geometric and analytical care, and corner singularities remain a genuine limitation rather than a purely technical detail [57].

Another strategy for exact enforcement is based on the Theory of Functional Connections (TFC) [59,60]. In TFC, the boundary or initial conditions are embedded analytically into a constrained expression, leaving the neural network to represent only the free function [61,62]. The multivariate formulation in [60] extends the original TFC framework [59] from one dimension to rectangular multivariate domains by constructing constrained expressions that satisfy the prescribed boundary value or derivative data, which generalizes the Coons-type interpolation to a broader analytical framework. While TFC provides an effective mechanism for BC enforcement by analytically embedding linear equality constraints, its main limitation is geometric flexibility. In its multivariate form, TFC is most natural on tensor-product domains such as rectangles and hyper-rectangles, where the constrained expressions can be built in separable coordinates [60,61]. Non-rectangular domains generally require additional bijective mappings to a rectangular domain [63]. Three mapping strategies are discussed in [63]: complex (conformal) mapping, projection mapping, and polynomial mapping. Complex mapping uses an analytic complex-variable transformation, which is smooth and preserves angles. Its limitation is universality because in practice a suitable complex map is not always available for a target geometry. Projection mapping constructs the transformation geometrically from projection points, which has a broader geometric applicability. But the mapping may fail to be  $C^1$ , which becomes less attractive for PDE problems in strong-form formulations. Polynomial mapping builds the geometry map from switching polynomials and control points, which is more flexible for representing complex shapes. That flexibility, however,

comes at the cost of exactness, since the target geometry may be represented only approximately. Recent reduced-TFC work [64] explicitly motivates itself by improved efficiency and the ability for more complex boundary geometries. Another issue arises with TFC when the problem dimension increases. The constrained expressions become increasingly cumbersome in higher dimensions, with the number of TFC terms growing exponentially [65].

Periodic BCs have motivated a distinct line of research for exact-enforcement techniques. A key work here is [66], which introduced periodic layers for feed-forward NNs that impose exactly  $C^\infty$ -periodic or  $C^k$ -periodic BCs, making periodicity an architectural property of the ansatz rather than a loss penalty. See also [67] for a recent structure-preserving extension using embedded periodic layers to geometric-flow problems.

A third category is weak enforcement in variational neural PDE techniques. In Deep Ritz method and in variational PINNs, the PDE is enforced through an energy or weak residual rather than through strong-form collocation [17,68]. In this setting, Neumann conditions often arise naturally through integration by parts, whereas Dirichlet conditions remain essential constraints that must be imposed separately. The Nitsche's method was adapted to this setting and a Deep Nitsche method was developed for essential BCs in [20].

A fourth category consists of multiplier-based constrained formulations. Rather than enforcing BCs by fixed penalties, these methods introduce auxiliary Lagrange multipliers or saddle-point formulations. Makridakis et al. recently proposed a Deep Uzawa approach for BC enforcement in PINNs and Deep Ritz methods [69]. For Neumann conditions, specialized architectural variants are also beginning to emerge; see e.g. [70] for a recent hard-constraint treatment based on embedded Fourier features.

In the current paper we present a systematic method for enforcing exactly Dirichlet, Neumann and Robin type conditions on general quadrilateral domains with arbitrary curved boundaries. This method is based on exact mappings between general quadrilateral domains and the standard domain, and leverages a combination of TFC constrained expressions [59] and the transfinite interpolations developed by Gordon and collaborators [71,72] for both the domain mapping and the trial-function formulation. The resultant trial ansätze are in parametric forms, formulated in terms of the standard domain. The formulation for exactly enforcing Dirichlet BCs on general quadrilateral domains is conceptually straightforward, once the exact domain mapping is achieved. As a matter of fact, the mapping problem between a general quadrilateral domain and the standard domain itself is treated as a problem involving solely Dirichlet boundary conditions, and is formulated by a combination of TFC constrained expression and transfinite interpolation. When Neumann or Robin boundaries are present over the domain, especially when two Neumann (or Robin) boundaries intersect with each other, the formulation becomes considerably more challenging. In this case, the TFC constrained expression and the transfinite interpolation need to be modified in order to exactly enforce not only these conditions, but also the compatibility constraints at the intersecting vertex induced by these conditions. Enforcing exactly the induced compatibility constraints at the intersection is critical, because otherwise the Dirichlet, Neumann or Robin conditions on the adjacent boundaries fail to be exactly satisfied. We analyze in detail and present formulations to handle the induced compatibility constraints for two types of situations: (i) when Neumann (or Robin) boundaries only intersect with Dirichlet boundaries, and (ii) when two Neumann (or Robin) boundaries intersect with each other. We present a four-step procedure for systematically formulating the general form of trial functions to satisfy these conditions and their compatibility constraints. When a combination of Dirichlet, Neumann, and Robin boundaries are present, the induced compatibility constraints can be decomposed into those of the aforementioned cases and the trial function can be formulated analogously based on the four-step procedure.

The method proposed herein for exact BC enforcement has been implemented with the extreme learning machine (ELM) technique we have developed previously [65,73–78]. ELM is a scientific machine learning approach based on randomized feedforward NNs, in which the hidden-layer coefficients are randomly assigned and fixed (non-trainable) and only the output-layer coefficients are trained. The ELM network is trained by the linear least squares method for linear problems or by the nonlinear least squares method (Gauss–Newton method) for nonlinear problems [73,79]. A sizeable volume of literature exists on ELM and variant techniques (with different aliases). We refer the reader to e.g. [62,80–90] (among others) for contributions from other researchers to this area.

Extensive numerical experiments have been conducted using several linear/nonlinear stationary/dynamic PDEs on a variety of domains with complex boundary geometries. Simulations demonstrate that the ELM network together with the current method for BC enforcement has produced highly accurate results. In particular, numerical results show that the current method has enforced the Dirichlet, Neumann, and Robin boundary conditions on curved domain boundaries to the machine accuracy.

The fundamental contribution of this work lies in the systematic method for formulating trial functions that exactly satisfy the Dirichlet, Neumann, and Robin type conditions on general quadrilateral domains with arbitrary curved boundaries. The analyses and formulations for enforcing the induced compatibility constraints and the BCs, especially when two Neumann (or Robin) boundaries intersect with each other or when a Neumann/Robin boundary and a Dirichlet boundary meet at a vertex, are particularly important. Another contribution is the numerical demonstration of the effectiveness of the proposed method for BC enforcement, with the numerical errors for Dirichlet, Neumann, and Robin BCs on curved domain boundaries achieving machine precision. We would like to emphasize that the method for BC enforcement presented here can be used with other NN architectures and training algorithms (e.g. PINNs), not limited to ELM or randomized neural networks. Because the boundary conditions are enforced exactly with the solution ansatz, it is agnostic to the NN used for learning the arbitrary free function therein.

The rest of this paper is organized as follows. In Section 2 we discuss how to map a general quadrilateral with arbitrary curved boundaries to the standard domain, and present a four-step procedure to systematically formulate trial functions on such domains to satisfy the imposed Dirichlet, Neumann, and Robin conditions. In Section 3 we present numerical examples of linear/nonlinear boundary/initial value problems to demonstrate the effectiveness of the proposed method on a variety of domains with complex geometries. Section 4 concludes the presentation with a summary of the key points. The Appendix provides the geometric parameters for all the problem domains from Section 3 and further tests on the simulation parameters.

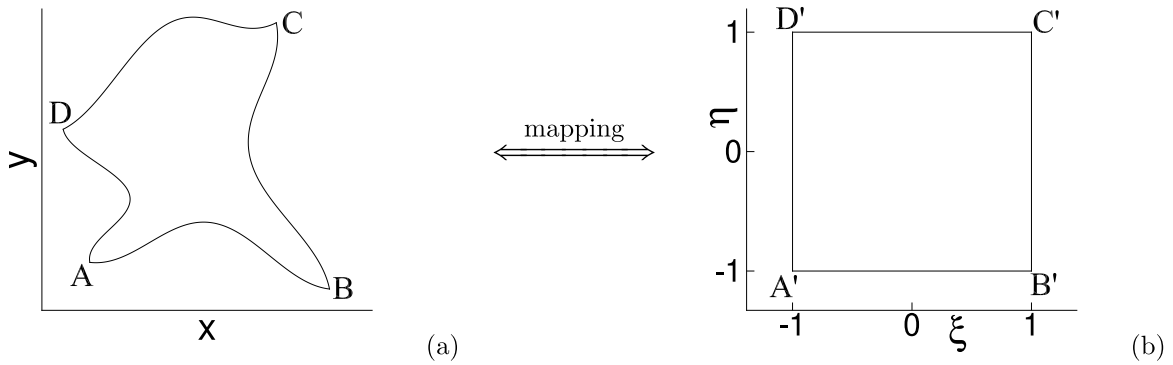


Fig. 1. Mapping between a general quadrilateral domain  $\Omega$  and the standard quadrilateral domain  $\Omega_{st} = [-1, 1]^2$ .

## 2. Exact enforcement of boundary conditions on general quadrilateral domains with curved boundaries

### 2.1. Mapping general quadrilateral domains with curved boundaries

We consider a general quadrilateral (Quad) domain  $\Omega = \overline{ABCD}$  as sketched in Fig. 1(a), whose boundaries can each be an arbitrary curve. To represent a field function  $u(\mathbf{x})$ ,  $\mathbf{x} = (x, y) \in \Omega$ , defined on this domain that exactly satisfy the prescribed boundary conditions, it is necessary to first consider the mapping of this domain to the standard quadrilateral domain,

$$(x, y) = \mathbf{x}(\xi) = \mathbf{x}(\xi, \eta) = (x(\xi, \eta), y(\xi, \eta)), \quad (1)$$

where  $\xi = (\xi, \eta) \in \Omega_{st}$ , and  $\Omega_{st} = \overline{A'B'C'D'} = [-1, 1] \times [-1, 1]$  denotes the standard domain (see Fig. 1(b)).

We assume that the map  $\mathbf{x}(\xi, \eta) = (x(\xi, \eta), y(\xi, \eta))$  represents a regular transformation [91] (i.e. of class at least  $C^1$ , univalent, with non-singular Jacobian matrix). It needs to satisfy the following Dirichlet boundary conditions,

$$\begin{cases} \mathbf{x}(-1, \eta) = \mathbf{x}_{AD}(\eta), & \eta \in [-1, 1], \\ \mathbf{x}(1, \eta) = \mathbf{x}_{BC}(\eta), & \eta \in [-1, 1], \\ \mathbf{x}(\xi, -1) = \mathbf{x}_{AB}(\xi), & \xi \in [-1, 1], \\ \mathbf{x}(\xi, 1) = \mathbf{x}_{CD}(\xi), & \xi \in [-1, 1], \end{cases} \quad (2)$$

where  $\mathbf{x}_{AB}(\xi)$ ,  $\mathbf{x}_{BC}(\eta)$ ,  $\mathbf{x}_{CD}(\xi)$ , and  $\mathbf{x}_{AD}(\eta)$  are the prescribed boundary curves for  $\overline{AB}$ ,  $\overline{BC}$ ,  $\overline{CD}$  and  $\overline{AD}$  in parametric forms. The prescribed forms should be compatible at the vertices,

$$\begin{cases} \mathbf{x}_{AB}(-1) = \mathbf{x}_{AD}(-1) = \mathbf{x}_A, & \mathbf{x}_{AB}(1) = \mathbf{x}_{BC}(-1) = \mathbf{x}_B, \\ \mathbf{x}_{BC}(1) = \mathbf{x}_{CD}(1) = \mathbf{x}_C, & \mathbf{x}_{CD}(-1) = \mathbf{x}_{AD}(1) = \mathbf{x}_D, \end{cases} \quad (3)$$

where  $\mathbf{x}_A$ ,  $\mathbf{x}_B$ ,  $\mathbf{x}_C$  and  $\mathbf{x}_D$  denote the coordinates of the four vertices  $A$ ,  $B$ ,  $C$  and  $D$ , respectively.

The general form of a vector-valued function that satisfies the conditions in (2) is given by the TFC constrained expression,

$$\mathbf{x}(\xi, \eta) = \mathbf{g}(\xi, \eta) - P\mathbf{g}(\xi, \eta) + P\mathbf{X}(\xi, \eta). \quad (4)$$

Here  $\mathbf{g}(\xi, \eta) \in \mathbb{R}^2$  is an arbitrary or free function (with sufficient regularity),  $P$  is a 2D projection operator and  $P\mathbf{X}(\xi, \eta)$  denotes the 2D transfinite interpolation [71] of  $\mathbf{X}(\xi, \eta)$ , where  $\mathbf{X}(\xi, \eta)$  is defined on the boundaries of  $\Omega_{st}$  by

$$\mathbf{X}(-1, \eta) = \mathbf{x}_{AD}(\eta), \quad \mathbf{X}(1, \eta) = \mathbf{x}_{BC}(\eta), \quad \mathbf{X}(\xi, -1) = \mathbf{x}_{AB}(\xi), \quad \mathbf{X}(\xi, 1) = \mathbf{x}_{CD}(\xi). \quad (5)$$

$P$  is given by the boolean sum of two 1D transfinite interpolation operators  $P_1$  and  $P_2$  (see [71] for details), for  $\xi$  and  $\eta$  directions respectively, defined by

$$\begin{cases} P_1\mathbf{X}(\xi, \eta) = \mathbf{X}(-1, \eta)\phi_0(\xi) + \mathbf{X}(1, \eta)\phi_1(\xi), \\ P_2\mathbf{X}(\xi, \eta) = \mathbf{X}(\xi, -1)\phi_0(\eta) + \mathbf{X}(\xi, 1)\phi_1(\eta), \end{cases} \quad (6)$$

where  $\phi_0(\xi)$  and  $\phi_1(\xi)$  are the so-called blending functions [71], also termed switching functions in TFC, on  $\xi \in [-1, 1]$  as given by

$$\phi_0(\xi) = \frac{1}{2}(1 - \xi), \quad \phi_1(\xi) = \frac{1}{2}(1 + \xi). \quad (7)$$

$P\mathbf{X}(\xi, \eta)$  is specifically given by,

$$\begin{aligned} P\mathbf{X}(\xi, \eta) &= (P_1 \oplus P_2)\mathbf{X}(\xi, \eta) = P_1\mathbf{X}(\xi, \eta) + P_2\mathbf{X}(\xi, \eta) - P_1P_2\mathbf{X}(\xi, \eta) \\ &= \mathbf{X}(-1, \eta)\phi_0(\xi) + \mathbf{X}(1, \eta)\phi_1(\xi) + \mathbf{X}(\xi, -1)\phi_0(\eta) + \mathbf{X}(\xi, 1)\phi_1(\eta) \\ &\quad - [\mathbf{X}(-1, -1)\phi_0(\eta) + \mathbf{X}(-1, 1)\phi_1(\eta)]\phi_0(\xi) - [\mathbf{X}(1, -1)\phi_0(\eta) + \mathbf{X}(1, 1)\phi_1(\eta)]\phi_1(\xi), \end{aligned} \quad (8)$$

where  $\oplus$  denotes the boolean sum.  $Pg(\xi, \eta)$  is defined in a fashion analogous to Eq. (8).

The map given by (4), for arbitrary  $g(\xi, \eta)$ , satisfies the boundary conditions in (2) exactly. For most numerical simulations in this paper, we employ simply  $g = 0$  in (4), leading to the map

$$\mathbf{x}(\xi, \eta) = P\mathbf{X}(\xi, \eta). \quad (9)$$

When the boundary curves are more complicated or highly distorted, other choices for  $g(\xi, \eta)$  in the mapping can be more favorable. These choices will be specified in the numerical simulations in later sections.

In practice, it is extremely difficult to ensure analytically the univalence of the map in (4) or (9). In this work we employ the strategy as advocated in [72] for constructing univalent mappings, by combining visualization with numerical simulation. By visualizing representative grid lines in the domain, one can effectively detect abnormalities in the domain mapping, such as the “overspill” or the intersection of generalized grid lines corresponding to different values of the same variable. These anomalies can then be remedied via measures such as boundary curve re-parameterization or incorporating auxiliary constraints into the mapping.

**Remark 2.1.** The requirement for non-singularity in the Jacobian matrix for the mapping function (1) can be relaxed on those vertices where two intersecting boundaries involve either (i) both Dirichlet BCs, or (ii) one Dirichlet BC and one Neumann BC, or (iii) one Dirichlet BC and one Robin BC. In this work we allow these types of boundary pairs to intersect at a vertex smoothly, i.e. with the same tangent line, leading to a singular Jacobian matrix at that vertex.

## 2.2. Enforcing Dirichlet boundary conditions

We next systematically develop formulations for field functions defined on the general quadrilateral domain as in Fig. 1(a) that exactly satisfy the prescribed conditions on its boundaries. We first consider Dirichlet boundary conditions (DBC). Specifically, we seek a scalar field function  $u(\mathbf{x})$  ( $\mathbf{x} \in \Omega = \overline{ABCD}$ ) satisfying the following conditions,

$$u(\mathbf{x})|_{\mathbf{x} \in \overline{AB}} = u_{AB}(\mathbf{x}), \quad u(\mathbf{x})|_{\mathbf{x} \in \overline{BC}} = u_{BC}(\mathbf{x}), \quad u(\mathbf{x})|_{\mathbf{x} \in \overline{CD}} = u_{CD}(\mathbf{x}), \quad u(\mathbf{x})|_{\mathbf{x} \in \overline{AD}} = u_{AD}(\mathbf{x}), \quad (10)$$

where  $u_{AB}(\mathbf{x})$ ,  $u_{BC}(\mathbf{x})$ ,  $u_{CD}(\mathbf{x})$ , and  $u_{AD}(\mathbf{x})$  are prescribed Dirichlet data on the boundaries. These boundary distributions should be compatible on the vertices,

$$u_{AB}(\mathbf{x}_A) = u_{AD}(\mathbf{x}_A) = u_A, \quad (11a)$$

$$u_{AB}(\mathbf{x}_B) = u_{BC}(\mathbf{x}_B) = u_B, \quad (11b)$$

$$u_{BC}(\mathbf{x}_C) = u_{CD}(\mathbf{x}_C) = u_C, \quad (11c)$$

$$u_{CD}(\mathbf{x}_D) = u_{AD}(\mathbf{x}_D) = u_D, \quad (11d)$$

where  $\mathbf{x}_A$ ,  $\mathbf{x}_B$ ,  $\mathbf{x}_C$  and  $\mathbf{x}_D$  are the vertex coordinates, and  $u_A$ ,  $u_B$ ,  $u_C$  and  $u_D$  are their function values. We aim to formulate the general form of  $u(\mathbf{x})$  that satisfies the DBCs in (10) exactly.

Employing the mapping  $\mathbf{x}(\xi, \eta)$  from Section 2.1, we transform the field function into,

$$u(\mathbf{x}) = u(\mathbf{x}(\xi, \eta)) = V(\xi, \eta), \quad (12)$$

and the boundary distributions in (10) into,

$$u_{AB}(\mathbf{x}) = u_{AB}(\mathbf{x}(\xi, -1)) = F(\xi, -1), \quad \xi \in [-1, 1]; \quad (13a)$$

$$u_{BC}(\mathbf{x}) = u_{BC}(\mathbf{x}(1, \eta)) = F(1, \eta), \quad \eta \in [-1, 1]; \quad (13b)$$

$$u_{CD}(\mathbf{x}) = u_{CD}(\mathbf{x}(\xi, 1)) = F(\xi, 1), \quad \xi \in [-1, 1]; \quad (13c)$$

$$u_{AD}(\mathbf{x}) = u_{AD}(\mathbf{x}(-1, \eta)) = F(-1, \eta), \quad \eta \in [-1, 1]. \quad (13d)$$

The problem of seeking  $u(\mathbf{x})$  is then transformed into the following. Find  $V(\xi, \eta)$ , for  $(\xi, \eta) \in \Omega_{st}$ , such that

$$V(\xi, -1) = F(\xi, -1), \quad \xi \in [-1, 1]; \quad (14a)$$

$$V(1, \eta) = F(1, \eta), \quad \eta \in [-1, 1]; \quad (14b)$$

$$V(\xi, 1) = F(\xi, 1), \quad \xi \in [-1, 1]; \quad (14c)$$

$$V(-1, \eta) = F(-1, \eta), \quad \eta \in [-1, 1], \quad (14d)$$

where  $F$  denotes the boundary distributions given in (13).

The general form of  $V(\xi, \eta)$  that satisfies the conditions in (14) is given by the following TFC constrained expression,

$$V(\xi, \eta) = g(\xi, \eta) - Pg(\xi, \eta) + PF(\xi, \eta), \quad (\xi, \eta) \in \Omega_{st}, \quad (15)$$

where  $g(\xi, \eta)$  is a free (arbitrary) function, and  $P$  is the transfinite interpolation operator defined in (8). More specifically,

$$Pg(\xi, \eta) = g(-1, \eta)\phi_0(\xi) + g(1, \eta)\phi_1(\xi) + g(\xi, -1)\phi_0(\eta) + g(\xi, 1)\phi_1(\eta)$$

$$- [g(-1, -1)\phi_0(\eta) + g(-1, 1)\phi_1(\eta)] \phi_0(\xi) - [g(1, -1)\phi_0(\eta) + g(1, 1)\phi_1(\eta)] \phi_1(\xi); \quad (16a)$$

$$PF(\xi, \eta) = F(-1, \eta)\phi_0(\xi) + F(1, \eta)\phi_1(\xi) + F(\xi, -1)\phi_0(\eta) + F(\xi, 1)\phi_1(\eta) \\ - [F(-1, -1)\phi_0(\eta) + F(-1, 1)\phi_1(\eta)] \phi_0(\xi) - [F(1, -1)\phi_0(\eta) + F(1, 1)\phi_1(\eta)] \phi_1(\xi). \quad (16b)$$

It is straightforward to verify that, for arbitrary  $g(\xi, \eta)$ , the function  $V(\xi, \eta)$  given by (15) satisfies the boundary conditions in (14) exactly.

Therefore, employing the parametric form (15) for  $u(\mathbf{x})$ , one can satisfy the DBCs in (10) exactly on the general quadrilateral domain  $\Omega$ . If  $u(\mathbf{x})$  represents the unknown solution field to a given PDE, one can restrict the free function  $g(\xi, \eta)$  in (15) to an appropriate function space or represent it by an artificial neural network (NN), and then determine the expansion coefficients or the NN trainable parameters based on the given PDE. We will discuss how to combine the extreme learning machine (ELM) method and the formulations developed here for solving PDEs on general quadrilateral domains later in Section 2.5.

### 2.3. Enforcing Neumann boundary conditions

We next consider how to enforce Neumann boundary conditions (NBCs) exactly on the general quadrilateral domain  $\Omega$ . Since the boundary condition involves function derivatives, the formulation for its exact enforcement becomes much more intricate.

We consider a combination of Dirichlet and Neumann BCs for the domain boundaries, and assume that the domain involves at least one Neumann boundary and one Dirichlet boundary, with the rest being either Dirichlet or Neumann types. We distinguish two cases: (i) when the Neumann boundary only intersects with Dirichlet boundaries (i.e. no two Neumann boundaries intersect), and (ii) when two Neumann boundaries intersect with each other. The formulations for the exact enforcement of NBCs/DBC of these cases are developed below separately.

#### 2.3.1. When Neumann boundary only intersects with Dirichlet boundary

This case occurs when the domain involves one Neumann boundary and three Dirichlet boundaries, or when two Neumann conditions are imposed on opposite sides of the domain. In the discussions below we assume that the domain involves a single Neumann boundary, and consider the exact enforcement of NBC/DBC. The formulation presented below, with some modification that involves no essential difficulties, can be used to enforce two Neumann conditions imposed on opposite boundaries of the quadrilateral domain.

Let us assume, without loss of generality, that the single Neumann condition is imposed on the boundary  $\overline{BC}$ . Specifically, we seek a scalar field function  $u(\mathbf{x})$ , for  $\mathbf{x} \in \Omega$ , which satisfies the Neumann condition on  $\overline{BC}$  and Dirichlet conditions on the other boundaries,

$$u|_{\mathbf{x} \in \overline{AB}} = u_{AB}(\mathbf{x}), \quad (17a)$$

$$\mathbf{n} \cdot \nabla u|_{\mathbf{x} \in \overline{BC}} = u_{nBC}(\mathbf{x}), \quad (17b)$$

$$u|_{\mathbf{x} \in \overline{CD}} = u_{CD}(\mathbf{x}), \quad (17c)$$

$$u|_{\mathbf{x} \in \overline{AD}} = u_{AD}(\mathbf{x}), \quad (17d)$$

where  $\mathbf{n}$  denotes the outward-pointing unit normal vector, and  $u_{nBC}(\mathbf{x})$  is the prescribed Neumann boundary distribution on  $\overline{BC}$ . The prescribed Dirichlet boundary functions  $u_{AB}(\mathbf{x})$ ,  $u_{AD}(\mathbf{x})$  and  $u_{CD}(\mathbf{x})$  must be compatible at the vertices  $A$  and  $D$  (see (11a) and (11d)). They must also be compatible with the Neumann boundary function  $u_{nBC}(\mathbf{x})$  at the vertices  $B$  and  $C$ , which will be elaborated below.

In the following development we assume that the boundaries  $\overline{BC}$  and  $\overline{AB}$  intersect at an angle at vertex  $B$  (i.e. no common tangent at  $B$ ), and that at vertex  $C$  the boundaries  $\overline{BC}$  and  $\overline{CD}$  also intersect at an angle, thus leading to a nonsingular Jacobian matrix of the map  $\mathbf{x}(\xi, \eta)$  at both vertices. We will discuss how to handle a smooth domain boundary at vertices  $B$  or  $C$ , i.e. with a common tangent at those locations (singular Jacobian matrix for the mapping), in a remark at the end of this section.

Employing the mapping function  $\mathbf{x}(\xi, \eta)$ , the function  $u(\mathbf{x})$  and the Dirichlet boundary functions in (17a) and (17c)–(17d) are transformed into (12), (13a) and (13c)–(13d), respectively. The Neumann condition (17b) is transformed into,

$$V_\xi(1, \eta) + S_{BC}(\eta)V_\eta(1, \eta) = T_{BC}(\eta), \quad \eta \in [-1, 1], \quad (18a)$$

$$\text{or } V_\xi(1, \eta) = T_{BC}(\eta) - S_{BC}(\eta)V_\eta(1, \eta) = F_\xi(1, \eta), \quad (18b)$$

where

$$\begin{cases} S_{BC}(\eta) = \frac{K_{yBC}(\eta)}{K_{xBC}(\eta)}, & T_{BC}(\eta) = \frac{F_{nBC}(\eta)}{K_{xBC}(\eta)} = \frac{u_{nBC}(\mathbf{x}(1, \eta))}{K_{xBC}(\eta)}, \\ \mathbf{K}_{BC}(\eta) = \begin{bmatrix} K_{xBC}(\eta) \\ K_{yBC}(\eta) \end{bmatrix} = \mathbf{J}^{-1}(1, \eta) \begin{bmatrix} n_{xBC}(\eta) \\ n_{yBC}(\eta) \end{bmatrix} = \frac{1}{\det \mathbf{J}(1, \eta)} \begin{bmatrix} \|\mathbf{x}_\eta(1, \eta)\| \\ -\frac{\mathbf{x}_\xi(1, \eta) \cdot \mathbf{x}_\eta(1, \eta)}{\|\mathbf{x}_\eta(1, \eta)\|} \end{bmatrix}, \\ \mathbf{J}(\xi, \eta) = \begin{bmatrix} x_\xi(\xi, \eta) & x_\eta(\xi, \eta) \\ y_\xi(\xi, \eta) & y_\eta(\xi, \eta) \end{bmatrix}. \end{cases} \quad (19)$$

**Table 1**Interpolation properties of  $C^1$  Hermite interpolation polynomials defined on  $\xi \in [-1, 1]$ .

| $\xi$ | $\varphi_0(\xi)$ | $\varphi'_0(\xi)$ | $\varphi_1(\xi)$ | $\varphi'_1(\xi)$ | $\psi_0(\xi)$ | $\psi'_0(\xi)$ | $\psi_1(\xi)$ | $\psi'_1(\xi)$ |
|-------|------------------|-------------------|------------------|-------------------|---------------|----------------|---------------|----------------|
| -1    | 1                | 0                 | 0                | 0                 | 0             | 1              | 0             | 0              |
| 1     | 0                | 0                 | 1                | 0                 | 0             | 0              | 0             | 1              |

In the above equations  $\mathbf{J}(\xi, \eta)$  is the Jacobian matrix of the map  $\mathbf{x}(\xi, \eta)$ , and we have used the relations

$$\begin{cases} \mathbf{n} \cdot \nabla u = [u_x & u_y] \begin{bmatrix} n_x \\ n_y \end{bmatrix} = [V_\xi & V_\eta] \mathbf{J}^{-1}(\xi, \eta) \begin{bmatrix} n_x \\ n_y \end{bmatrix} = [V_\xi & V_\eta] \begin{bmatrix} K_x \\ K_y \end{bmatrix}; \\ \begin{bmatrix} n_x \\ n_y \end{bmatrix} = \mathbf{n} = \sigma \boldsymbol{\tau} = \sigma \begin{bmatrix} \tau_x \\ \tau_y \end{bmatrix}; \quad \sigma = \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix} \text{ on } \overline{AB} \text{ or } \overline{BC} \text{ and } \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix} \text{ on } \overline{CD} \text{ or } \overline{AD}, \end{cases} \quad (20)$$

in which  $\boldsymbol{\tau} = (\tau_x, \tau_y)$  is the unit tangent vector of the domain boundary.

The requirement for continuity of  $V(\xi, \eta)$  and  $V_\xi(\xi, \eta)$  at the vertices  $B$  and  $C$  leads to,

$$V(1, -1) = F(1, -1) = \lim_{\xi \rightarrow 1} F(\xi, -1), \quad (21a)$$

$$V(1, 1) = F(1, 1) = \lim_{\xi \rightarrow 1} F(\xi, 1), \quad (21b)$$

$$V_\xi(1, -1) = F_\xi(1, -1) = \lim_{\xi \rightarrow 1} F_\xi(\xi, -1), \quad (21c)$$

$$V_\xi(1, 1) = F_\xi(1, 1) = \lim_{\xi \rightarrow 1} F_\xi(\xi, 1), \quad (21d)$$

where  $F_\xi(\xi, -1)$  and  $F_\xi(\xi, 1)$  can be computed from the Dirichlet boundary functions on  $\overline{AB}$  and  $\overline{CD}$  (see (13a) and (13c)). Therefore, the compatibility between (18) and (14a) at vertex  $B$  leads to,

$$V_\eta(1, -1) = \frac{1}{S_{BC}(-1)} [T_{BC}(-1) - F_\xi(1, -1)] = F_\eta^a(1, -1), \quad \text{if } \overline{AB} \not\perp \overline{BC} \text{ at } B, \quad (22a)$$

$$F_\xi(1, -1) - T_{BC}(-1) = 0, \quad \text{if } \overline{AB} \perp \overline{BC} \text{ at } B, \quad (22b)$$

where we have used (21c). The compatibility between (18) and (14c) at vertex  $C$  leads to,

$$V_\eta(1, 1) = \frac{1}{S_{BC}(1)} [T_{BC}(1) - F_\xi(1, 1)] = F_\eta^a(1, 1), \quad \text{if } \overline{CD} \not\perp \overline{BC} \text{ at } C, \quad (23a)$$

$$F_\xi(1, 1) - T_{BC}(1) = 0, \quad \text{if } \overline{CD} \perp \overline{BC} \text{ at } C. \quad (23b)$$

The conditions (22b) and (23b) are constraints on the prescribed Dirichlet and Neumann boundary functions when the boundaries are orthogonal at these vertices, which we assume will always be satisfied by the prescribed data. The Eqs. (22a) and (23a) are constraints on  $V_\eta(1, -1)$  and  $V_\eta(1, 1)$  for the unknown field function  $V(\xi, \eta)$ , imposed only when the boundary curves are not orthogonal at vertices  $B$  or  $C$ .

Our objective is to develop general forms of  $V(\xi, \eta)$  that exactly satisfy the conditions (18), (21a)–(21b), (22a), (23a), as well as (14a) and (14c)–(14d). The primary challenge here is caused by (18), in which the unknowns  $V_\xi(1, \eta)$  and  $V_\eta(1, \eta)$  are coupled together. We will use (18b) for enforcing this condition, by treating it as a constraint on  $V_\xi(1, \eta)$ , where the imposed data  $F_\xi(1, \eta)$  contains the unknown  $V_\eta(1, \eta)$ .

We will follow a four-step procedure to formulate the general form of  $V(\xi, \eta)$  that satisfies the above conditions:

- (step #1) Identify the set of variables on which the boundary conditions, and the compatibility constraints induced by these boundary conditions, are imposed.
- (step #2) Construct a transfinite interpolation for the types of identified variables, including the induced forms of constraints.
- (step #3) Formulate a preliminary TFC constrained expression based on this transfinite interpolation, thus giving rise to a preliminary form for  $V(\xi, \eta)$  with a free function.
- (step #4) Update the terms in the transfinite interpolant that involve the unknown function  $V(\xi, \eta)$ , by replacing those  $V$  terms with corresponding terms that result from the preliminary TFC form of the previous step. The updated TFC constrained expression provides the final form for  $V(\xi, \eta)$ .

To facilitate the subsequent discussions, let us recall the  $C^1$  Hermite interpolation polynomials  $\varphi_0(\xi)$ ,  $\varphi_1(\xi)$ ,  $\psi_0(\xi)$  and  $\psi_1(\xi)$  defined on  $\xi \in [-1, 1]$ , which satisfy the interpolation properties listed in Table 1. These polynomials are given by

$$\begin{aligned} \varphi_0(\xi) &= \phi_0^2(\xi) [1 + 2\phi_1(\xi)], & \varphi_1(\xi) &= \phi_1^2(\xi) [1 + 2\phi_0(\xi)], \\ \psi_0(\xi) &= 2\phi_0^2(\xi)\phi_1(\xi), & \psi_1(\xi) &= -2\phi_0(\xi)\phi_1^2(\xi), \end{aligned} \quad (24)$$

where  $\phi_0(\xi)$  and  $\phi_1(\xi)$  are defined in (7). We define two constants  $\lambda_B$  and  $\lambda_C$  as flags on whether the boundary curves are orthogonal at the vertices  $B$  and  $C$ ,

$$\lambda_B = \begin{cases} 0, & \text{if } \overline{AB} \perp \overline{BC} \text{ at } B, \\ 1, & \text{if } \overline{AB} \not\perp \overline{BC} \text{ at } B; \end{cases} \quad \lambda_C = \begin{cases} 0, & \text{if } \overline{CD} \perp \overline{BC} \text{ at } C, \\ 1, & \text{if } \overline{CD} \not\perp \overline{BC} \text{ at } C. \end{cases} \quad (25)$$

Following the aforementioned procedure, we first identify the set of variables on which the conditions (or induced constraints) are imposed (step #1). The conditions for (18b), (14a), and (14c)–(14d) are very clear. While the conditions (22a), (23a), and also (21a)–(21b) are apparently about the function values or derivatives on the vertices  $B$  and  $C$ , they are actually constraints on the boundary distribution  $V(1, \eta)$ .  $V(1, \eta)$  is not involved in the original boundary conditions, but should ensure that these conditions for the vertices be satisfied. Employing the  $C^1$  Hermite interpolation polynomials, the following distribution for  $\overline{BC}$  satisfies the conditions (22a), (23a), and (21a)–(21b),

$$F(1, \eta) = F(1, -1)\varphi_0(\eta) + F(1, 1)\varphi_1(\eta) + \lambda_B F_\eta(1, -1)\psi_0(\eta) + \lambda_C F_\eta(1, 1)\psi_1(\eta), \quad (26)$$

Here,

$$F_\eta(1, -1) = F_\eta^a(1, -1), \quad F_\eta(1, 1) = F_\eta^a(1, 1), \quad (27)$$

with  $F_\eta^a(1, -1)$  and  $F_\eta^a(1, 1)$  defined in (22a) and (23a). The constants  $\lambda_B$  and  $\lambda_C$  ensure that the conditions (22a) and (23a) are imposed only when the boundary curves are not orthogonal at  $B$  or  $C$ . The distribution (26) is compatible with the Dirichlet boundary functions for  $\overline{AB}$  and  $\overline{CD}$  at these vertices. Our task is then reduced to the following: find  $V(\xi, \eta)$  such that

$$V(-1, \eta) = F(-1, \eta), \quad (28a)$$

$$V(1, \eta) = F(1, \eta) = F(1, -1)\varphi_0(\eta) + F(1, 1)\varphi_1(\eta) + \lambda_B F_\eta(1, -1)\psi_0(\eta) + \lambda_C F_\eta(1, 1)\psi_1(\eta), \quad (28b)$$

$$V_\xi(1, \eta) = F_\xi(1, \eta), \quad (28c)$$

$$V(\xi, -1) = F(\xi, -1), \quad (28d)$$

$$V(\xi, 1) = F(\xi, 1), \quad (28e)$$

where Eq. (26) has been used, the functions  $F(-1, \eta)$ ,  $F(\xi, -1)$  and  $F(\xi, 1)$  are known and given in (13a), (13c)–(13d), and  $F_\xi(1, \eta)$  is defined in (18b), which contains the unknown  $V_\eta(1, \eta)$ .

Next, we construct a transfinite interpolation for the conditions in (28) (step #2). Let  $P_1 F$  and  $P_2 F$  denote two 1D transfinite interpolations along the  $\xi$  and  $\eta$  directions, respectively defined by,

$$P_1 F(\xi, \eta) = F(-1, \eta)\varphi_0(\xi) + F(1, \eta)\varphi_1(\xi) + F_\xi(1, \eta)\psi_1(\xi), \quad (29a)$$

$$P_2 F(\xi, \eta) = F(\xi, -1)\varphi_0(\eta) + F(\xi, 1)\varphi_1(\eta). \quad (29b)$$

Let  $PF$  denote the boolean sum of  $P_1 F$  and  $P_2 F$ , given by,

$$\begin{aligned} PF(\xi, \eta) &= (P_1 \oplus P_2)F(\xi, \eta) = P_1 F(\xi, \eta) + P_2 F(\xi, \eta) - P_1 P_2 F(\xi, \eta) \\ &= F(-1, \eta)\varphi_0(\xi) + F_\xi(1, \eta)\psi_1(\xi) + F(\xi, -1)\varphi_0(\eta) + F(\xi, 1)\varphi_1(\eta) \\ &\quad - [F(-1, -1)\varphi_0(\eta) + F(-1, 1)\varphi_1(\eta)] \varphi_0(\xi) - [F_\xi(1, -1)\varphi_0(\eta) + F_\xi(1, 1)\varphi_1(\eta)] \psi_1(\xi) \\ &\quad + [\lambda_B F_\eta(1, -1)\psi_0(\eta) + \lambda_C F_\eta(1, 1)\psi_1(\eta)] \varphi_1(\xi) \end{aligned} \quad (30)$$

where we have used (26). It is straightforward to verify that  $V = PF(\xi, \eta)$  satisfies the conditions in (28a), (28c)–(28e), (22a), (21a) and (23a)–(21b), by noting (27).

We can now formulate a preliminary general form of  $V(\xi, \eta)$  for the conditions (28) using the TFC constrained expression (step #3),

$$V(\xi, \eta) = g(\xi, \eta) - Pg(\xi, \eta) + PF(\xi, \eta), \quad (31)$$

where  $g(\xi, \eta)$  is a free (arbitrary) function,  $PF$  is defined by (30), and  $Pg(\xi, \eta)$  is defined analogously and specifically given by

$$\begin{aligned} Pg(\xi, \eta) &= g(-1, \eta)\varphi_0(\xi) + g_\xi(1, \eta)\psi_1(\xi) + g(\xi, -1)\varphi_0(\eta) + g(\xi, 1)\varphi_1(\eta) \\ &\quad - [g(-1, -1)\varphi_0(\eta) + g(-1, 1)\varphi_1(\eta)] \varphi_0(\xi) - [g_\xi(1, -1)\varphi_0(\eta) + g_\xi(1, 1)\varphi_1(\eta)] \psi_1(\xi) \\ &\quad + [\lambda_B g_\eta(1, -1)\psi_0(\eta) + \lambda_C g_\eta(1, 1)\psi_1(\eta)] \varphi_1(\xi). \end{aligned} \quad (32)$$

For any  $g(\xi, \eta)$ , the  $V(\xi, \eta)$  given by (31) satisfies (28a), (28c)–(28e), (22a), (21a) and (23a)–(21b), by noting (27). In light of (31),  $V_\eta(\xi, \eta)$  on  $\overline{BC}$  is reduced to,

$$\begin{aligned} V_\eta(1, \eta) &= g_\eta(1, \eta) - [g(1, -1) - F(1, -1)] \varphi'_0(\eta) - [g(1, 1) - F(1, 1)] \varphi'_1(\eta) \\ &\quad - \lambda_B [g_\eta(1, -1) - F_\eta(1, -1)] \psi'_0(\eta) - \lambda_C [g_\eta(1, 1) - F_\eta(1, 1)] \psi'_1(\eta). \end{aligned} \quad (33)$$

Finally, we update the terms in the transfinite interpolant  $PF(\xi, \eta)$  that involve the unknown function  $V(\xi, \eta)$  (step #4). In (30) and (33), we update  $F_\eta(1, -1)$  and  $F_\eta(1, 1)$  using (27).  $F_\xi(1, \eta)$  in (30) is given by (18b), in which we replace  $V_\eta(1, \eta)$  by the expression (33). We define the updated terms,

$$F_\xi^g(1, \eta) = T_{BC}(\eta) - S_{BC}(\eta) \left\{ g_\eta(1, \eta) - [g(1, -1) - F(1, -1)] \varphi'_0(\eta) - [g(1, 1) - F(1, 1)] \varphi'_1(\eta) - \lambda_B [g_\eta(1, -1) - F_\eta^a(1, -1)] \psi'_0(\eta) - \lambda_C [g_\eta(1, 1) - F_\eta^a(1, 1)] \psi'_1(\eta) \right\}, \quad (34a)$$

$$PF^g(\xi, \eta) = F(-1, \eta) \varphi_0(\xi) + F_\xi^g(1, \eta) \psi_1(\xi) + F(\xi, -1) \varphi_0(\eta) + F(\xi, 1) \varphi_1(\eta) - [F(-1, -1) \varphi_0(\eta) + F(-1, 1) \varphi_1(\eta)] \varphi_0(\xi) - [F_\xi(1, -1) \varphi_0(\eta) + F_\xi(1, 1) \varphi_1(\eta)] \psi_1(\xi) + [\lambda_B F_\eta^a(1, -1) \psi_0(\eta) + \lambda_C F_\eta^a(1, 1) \psi_1(\eta)] \varphi_1(\xi). \quad (34b)$$

The final form for  $V(\xi, \eta)$  is then given by

$$V(\xi, \eta) = g(\xi, \eta) - Pg(\xi, \eta) + PF^g(\xi, \eta). \quad (35)$$

Here  $g(\xi, \eta)$  is the free function,  $Pg(\xi, \eta)$  is given by (32), and  $PF^g(\xi, \eta)$  is defined by (34b).  $F(-1, \eta)$ ,  $F(\xi, -1)$  and  $F(\xi, 1)$  are given by (13a) and (13c)–(13d).  $F(-1, -1) = u_A$  and  $F(-1, 1) = u_D$  in light of (13a), (13d), (11a) and (11d).  $F(1, -1) = \lim_{\xi \rightarrow 1} F(\xi, -1) = u_B$  and  $F(1, 1) = \lim_{\xi \rightarrow 1} F(\xi, 1) = u_C$  in light of (13a) and (13c).  $F_\xi(1, -1) = \lim_{\xi \rightarrow 1} F_\xi(\xi, -1)$  and  $F_\xi(1, 1) = \lim_{\xi \rightarrow 1} F_\xi(\xi, 1)$ .  $F_\eta^a(1, -1)$  and  $F_\eta^a(1, 1)$  are given by (22a) and (23a), and  $\lambda_B$  and  $\lambda_C$  are defined in (25).  $S_{BC}(\eta)$  and  $T_{BC}(\eta)$  are defined in (19).

**Theorem 2.1.** The form  $V(\xi, \eta)$  given by (35) satisfies the conditions (28a), (28d), (28e) and (18a), for any  $g(\xi, \eta)$  therein that is sufficiently differentiable.

**Proof.** We only verify the Neumann condition (18a) here. The verification of the other conditions is straightforward. From (35), we have

$$\begin{aligned} V_\xi(1, \eta) &= F_\xi^g(1, \eta), \\ V_\eta(1, \eta) &= g_\eta(1, \eta) - [g(1, -1) - F(1, -1)] \varphi'_0(\eta) - [g(1, 1) - F(1, 1)] \varphi'_1(\eta) \\ &\quad - \lambda_B [g_\eta(1, -1) - F_\eta^a(1, -1)] \psi'_0(\eta) - \lambda_C [g_\eta(1, 1) - F_\eta^a(1, 1)] \psi'_1(\eta). \end{aligned}$$

In light of (34a), we conclude that (18a) holds for any  $g(\xi, \eta)$ .  $\square$

**Remark 2.2.** The use of  $C^1$  Hermite interpolation polynomials  $\varphi_0$ ,  $\varphi_1$ ,  $\psi_0$  and  $\psi_1$  in the construction of  $V(\xi, \eta)$  is crucial, which enables one to de-couple  $V_\xi(1, \eta)$  and  $V_\eta(1, \eta)$  when handling the Neumann condition (18).

**Remark 2.3.** If the boundaries  $\overline{AB}$  and  $\overline{BC}$  connect smoothly at vertex  $B$ , i.e. having a common tangent, the Jacobian matrix of the map  $\mathbf{x}(\xi, \eta)$  will be singular at  $B$ . Similarly, if the boundary curve is smooth at vertex  $C$  the Jacobian matrix will be singular there. Let us suppose the boundary is smooth at both  $B$  and  $C$ , and we next comment on how to handle this situation. In this case the form (18) for the Neumann condition holds only for  $\eta \in (-1, 1)$ . At these vertices the formulas (22a) and (23a) for computing  $F_\eta^a(1, -1)$  and  $F_\eta^a(1, 1)$  are no longer valid. However,  $F_\eta(1, -1)$  and  $F_\eta(1, 1)$  can still be determined, based on the existence of a common tangent at these vertices. Specifically, let

$$\boldsymbol{\tau}_{BC}(\eta) = \frac{\mathbf{x}_\eta(1, \eta)}{\|\mathbf{x}_\eta(1, \eta)\|}, \quad \boldsymbol{\tau}_{AB}(\xi) = \frac{\mathbf{x}_\xi(\xi, -1)}{\|\mathbf{x}_\xi(\xi, -1)\|}, \quad \boldsymbol{\tau}_{CD}(\xi) = \frac{\mathbf{x}_\xi(\xi, 1)}{\|\mathbf{x}_\xi(\xi, 1)\|}, \quad (36)$$

denote the unit tangent vectors on  $\overline{BC}$ ,  $\overline{AB}$  and  $\overline{CD}$ . The existence of a common tangent at  $B$  and  $C$  implies that

$$\boldsymbol{\tau}_{BC}(-1) = \boldsymbol{\tau}_{AB}(1), \quad \boldsymbol{\tau}_{BC}(1) = -\boldsymbol{\tau}_{CD}(1). \quad (37)$$

This leads to the relations

$$\frac{\mathbf{x}_\eta(1, -1)}{\|\mathbf{x}_\eta(1, -1)\|} = \frac{\mathbf{x}_\xi(1, -1)}{\|\mathbf{x}_\xi(1, -1)\|}, \quad \frac{\mathbf{x}_\eta(1, 1)}{\|\mathbf{x}_\eta(1, 1)\|} = -\frac{\mathbf{x}_\xi(1, 1)}{\|\mathbf{x}_\xi(1, 1)\|}. \quad (38)$$

At vertices  $B$  and  $C$ , the relations (21c)–(21d) are still valid due to the Dirichlet BCs on  $\overline{AB}$  and  $\overline{CD}$ . Employing the chain rule, we have

$$\begin{aligned} V_\eta(1, -1) &= \nabla u(\mathbf{x}_B) \cdot \mathbf{x}_\eta(1, -1) = \nabla u(\mathbf{x}_B) \cdot \mathbf{x}_\xi(1, -1) \frac{\|\mathbf{x}_\eta(1, -1)\|}{\|\mathbf{x}_\xi(1, -1)\|} = \frac{\|\mathbf{x}_\eta(1, -1)\|}{\|\mathbf{x}_\xi(1, -1)\|} V_\xi(1, -1) \\ &= \frac{\|\mathbf{x}_\eta(1, -1)\|}{\|\mathbf{x}_\xi(1, -1)\|} F_\xi(1, -1) = F_\eta^a(1, -1), \\ V_\eta(1, 1) &= \nabla u(\mathbf{x}_C) \cdot \mathbf{x}_\eta(1, 1) = -\nabla u(\mathbf{x}_C) \cdot \mathbf{x}_\xi(1, 1) \frac{\|\mathbf{x}_\eta(1, 1)\|}{\|\mathbf{x}_\xi(1, 1)\|} = -\frac{\|\mathbf{x}_\eta(1, 1)\|}{\|\mathbf{x}_\xi(1, 1)\|} V_\xi(1, 1) \end{aligned} \quad (39a)$$

$$= -\frac{\|\mathbf{x}_\eta(1, 1)\|}{\|\mathbf{x}_\xi(1, 1)\|} F_\xi(1, 1) = F_\eta^a(1, 1), \quad (39b)$$

where we have used (38) and (21c)–(21d). The  $V(\xi, \eta)$  given by (35) is still valid, in which  $F_\eta^a(1, -1)$  and  $F_\eta^a(1, 1)$  should now be computed using (39). The gradients  $\nabla u(\mathbf{x}_B)$  and  $\nabla u(\mathbf{x}_C)$  can be determined by combining the Neumann condition (17b), evaluated at these vertices, with the Eqs. (21c)–(21d), and by applying the chain rule and using (37). This leads to the result,

$$\nabla u(\mathbf{x}_B) = \begin{bmatrix} F_{nBC}(-1) & \frac{1}{\|\mathbf{x}_\xi(1, -1)\|} F_\xi(1, -1) \end{bmatrix} \begin{bmatrix} \tau_{yBC}(-1) & -\tau_{xBC}(-1) \\ \tau_{xBC}(-1) & \tau_{yBC}(-1) \end{bmatrix}, \quad (40a)$$

$$\nabla u(\mathbf{x}_C) = \begin{bmatrix} F_{nBC}(1) & -\frac{1}{\|\mathbf{x}_\xi(1, 1)\|} F_\xi(1, 1) \end{bmatrix} \begin{bmatrix} \tau_{yBC}(1) & -\tau_{xBC}(1) \\ \tau_{xBC}(1) & \tau_{yBC}(1) \end{bmatrix}, \quad (40b)$$

where  $\tau_{BC}(\eta) = (\tau_{xBC}, \tau_{yBC})$ , and  $F_{nBC}(\eta)$  is defined in (19).

**Remark 2.4.** When two Neumann conditions are imposed on opposite sides of the quadrilateral domain  $\Omega$ , with Dirichlet conditions on the other boundaries, the general form for  $V(\xi, \eta)$  that exactly satisfies these boundary conditions can be developed analogously by following the four-step procedure as described above.

### 2.3.2. When two Neumann boundaries intersect at a vertex

For this case we focus on the setting with the Neumann conditions imposed on two adjacent boundaries of the quadrilateral domain and with the rest being Dirichlet boundaries. The settings with more than two Neumann boundaries are discussed in a remark at the end of this section. Without loss of generality, we assume that the Neumann conditions are imposed on the boundaries  $\overline{BC}$  and  $\overline{CD}$  and that  $\overline{AB}$  and  $\overline{AD}$  are Dirichlet boundaries. The Jacobian matrix of the map  $\mathbf{x}(\xi, \eta)$  is assumed to be non-singular everywhere in the domain.

Specifically, we seek a scalar field function  $u(\mathbf{x})$ , for  $\mathbf{x} \in \Omega = \overline{ABCD}$  in Fig. 1(a), which satisfies the following boundary conditions,

$$u|_{\mathbf{x} \in \overline{AB}} = u_{AB}(\mathbf{x}), \quad (41a)$$

$$\mathbf{n} \cdot \nabla u|_{\mathbf{x} \in \overline{BC}} = u_{nBC}(\mathbf{x}), \quad (41b)$$

$$\mathbf{n} \cdot \nabla u|_{\mathbf{x} \in \overline{CD}} = u_{nCD}(\mathbf{x}), \quad (41c)$$

$$u|_{\mathbf{x} \in \overline{AD}} = u_{AD}(\mathbf{x}), \quad (41d)$$

where  $u_{nCD}(\mathbf{x})$  is the prescribed Neumann boundary distribution on  $\overline{CD}$ , and the other notations follow those in the previous sections. The prescribed Dirichlet and Neumann boundary functions must be compatible on the shared vertices.

Employing the map  $\mathbf{x}(\xi, \eta)$ , we transform  $u(\mathbf{x})$ ,  $u_{AB}(\mathbf{x})$  and  $u_{AD}(\mathbf{x})$  into  $V(\xi, \eta)$ ,  $F(\xi, -1)$  and  $F(-1, \eta)$  according to Eqs. (12), (13a) and (13d). The Neumann condition (41b) is accordingly transformed into (18). The Neumann condition (41c) becomes

$$V_\eta(\xi, 1) + S_{CD}(\xi) V_\xi(\xi, 1) = T_{CD}(\xi), \quad \xi \in [-1, 1], \quad (42a)$$

$$\text{or } V_\eta(\xi, 1) = T_{CD}(\xi) - S_{CD}(\xi) V_\xi(\xi, 1) = F_\eta(\xi, 1), \quad (42b)$$

where we have used (20) and

$$\begin{cases} S_{CD}(\xi) = \frac{K_{xCD}(\xi)}{K_{yCD}(\xi)}, & T_{CD}(\xi) = \frac{F_{nCD}(\xi)}{K_{yCD}(\xi)} = \frac{u_{nCD}(\mathbf{x}(\xi, 1))}{K_{yCD}(\xi)}, \\ \mathbf{K}_{CD}(\xi) = \begin{bmatrix} K_{xCD}(\xi) \\ K_{yCD}(\xi) \end{bmatrix} = \mathbf{J}^{-1}(\xi, 1) \begin{bmatrix} n_{xCD}(\xi) \\ n_{yCD}(\xi) \end{bmatrix} = \frac{1}{\det \mathbf{J}(\xi, 1)} \begin{bmatrix} -\frac{\mathbf{x}_\xi(\xi, 1) \cdot \mathbf{x}_\eta(\xi, 1)}{\|\mathbf{x}_\xi(\xi, 1)\|} \\ \|\mathbf{x}_\xi(\xi, 1)\| \end{bmatrix}. \end{cases} \quad (43)$$

The Neumann boundary  $\overline{BC}$  and the Dirichlet boundary  $\overline{AB}$  intersect at vertex  $B$ , inducing the compatibility constraints (21a), (21c), and (22) at vertex  $B$ . Similar compatibility conditions exist at vertex  $D$ , where the Neumann boundary  $\overline{CD}$  and Dirichlet boundary  $\overline{AD}$  intersect. These are

$$V(-1, 1) = F(-1, 1) = \lim_{\eta \rightarrow 1} F(-1, \eta), \quad (44a)$$

$$V_\eta(-1, 1) = F_\eta(-1, 1) = \lim_{\eta \rightarrow 1} F_\eta(-1, \eta), \quad (44b)$$

and

$$V_\xi(-1, 1) = \frac{1}{S_{CD}(-1)} [T_{CD}(-1) - F_\eta(-1, 1)] = F_\xi^a(-1, 1), \quad \text{if } \overline{AD} \not\perp \overline{CD} \text{ at } D; \quad (45a)$$

$$F_\eta(-1, 1) - T_{CD}(-1) = 0, \quad \text{if } \overline{AD} \perp \overline{CD} \text{ at } D. \quad (45b)$$

The Neumann conditions (18) and (42) must be compatible at vertex  $C$ . Evaluating (18a) and (42a) at vertex  $C$  and combining them leads to

$$V_\xi(1, 1) = \frac{T_{BC}(1) - S_{BC}(1) T_{CD}(1)}{1 - S_{BC}(1) S_{CD}(1)} = F_\xi^a(1, 1), \quad (46a)$$

**Table 2**Interpolation properties of  $C^2$  Hermite interpolation polynomials defined on  $\xi \in [-1, 1]$ .

| $\xi$ | $\rho_0(\xi)$ | $\rho'_0(\xi)$ | $\rho''_0(\xi)$ | $\rho_1(\xi)$   | $\rho'_1(\xi)$   | $\rho''_1(\xi)$   | $v_0(\xi)$      | $v'_0(\xi)$      | $v''_0(\xi)$      |
|-------|---------------|----------------|-----------------|-----------------|------------------|-------------------|-----------------|------------------|-------------------|
| -1    | 1             | 0              | 0               | 0               | 0                | 0                 | 0               | 1                | 0                 |
| 1     | 0             | 0              | 0               | 1               | 0                | 0                 | 0               | 0                | 0                 |
| $\xi$ | $v_1(\xi)$    | $v'_1(\xi)$    | $v''_1(\xi)$    | $\omega_0(\xi)$ | $\omega'_0(\xi)$ | $\omega''_0(\xi)$ | $\omega_1(\xi)$ | $\omega'_1(\xi)$ | $\omega''_1(\xi)$ |
| -1    | 0             | 0              | 0               | 0               | 0                | 1                 | 0               | 0                | 0                 |
| 1     | 0             | 1              | 0               | 0               | 0                | 0                 | 0               | 0                | 1                 |

$$V_\eta(1, 1) = \frac{T_{CD}(1) - S_{CD}(1)T_{BC}(1)}{1 - S_{BC}(1)S_{CD}(1)} = F_\eta^a(1, 1). \quad (46b)$$

Note that  $S_{BC}(1)S_{CD}(1) < 1$  by the Cauchy–Schwarz inequality for a non-singular Jacobian matrix at  $C$ . By differentiating (18b) with respect to (w.r.t.)  $\eta$  and (42b) w.r.t.  $\xi$ , and evaluating them at vertex  $C$ , we get

$$V_{\xi\eta}(1, 1) = T'_{BC}(1) - S'_{BC}(1)F_\eta^a(1, 1) - S_{BC}(1)V_{\eta\eta}(1, 1) = F_{\xi\eta}(1, 1), \quad (47a)$$

$$V_{\eta\xi}(1, 1) = T'_{CD}(1) - S'_{CD}(1)F_\xi^a(1, 1) - S_{CD}(1)V_{\xi\xi}(1, 1) = F_{\eta\xi}(1, 1), \quad (47b)$$

where we have used (46). A combination of (47a) and (47b) (requiring  $V_{\xi\eta}(1, 1) = V_{\eta\xi}(1, 1)$ ) leads to the following compatibility constraints,

$$V_{\xi\xi}(1, 1) = \frac{S_{BC}(1)}{S_{CD}(1)}V_{\eta\eta}(1, 1) + \frac{R_C}{S_{CD}(1)} = F_{\xi\xi}(1, 1), \quad \text{if } \overline{BC} \not\perp \overline{CD} \text{ at } C, \quad (48a)$$

$$T'_{CD}(1) - S'_{CD}(1)F_\xi^a(1, 1) = T'_{BC}(1) - S'_{BC}(1)F_\eta^a(1, 1), \quad \text{if } \overline{BC} \perp \overline{CD} \text{ at } C, \quad (48b)$$

where  $R_C = [T'_{CD}(1) - S'_{CD}(1)F_\xi^a(1, 1)] - [T'_{BC}(1) - S'_{BC}(1)F_\eta^a(1, 1)]$ , and we have used the fact that  $S_{BC}(1) = S_{CD}(1) = 0$  when  $\overline{BC}$  and  $\overline{CD}$  are orthogonal at vertex  $C$ . Eq. (48b) imposes a constraint on the prescribed Neumann boundary data when  $\overline{BC}$  and  $\overline{CD}$  are orthogonal at  $C$ , and if they are not orthogonal, Eq. (48a) imposes a constraint between  $V_{\xi\xi}(1, 1)$  and  $V_{\eta\eta}(1, 1)$  at vertex  $C$ .

Our goal is to formulate the field function  $V(\xi, \eta)$ , for  $(\xi, \eta) \in \Omega_{st}$ , so that it exactly satisfies the boundary conditions (14a), (14d), (18), and (42), together with the compatibility constraints (21a), (22a), (44a), (45a), (46), (48a) and (47a).

To facilitate the subsequent discussions, we recall the  $C^2$  Hermite interpolation polynomials  $\rho_0(\xi)$ ,  $\rho_1(\xi)$ ,  $v_0(\xi)$ ,  $v_1(\xi)$ ,  $\omega_0(\xi)$  and  $\omega_1(\xi)$  defined on  $\xi \in [-1, 1]$  that satisfy the interpolation properties listed in Table 2. These polynomials are given by

$$\begin{aligned} \rho_0(\xi) &= \phi_0^3(\xi) [1 + 3\phi_1(\xi) + 6\phi_1^2(\xi)], & v_0(\xi) &= 2\phi_0^3(\xi)\phi_1(\xi) [1 + 3\phi_1(\xi)], & \omega_0(\xi) &= 2\phi_0^3(\xi)\phi_1^2(\xi), \\ \rho_1(\xi) &= \phi_1^3(\xi) [1 + 3\phi_0(\xi) + 6\phi_0^2(\xi)], & v_1(\xi) &= -2\phi_1^3(\xi)\phi_0(\xi) [1 + 3\phi_0(\xi)], & \omega_1(\xi) &= 2\phi_1^3(\xi)\phi_0^2(\xi), \end{aligned} \quad (49)$$

where  $\phi_0(\xi)$  and  $\phi_1(\xi)$  are defined in (7). Besides the constants  $\lambda_B$  and  $\lambda_C$  defined in (25), we define an additional constant  $\lambda_D$  to flag whether the boundaries  $\overline{CD}$  and  $\overline{AD}$  are orthogonal at vertex  $D$ ,

$$\lambda_D = \begin{cases} 0, & \text{if } \overline{CD} \perp \overline{AD} \text{ at } D, \\ 1, & \text{if } \overline{CD} \not\perp \overline{AD} \text{ at } D. \end{cases} \quad (50)$$

We follow the four-step procedure as described in Section 2.3.1 to develop the general form for  $V(\xi, \eta)$ . First, we note that the conditions (21a), (22a) and (46b) are actually constraints on  $V(1, \eta)$ , which can be satisfied by the following profile on  $\overline{BC}$ ,

$$F(1, \eta) = F(1, -1)\rho_0(\eta) + \lambda_B F_\eta(1, -1)v_0(\eta) + F_\eta(1, 1)v_1(\eta), \quad (51)$$

where the constant  $\lambda_B$  ensures that the condition (22a) is enforced only when  $\overline{AB}$  and  $\overline{BC}$  are not orthogonal at  $B$ , and

$$F_\eta(1, -1) = F_\eta^a(1, -1), \quad F_\eta(1, 1) = F_\eta^a(1, 1), \quad (52)$$

with  $F_\eta^a(1, -1)$  and  $F_\eta^a(1, 1)$  given in (22a) and (46b). The conditions (44a), (45a), (46a) and (48a) are actually constraints on  $V(\xi, 1)$ . They can be satisfied by the following profile on  $\overline{CD}$ ,

$$F(\xi, 1) = F(-1, 1)\rho_0(\xi) + \lambda_D F_\xi(-1, 1)v_0(\xi) + F_\xi(1, 1)v_1(\xi) + \lambda_C F_{\xi\xi}(1, 1)\omega_1(\xi), \quad (53)$$

where the constants  $\lambda_D$  and  $\lambda_C$  ensure that the conditions (45a) and (48a) are only imposed when the boundary curves are not orthogonal at  $D$  or  $C$ ,  $F_{\xi\xi}(1, 1)$  is given in (48a), and

$$F_\xi(-1, 1) = F_\xi^a(-1, 1), \quad F_\xi(1, 1) = F_\xi^a(1, 1), \quad (54)$$

with  $F_\xi^a(-1, 1)$  and  $F_\xi^a(1, 1)$  defined in (45a) and (46a). Note that the profile (51) for  $\overline{BC}$  and the profile (53) for  $\overline{CD}$  are compatible at vertex  $C$ , resulting in  $F(1, 1) = 0$ .

With  $F(1, \eta)$  and  $F(\xi, 1)$  introduced above, our task is then reduced to: find  $V(\xi, \eta)$  such that

$$V(-1, \eta) = F(-1, \eta), \quad (55a)$$

$$V(1, \eta) = F(1, \eta) = F(1, -1)\rho_0(\eta) + \lambda_B F_\eta(1, -1)v_0(\eta) + F_\eta(1, 1)v_1(\eta), \quad (55b)$$

$$V_\xi(1, \eta) = F_\xi(1, \eta), \quad (55c)$$

$$V(\xi, -1) = F(\xi, -1), \quad (55d)$$

$$V(\xi, 1) = F(\xi, 1) = F(-1, 1)\rho_0(\xi) + \lambda_D F_\xi(-1, 1)v_0(\xi) + F_\xi(1, 1)v_1(\xi) + \lambda_C F_{\xi\xi}(1, 1)\omega_1(\xi), \quad (55e)$$

$$V_\eta(\xi, 1) = F_\eta(\xi, 1), \quad (55f)$$

$$V_{\xi\eta}(1, 1) = F_{\xi\eta}(1, 1), \quad (55g)$$

where  $F(-1, \eta)$  and  $F(\xi, -1)$  are given in (13d) and (13a),  $F_\xi(1, \eta)$  is given in (18b),  $F_\eta(\xi, 1)$  is given in (42b), and  $F_{\xi\eta}(1, 1)$  is given in (47a), and we have used (51) and (53).

The transfinite interpolation for the conditions in (55) is given by

$$\begin{aligned} PF(\xi, \eta) &= (P_1 \oplus P_2)F(\xi, \eta) = P_1 F(\xi, \eta) + P_2 F(\xi, \eta) - P_1 P_2 F(\xi, \eta) \\ &= F(-1, \eta)\rho_0(\xi) + F_\xi(1, \eta)v_1(\xi) + F(\xi, -1)\rho_0(\eta) + F_\eta(\xi, 1)v_1(\eta) \\ &\quad - [F(-1, -1)\rho_0(\eta) + F_\eta(-1, 1)v_1(\eta)] \rho_0(\xi) - [F_\xi(1, -1)\rho_0(\eta) + F_{\xi\eta}(1, 1)v_1(\eta)] v_1(\xi) \\ &\quad + \lambda_B F_\eta(1, -1)v_0(\eta)\rho_1(\xi) + [\lambda_D F_\xi(-1, 1)v_0(\xi) + \lambda_C F_{\xi\xi}(1, 1)\omega_1(\xi)] \rho_1(\eta), \end{aligned} \quad (56)$$

where

$$\left\{ \begin{aligned} P_1 F(\xi, \eta) &= F(-1, \eta)\rho_0(\xi) + F_\xi(1, \eta)v_1(\xi) + F(1, \eta)\rho_1(\xi) \\ &= F(-1, \eta)\rho_0(\xi) + F_\xi(1, \eta)v_1(\xi) \\ &\quad + [F(1, -1)\rho_0(\eta) + \lambda_B F_\eta(1, -1)v_0(\eta) + F_\eta(1, 1)v_1(\eta)] \rho_1(\xi), \\ P_2 F(\xi, \eta) &= F(\xi, -1)\rho_0(\eta) + F_\eta(\xi, 1)v_1(\eta) + F(\xi, 1)\rho_1(\eta) \\ &= F(\xi, -1)\rho_0(\eta) + F_\eta(\xi, 1)v_1(\eta) \\ &\quad + [F(-1, 1)\rho_0(\xi) + \lambda_D F_\xi(-1, 1)v_0(\xi) + F_\xi(1, 1)v_1(\xi) + \lambda_C F_{\xi\xi}(1, 1)\omega_1(\xi)] \rho_1(\eta). \end{aligned} \right. \quad (57)$$

This leads to the preliminary form  $V(\xi, \eta)$  for the conditions in (55),

$$V(\xi, \eta) = g(\xi, \eta) - Pg(\xi, \eta) + PF(\xi, \eta), \quad (58)$$

where  $g(\xi, \eta)$  is a free (arbitrary) function, and

$$\begin{aligned} Pg(\xi, \eta) &= g(-1, \eta)\rho_0(\xi) + g_\xi(1, \eta)v_1(\xi) + g(\xi, -1)\rho_0(\eta) + g_\eta(\xi, 1)v_1(\eta) \\ &\quad - [g(-1, -1)\rho_0(\eta) + g_\eta(-1, 1)v_1(\eta)] \rho_0(\xi) - [g_\xi(1, -1)\rho_0(\eta) + g_{\xi\eta}(1, 1)v_1(\eta)] v_1(\xi) \\ &\quad + \lambda_B g_\eta(1, -1)v_0(\eta)\rho_1(\xi) + [\lambda_D g_\xi(-1, 1)v_0(\xi) + \lambda_C g_{\xi\xi}(1, 1)\omega_1(\xi)] \rho_1(\eta). \end{aligned} \quad (59)$$

$V(\xi, \eta)$  from (58) has the following properties,

$$V_{\eta\eta}(1, 1) = g_{\eta\eta}(1, 1), \quad (60a)$$

$$\begin{aligned} V_\eta(1, \eta) &= g_\eta(1, \eta) - [g(1, -1) - F(1, -1)] \rho'_0(\eta) - [g_\eta(1, 1) - F_\eta(1, 1)] v'_1(\eta) \\ &\quad - \lambda_B [g_\eta(1, -1) - F_\eta(1, -1)] v'_0(\eta), \end{aligned} \quad (60b)$$

$$\begin{aligned} V_\xi(\xi, 1) &= g_\xi(\xi, 1) - [g(-1, 1) - F(-1, 1)] \rho'_0(\xi) - [g_\xi(1, 1) - F_\xi(1, 1)] v'_1(\xi) \\ &\quad - \lambda_D [g_\xi(-1, 1) - F_\xi(-1, 1)] v'_0(\xi) - \lambda_C [g_{\xi\xi}(1, 1) - F_{\xi\xi}(1, 1)] \omega'_1(\xi). \end{aligned} \quad (60c)$$

Employing (18b), (42b), (47a), (48a), (52), (54), and (60), we update the transfinite interpolation  $PF(\xi, \eta)$  in (56) by,

$$\begin{aligned} PF^g(\xi, \eta) &= F(-1, \eta)\rho_0(\xi) + F_\xi^g(1, \eta)v_1(\xi) + F(\xi, -1)\rho_0(\eta) + F_\eta^g(\xi, 1)v_1(\eta) \\ &\quad - [F(-1, -1)\rho_0(\eta) + F_\eta(-1, 1)v_1(\eta)] \rho_0(\xi) - [F_\xi(1, -1)\rho_0(\eta) + F_{\xi\eta}^g(1, 1)v_1(\eta)] v_1(\xi) \\ &\quad + \lambda_B F_\eta^g(1, -1)v_0(\eta)\rho_1(\xi) + [\lambda_D F_\xi^g(-1, 1)v_0(\xi) + \lambda_C F_{\xi\xi}^g(1, 1)\omega_1(\xi)] \rho_1(\eta), \end{aligned} \quad (61)$$

where

$$F_{\xi\xi}^g(1, 1) = \frac{S_{BC}(1)}{S_{CD}(1)} g_{\eta\eta}(1, 1) + \frac{R_C}{S_{CD}(1)}, \quad (62a)$$

$$F_{\xi\eta}^g(1, 1) = T'_{BC}(1) - S'_{BC}(1)F_\eta^a(1, 1) - S_{BC}(1)g_{\eta\eta}(1, 1), \quad (62b)$$

$$\begin{aligned} F_\xi^g(1, \eta) &= T_{BC}(\eta) - S_{BC}(\eta) \left\{ g_\eta(1, \eta) - [g(1, -1) - F(1, -1)] \rho'_0(\eta) - [g_\eta(1, 1) - F_\eta^a(1, 1)] v'_1(\eta) \right. \\ &\quad \left. - \lambda_B [g_\eta(1, -1) - F_\eta^a(1, -1)] v'_0(\eta) \right\}, \end{aligned} \quad (62c)$$

$$F_\eta^g(\xi, 1) = T_{CD}(\xi) - S_{CD}(\xi) \left\{ g_\xi(\xi, 1) - [g(-1, 1) - F(-1, 1)] \rho'_0(\xi) - [g_\xi(1, 1) - F_\xi^a(1, 1)] v'_1(\xi) \right.$$

$$-\lambda_D \left[ g_\xi(-1, 1) - F_\xi^a(-1, 1) \right] v_0'(\xi) - \lambda_C \left[ g_{\xi\xi}(1, 1) - F_{\xi\xi}^g(1, 1) \right] \omega_1'(\xi) \}. \quad (62d)$$

The final form for  $V(\xi, \eta)$  is then given by

$$V(\xi, \eta) = g(\xi, \eta) - Pg(\xi, \eta) + PF^g(\xi, \eta) \quad (63)$$

where  $g(\xi, \eta)$  is a free function,  $Pg(\xi, \eta)$  is given by (59), and  $PF^g(\xi, \eta)$  is given by (61). In this expression  $F(-1, -1) = \lim_{\xi \rightarrow -1} F(\xi, -1) = u_A$ , and  $F_\eta(-1, 1)$  and  $F_\xi(1, -1)$  are given by (44b) and (21c).  $F_\xi^a(-1, 1)$  and  $F_\eta^a(1, -1)$  are given by (45a) and (22a).  $F_\xi^a(1, 1)$  and  $F_\eta^a(1, 1)$  are given in (46).

**Theorem 2.2.**  $V(\xi, \eta)$  given by (63) satisfies the Dirichlet conditions (55a) and (55d) and the Neumann conditions (18a) and (42a), for any  $g(\xi, \eta)$  therein that is sufficiently differentiable.

**Proof.** To verify (55a) and (55d), one only needs to notice that  $F_\eta^g(-1, 1) = F_\eta(-1, 1)$  and  $F_\xi^g(1, -1) = F_\xi(1, -1)$ , due to (45) and (22). Consequently  $PF^g(-1, \eta) = F(-1, \eta)$  and  $PF^g(\xi, -1) = F(\xi, -1)$ .

To verify (42a), one notes that  $F_{\xi, \eta}^g(1, 1) = F_{\xi, \eta}^g(1, 1)$ . Hence  $PF_\eta^g(\xi, 1) = F_\eta^g(\xi, 1)$  and  $V_\eta(\xi, 1) = F_\eta^g(\xi, 1)$ . On the other hand,

$$\begin{aligned} V_\xi(\xi, 1) &= g_\xi(\xi, 1) - [g(-1, 1) - F(-1, 1)] \rho_0'(\xi) - \left[ g_\xi(1, 1) - F_\xi^a(1, 1) \right] v_1'(\xi) \\ &\quad - \lambda_D \left[ g_\xi(-1, 1) - F_\xi^a(-1, 1) \right] v_0'(\xi) - \lambda_C \left[ g_{\xi\xi}(1, 1) - F_{\xi\xi}^g(1, 1) \right] \omega_1'(\xi). \end{aligned}$$

Therefore (42a) holds.

To verify (18a), one notes that  $F_{\eta, \xi}^g(1, 1) = F_{\eta, \xi}^g(1, 1)$ . Hence  $PF_\xi^g(1, \eta) = F_\xi^g(1, \eta)$  and  $V_\xi(1, \eta) = F_\xi^g(1, \eta)$ . On the other hand,

$$\begin{aligned} V_\eta(1, \eta) &= g_\eta(1, \eta) - [g(1, -1) - F(1, -1)] \rho_0'(\eta) - \left[ g_\eta(1, 1) - F_\eta^a(1, 1) \right] v_1'(\eta) \\ &\quad - \lambda_B \left[ g_\eta(1, -1) - F_\eta^a(1, -1) \right] v_0'(\eta). \end{aligned}$$

Therefore (18a) holds.  $\square$

**Remark 2.5.** When the domain involves more than two Neumann boundaries, the compatibility constraints as discussed above exist at any vertex where two Neumann boundaries intersect, and those constraints discussed in Section 2.3.1 exist at any vertex where a Neumann boundary and a Dirichlet boundary intersect. The field function that exactly satisfies these conditions can be formulated analogously using the four-step procedure described in Section 2.3.1.

## 2.4. Enforcing Robin boundary conditions

We next look into how to enforce Robin boundary conditions (RBC) exactly on the general quadrilateral domain  $\Omega$  as shown in Fig. 1(a). The formulation for Robin condition is largely similar to that for the Neumann condition, apart from the complication caused by the unknown function value on the Robin boundary. Here we assume that the domain involves a combination of Dirichlet and Robin type boundaries, with at least one Robin boundary and one Dirichlet boundary. The cases when a Robin boundary only intersects with Dirichlet boundaries and when two Robin boundaries intersect with each other are discussed individually below.

### 2.4.1. When Robin boundary only intersects with Dirichlet boundary

For this case we focus on the setting in which the domain has a single Robin boundary, with the rest being Dirichlet types. Without loss of generality, we assume that the Robin condition is imposed on  $\overline{BC}$ . The goal is to formulate the field function  $u(\mathbf{x})$ , for  $\mathbf{x} \in \Omega = ABCD$ , which satisfies the following conditions,

$$u|_{\mathbf{x} \in \overline{AB}} = u_{AB}(\mathbf{x}), \quad (64a)$$

$$\mathbf{n} \cdot \nabla u|_{\mathbf{x} \in \overline{BC}} + \alpha_{BC} u|_{\mathbf{x} \in \overline{BC}} = u_{rBC}(\mathbf{x}), \quad (64b)$$

$$u|_{\mathbf{x} \in \overline{CD}} = u_{CD}(\mathbf{x}), \quad (64c)$$

$$u|_{\mathbf{x} \in \overline{AD}} = u_{AD}(\mathbf{x}), \quad (64d)$$

where  $\alpha_{BC}$  is a prescribed constant, and  $u_{rBC}(\mathbf{x})$  denotes the prescribed Robin boundary function.

By leveraging the map  $\mathbf{x}(\xi, \eta)$ , we transform  $u(\mathbf{x})$  into  $V(\xi, \eta)$  according to (12), and the conditions (64a) and (64c)–(64d) into (14a) and (14c)–(14d) according to (13a) and (13c)–(13d). The Robin condition (64b) is transformed into

$$V_\xi(1, \eta) + S_{BC}(\eta) V_\eta(1, \eta) + \alpha_{BC} W_{BC}(\eta) V(1, \eta) = T_{rBC}(\eta), \quad \eta \in [-1, 1], \quad (65a)$$

$$\text{or } V_\xi(1, \eta) = T_{rBC}(\eta) - S_{BC}(\eta) V_\eta(1, \eta) - \alpha_{BC} W_{BC}(\eta) V(1, \eta) = F_\xi(1, \eta), \quad (65b)$$

where  $S_{BC}(\eta)$  and  $K_{xBC}(\eta)$  are defined in (19), and

$$W_{BC}(\eta) = \frac{1}{K_{xBC}(\eta)}, \quad T_{rBC}(\eta) = F_{rBC}(\eta) W_{BC}(\eta), \quad F_{rBC}(\eta) = u_{rBC}(\mathbf{x}(1, \eta)). \quad (66)$$

These conditions must be compatible at the shared vertices.

The compatibility between the Robin condition (65) and the Dirichlet condition (14a) at vertex  $B$  results in (21a) and (21c), together with

$$\begin{aligned} V_\eta(1, -1) &= \frac{1}{S_{BC}(-1)} [T_{rBC}(-1) - F_\xi(1, -1) - \alpha_{BC} W_{BC}(-1)F(1, -1)] \\ &= F_\eta^a(1, -1), \end{aligned} \quad \text{if } \overline{AB} \not\perp \overline{BC} \text{ at } B, \quad (67a)$$

$$F_\xi(1, -1) + \alpha_{BC} W_{BC}(-1)F(1, -1) - T_{rBC}(-1) = 0, \quad \text{if } \overline{AB} \perp \overline{BC} \text{ at } B, \quad (67b)$$

by noting that  $S_{BC}(-1) = 0$  when  $\overline{AB} \perp \overline{BC}$  at  $B$ . The compatibility between (65) and (14c) at vertex  $C$  results in (21b) and (21d), together with

$$V_\eta(1, 1) = \frac{1}{S_{BC}(1)} [T_{rBC}(1) - F_\xi(1, 1) - \alpha_{BC} W_{BC}(1)F(1, 1)] = F_\eta^a(1, 1), \quad \text{if } \overline{CD} \not\perp \overline{BC} \text{ at } C, \quad (68a)$$

$$F_\xi(1, 1) + \alpha_{BC} W_{BC}(1)F(1, 1) - T_{rBC}(1) = 0, \quad \text{if } \overline{CD} \perp \overline{BC} \text{ at } C. \quad (68b)$$

The conditions (14a), (14c), (14d), (65), (67a), (68a), and (21a)–(21b) constitute the constraints that the function  $V(\xi, \eta)$  to be formulated must satisfy.

We follow the four-step procedure, to first introduce the same transfinite interpolation as in (30), where  $F_\eta(1, -1)$  and  $F_\eta(1, 1)$  are given by (27), with  $F_\eta^a(1, -1)$  and  $F_\eta^a(1, 1)$  therein now given by (67a) and (68a), and  $F_\xi(1, \eta)$  is now given by (65b). The preliminary form for  $V(\xi, \eta)$  is given by (31), in which  $Pg(\xi, \eta)$  is given by (32). This preliminary form has the following properties on  $\overline{BC}$ ,

$$\begin{aligned} V(1, \eta) &= g(1, \eta) - [g(1, -1) - F(1, -1)] \varphi_0(\eta) - [g(1, 1) - F(1, 1)] \varphi_1(\eta) \\ &\quad - \lambda_B [g_\eta(1, -1) - F_\eta(1, -1)] \psi_0(\eta) - \lambda_C [g_\eta(1, 1) - F_\eta(1, 1)] \psi_1(\eta), \end{aligned} \quad (69a)$$

$$\begin{aligned} V_\eta(1, \eta) &= g_\eta(1, \eta) - [g(1, -1) - F(1, -1)] \varphi'_0(\eta) - [g(1, 1) - F(1, 1)] \varphi'_1(\eta) \\ &\quad - \lambda_B [g_\eta(1, -1) - F_\eta(1, -1)] \psi'_0(\eta) - \lambda_C [g_\eta(1, 1) - F_\eta(1, 1)] \psi'_1(\eta). \end{aligned} \quad (69b)$$

In light of (69) and (27), we update  $F_\xi(1, \eta)$  in (65b) by

$$\begin{aligned} F_\xi^g(1, \eta) &= T_{rBC}(\eta) - S_{BC}(\eta) \{g_\eta(1, \eta) - [g(1, -1) - F(1, -1)] \varphi'_0(\eta) - [g(1, 1) - F(1, 1)] \varphi'_1(\eta) \\ &\quad - \lambda_B [g_\eta(1, -1) - F_\eta^a(1, -1)] \psi'_0(\eta) - \lambda_C [g_\eta(1, 1) - F_\eta^a(1, 1)] \psi'_1(\eta)\} \\ &\quad - \alpha_{BC} W_{BC}(\eta) \{g(1, \eta) - [g(1, -1) - F(1, -1)] \varphi_0(\eta) - [g(1, 1) - F(1, 1)] \varphi_1(\eta) \\ &\quad - \lambda_B [g_\eta(1, -1) - F_\eta^a(1, -1)] \psi_0(\eta) - \lambda_C [g_\eta(1, 1) - F_\eta^a(1, 1)] \psi_1(\eta)\}. \end{aligned} \quad (70)$$

Therefore, the updated transfinite interpolation is given by (34b), in which  $F_\xi^g(1, \eta)$  is now given by (70), and  $F_\eta^a(1, -1)$  and  $F_\eta^a(1, 1)$  are now given by (67a) and (68a). The final  $V(\xi, \eta)$  has the same form as in (63).

**Theorem 2.3.**  $V(\xi, \eta)$  given by (35), with  $F_\xi^g(1, \eta)$ ,  $F_\eta^a(1, -1)$  and  $F_\eta^a(1, 1)$  therein given by (70), (67a) and (68a), satisfies the Dirichlet conditions (14a) and (14c)–(14d) and the Robin condition (65a), for any  $g(\xi, \eta)$  therein that is sufficiently differentiable.

**Proof.** By verification.  $\square$

**Remark 2.6.** When two Robin boundaries are imposed on opposite sides of the quadrilateral domain, with the rest being Dirichlet boundaries, the general form for  $V(\xi, \eta)$  that exactly satisfies these conditions can be formulated analogously by following the four-step procedure.

#### 2.4.2. When two Robin boundaries intersect at a vertex

For this case we focus on the setting in which the Robin conditions are imposed on two adjacent boundaries with the rest being Dirichlet boundaries. Without loss of generality we assume that the Robin conditions are imposed on the boundaries  $\overline{BC}$  and  $\overline{CD}$ .

Specifically, we seek a field function  $u(\mathbf{x})$ , for  $\mathbf{x} \in \Omega = \overline{ABCD}$ , which satisfies the following boundary conditions,

$$u|_{\mathbf{x} \in \overline{AB}} = u_{AB}(\mathbf{x}), \quad (71a)$$

$$\mathbf{n} \cdot \nabla u|_{\mathbf{x} \in \overline{BC}} + \alpha_{BC} u|_{\mathbf{x} \in \overline{BC}} = u_{rBC}(\mathbf{x}), \quad (71b)$$

$$\mathbf{n} \cdot \nabla u|_{\mathbf{x} \in \overline{CD}} + \alpha_{CD} u|_{\mathbf{x} \in \overline{CD}} = u_{rCD}(\mathbf{x}), \quad (71c)$$

$$u|_{\mathbf{x} \in \overline{AD}} = u_{AD}(\mathbf{x}), \quad (71d)$$

where  $\alpha_{BC}$  and  $\alpha_{CD}$  are prescribed constants, and  $u_{rBC}(\mathbf{x})$  and  $u_{rCD}(\mathbf{x})$  are prescribed Robin boundary functions.

Employing the map  $\mathbf{x}(\xi, \eta)$ , we transform  $u(\mathbf{x})$  into  $V(\xi, \eta)$  according to (12), and the Dirichlet conditions (71a) and (71d) into (14a) and (14d) based on (13a) and (13d). The Robin condition (71b) is accordingly transformed into (65). The Robin condition (71c) is transformed into

$$V_\eta(\xi, 1) + S_{CD}(\xi) V_\xi(\xi, 1) + \alpha_{CD} W_{CD}(\xi) V(\xi, 1) = T_{rCD}(\xi), \quad \xi \in [-1, 1], \quad (72a)$$

$$\text{or } V_\eta(\xi, 1) = T_{rCD}(\xi) - S_{CD}(\xi)V_\xi(\xi, 1) - \alpha_{CD}W_{CD}(\xi)V(\xi, 1) = F_\eta(\xi, 1), \quad (72b)$$

where

$$W_{CD}(\xi) = \frac{1}{K_{yCD}(\xi)}, \quad T_{rCD}(\xi) = W_{CD}(\xi)F_{rCD}(\xi) = W_{CD}(\xi)u_{rCD}(\mathbf{x}(\xi, 1)), \quad (73)$$

and  $S_{CD}(\xi)$  and  $K_{yCD}(\xi)$  are defined in (43). The objective here is to formulate  $V(\xi, \eta)$  to exactly satisfy the conditions (14a), (14d), (65) and (72).

The boundary conditions must be compatible at the shared vertices. The Robin condition (65) on  $\overline{BC}$  and the Dirichlet condition (14a) on  $\overline{AB}$  should be compatible at vertex  $B$ , leading to the conditions (21a), (21c), and (67). Similarly, at vertex  $D$  the Robin condition (72) on  $\overline{CD}$  and the Dirichlet condition (14d) on  $\overline{AD}$  should be compatible with each other, inducing the conditions (44a) and (44b), together with

$$\begin{aligned} V_\xi(-1, 1) &= \frac{1}{S_{CD}(-1)} [T_{rCD}(-1) - F_\eta(-1, 1) - \alpha_{CD}W_{CD}(-1)F(-1, 1)] \\ &= F_\xi^a(-1, 1), \end{aligned} \quad \text{if } \overline{AD} \not\perp \overline{CD} \text{ at } D, \quad (74a)$$

$$F_\eta(-1, 1) + \alpha_{CD}W_{CD}(-1)F(-1, 1) - T_{rCD}(-1) = 0, \quad \text{if } \overline{AD} \perp \overline{CD} \text{ at } D, \quad (74b)$$

by noting that  $S_{CD}(-1) = 0$  if  $\overline{AD} \perp \overline{CD}$  at  $D$ .

The Robin conditions (65) and (72) should be compatible at the common vertex  $C$ . This leads to

$$V_\xi(1, 1) + S_{BC}(1)V_\eta(1, 1) = T_{rBC}(1) - \alpha_{BC}W_{BC}(1)V(1, 1), \quad (75a)$$

$$S_{CD}(1)V_\xi(1, 1) + V_\eta(1, 1) = T_{rCD}(1) - \alpha_{CD}W_{CD}(1)V(1, 1). \quad (75b)$$

It follows that

$$V_\xi(1, 1) = F_\xi^a(1, 1) - F_\xi^b(1, 1)V(1, 1) = F_\xi(1, 1), \quad (76a)$$

$$V_\eta(1, 1) = F_\eta^a(1, 1) - F_\eta^b(1, 1)V(1, 1) = F_\eta(1, 1), \quad (76b)$$

where

$$\begin{bmatrix} F_\xi^a(1, 1) \\ F_\eta^a(1, 1) \end{bmatrix} = \frac{1}{1 - S_{BC}(1)S_{CD}(1)} \begin{bmatrix} T_{rBC}(1) - S_{BC}(1)T_{rCD}(1) \\ T_{rCD}(1) - S_{CD}(1)T_{rBC}(1) \end{bmatrix}, \quad (77a)$$

$$\begin{bmatrix} F_\xi^b(1, 1) \\ F_\eta^b(1, 1) \end{bmatrix} = \frac{1}{1 - S_{BC}(1)S_{CD}(1)} \begin{bmatrix} \alpha_{BC}W_{BC}(1) - S_{BC}(1)\alpha_{CD}W_{CD}(1) \\ \alpha_{CD}W_{CD}(1) - S_{CD}(1)\alpha_{BC}W_{BC}(1) \end{bmatrix}. \quad (77b)$$

Differentiating (65b) w.r.t.  $\eta$  and evaluating it at vertex  $C$ , we get

$$V_{\xi\eta}(1, 1) = Q_{BC}^a(1) - S_{BC}(1)V_{\eta\eta}(1, 1) + Q_{BC}^b(1)V(1, 1) = F_{\xi\eta}(1, 1), \quad (78)$$

where (76b) has been used, and

$$Q_{BC}^a(1) = T'_{rBC}(1) - [S'_{BC}(1) + \alpha_{BC}W_{BC}(1)]F_\eta^a(1, 1), \quad (79a)$$

$$Q_{BC}^b(1) = [S'_{BC}(1) + \alpha_{BC}W_{BC}(1)]F_\eta^b(1, 1) - \alpha_{BC}W'_{BC}(1). \quad (79b)$$

Similarly, differentiating (72b) w.r.t.  $\xi$  and evaluating it at vertex  $C$  result in

$$V_{\eta\xi}(1, 1) = Q_{CD}^a(1) - S_{CD}(1)V_{\xi\xi}(1, 1) + Q_{CD}^b(1)V(1, 1) = F_{\eta\xi}(1, 1), \quad (80)$$

where (76a) has been used and

$$Q_{CD}^a(1) = T'_{rCD}(1) - [S'_{CD}(1) + \alpha_{CD}W_{CD}(1)]F_\xi^a(1, 1), \quad (81a)$$

$$Q_{CD}^b(1) = [S'_{CD}(1) + \alpha_{CD}W_{CD}(1)]F_\xi^b(1, 1) - \alpha_{CD}W'_{CD}(1). \quad (81b)$$

Combining (78) and (80) and requiring that  $V_{\xi\eta}(1, 1) = V_{\eta\xi}(1, 1)$ , we have the following constraints,

$$\begin{aligned} V_{\xi\xi}(1, 1) &= \frac{S_{BC}(1)}{S_{CD}(1)}V_{\eta\eta}(1, 1) - \frac{1}{S_{CD}(1)}[Q_{BC}^b(1) - Q_{CD}^b(1)]V(1, 1) \\ &\quad - \frac{1}{S_{CD}(1)}[Q_{BC}^a(1) - Q_{CD}^a(1)] = F_{\xi\xi}(1, 1), \quad \text{if } \overline{BC} \not\perp \overline{CD} \text{ at } C; \end{aligned} \quad (82a)$$

$$V(1, 1) = -\frac{Q_{BC}^a(1) - Q_{CD}^a(1)}{Q_{BC}^b(1) - Q_{CD}^b(1)} = F^a(1, 1), \quad \text{if } \overline{BC} \perp \overline{CD} \text{ at } C \text{ and } Q_{BC}^b(1) \neq Q_{CD}^b(1); \quad (82b)$$

$$Q_{BC}^a(1) = Q_{CD}^a(1), \quad \text{if } \overline{BC} \perp \overline{CD} \text{ at } C \text{ and } Q_{BC}^b(1) = Q_{CD}^b(1). \quad (82c)$$

Eq. (82c) is a constraint on the prescribed Robin boundary data  $u_{rBC}$  and  $u_{rCD}$  when  $\overline{BC} \perp \overline{CD}$  at vertex  $C$  and  $Q_{BC}^b(1) = Q_{CD}^b(1)$ , while otherwise (82b) is a constraint on the value  $V(1, 1)$  and (82a) imposes a relation on  $V_{\xi\xi}(1, 1)$ ,  $V_{\eta\eta}(1, 1)$  and  $V(1, 1)$ .

The Eqs. (14a), (14d), (65b), (72b), (21a), (67a), (44a), (74a), (76), (82a)–(82b), and (78) constitute the set of constraints the function  $V(\xi, \eta)$  to be formulated must satisfy. In addition to the flags  $\lambda_B$ ,  $\lambda_C$  and  $\lambda_D$  introduced previously, we define another constant  $\gamma_C$  to flag whether  $Q_{BC}^b(1) = Q_{CD}^b(1)$ ,

$$\gamma_C = \begin{cases} 0, & \text{if } Q_{BC}^b(1) = Q_{CD}^b(1), \\ 1, & \text{if } Q_{BC}^b(1) \neq Q_{CD}^b(1). \end{cases} \quad (83)$$

We follow the four-step procedure described in Section 2.3.1 to formulate  $V(\xi, \eta)$ . To handle the conditions (21a), (67a), (44a), (74a), (76a)–(76b), and (82a)–(82b), we introduce

$$F(1, \eta) = F(1, -1)\rho_0(\eta) + \lambda_B F_\eta(1, -1)v_0(\eta) + (1 - \lambda_C)\gamma_C F(1, 1)\rho_1(\eta) + F_\eta(1, 1)v_1(\eta), \quad (84a)$$

$$F(\xi, 1) = F(-1, 1)\rho_0(\xi) + \lambda_D F_\xi(-1, 1)v_0(\xi) + (1 - \lambda_C)\gamma_C F(1, 1)\rho_1(\xi) + F_\xi(1, 1)v_1(\xi) \\ + \lambda_C F_{\xi\xi}(1, 1)\omega_1(\xi), \quad (84b)$$

where  $F_\eta(1, 1)$ ,  $F_\xi(1, 1)$  and  $F_{\xi\xi}(1, 1)$  are defined in (76) and (82a), and

$$F_\eta(1, -1) = F_\eta^a(1, -1), \quad F_\xi(-1, 1) = F_\xi^a(-1, 1), \quad F(1, 1) = F^a(1, 1), \quad (85)$$

with  $F_\eta^a(1, -1)$ ,  $F_\xi^a(-1, 1)$  and  $F^a(1, 1)$  defined in (67a), (74a) and (82b). Then the construction problem becomes the following: find  $V(\xi, \eta)$  such that

$$V(-1, \eta) = F(-1, \eta), \quad (86a)$$

$$V(1, \eta) = F(1, \eta) = F(1, -1)\rho_0(\eta) + \lambda_B F_\eta(1, -1)v_0(\eta) + (1 - \lambda_C)\gamma_C F(1, 1)\rho_1(\eta) \\ + F_\eta(1, 1)v_1(\eta), \quad (86b)$$

$$V_\xi(1, \eta) = F_\xi(1, \eta), \quad (86c)$$

$$V(\xi, -1) = F(\xi, -1), \quad (86d)$$

$$V(\xi, 1) = F(\xi, 1) = F(-1, 1)\rho_0(\xi) + \lambda_D F_\xi(-1, 1)v_0(\xi) + (1 - \lambda_C)\gamma_C F(1, 1)\rho_1(\xi) \\ + F_\xi(1, 1)v_1(\xi) + \lambda_C F_{\xi\xi}(1, 1)\omega_1(\xi), \quad (86e)$$

$$V_\eta(\xi, 1) = F_\eta(\xi, 1), \quad (86f)$$

$$V_{\xi\eta}(1, 1) = F_{\xi\eta}(1, 1), \quad (86g)$$

where  $F_\xi(1, \eta)$ ,  $F_\eta(\xi, 1)$  and  $F_{\xi\eta}(1, 1)$  are defined in (65b), (72b), and (78), respectively.

We define the transfinite interpolation

$$PF(\xi, \eta) = F(-1, \eta)\rho_0(\xi) + F_\xi(1, \eta)v_1(\xi) + F(\xi, -1)\rho_0(\eta) + F_\eta(\xi, 1)v_1(\eta) \\ - [F(-1, -1)\rho_0(\eta) + F_\eta(-1, 1)v_1(\eta)] \rho_0(\xi) - [F_\xi(1, -1)\rho_0(\eta) + F_{\xi\eta}(1, 1)v_1(\eta)] v_1(\xi) \\ + [\lambda_D F_\xi(-1, 1)v_0(\xi) + \lambda_C F_{\xi\xi}(1, 1)\omega_1(\xi) + (1 - \lambda_C)\gamma_C F(1, 1)\rho_1(\xi)] \rho_1(\eta) \\ + \lambda_B F_\eta(1, -1)v_0(\eta)\rho_1(\xi). \quad (87)$$

One can verify that the function  $V(\xi, \eta) = PF(\xi, \eta)$  satisfies the conditions in (86) exactly.

The preliminary general form for  $V(\xi, \eta)$  is then given by,

$$V(\xi, \eta) = g(\xi, \eta) - Pg(\xi, \eta) + PF(\xi, \eta), \quad (88)$$

where  $g(\xi, \eta)$  is a free (arbitrary) function, and

$$Pg(\xi, \eta) = g(-1, \eta)\rho_0(\xi) + g_\xi(1, \eta)v_1(\xi) + g(\xi, -1)\rho_0(\eta) + g_\eta(\xi, 1)v_1(\eta) \\ - [g(-1, -1)\rho_0(\eta) + g_\eta(-1, 1)v_1(\eta)] \rho_0(\xi) - [g_\xi(1, -1)\rho_0(\eta) + g_{\xi\eta}(1, 1)v_1(\eta)] v_1(\xi) \\ + [\lambda_D g_\xi(-1, 1)v_0(\xi) + \lambda_C g_{\xi\xi}(1, 1)\omega_1(\xi) + (1 - \lambda_C)\gamma_C g(1, 1)\rho_1(\xi)] \rho_1(\eta) \\ + \lambda_B g_\eta(1, -1)v_0(\eta)\rho_1(\xi). \quad (89)$$

The modified transfinite interpolation is,

$$PF^g(\xi, \eta) = F(-1, \eta)\rho_0(\xi) + F_\xi^g(1, \eta)v_1(\xi) + F(\xi, -1)\rho_0(\eta) + F_\eta^g(\xi, 1)v_1(\eta) \\ - [F(-1, -1)\rho_0(\eta) + F_\eta(-1, 1)v_1(\eta)] \rho_0(\xi) - [F_\xi(1, -1)\rho_0(\eta) + F_{\xi\eta}^g(1, 1)v_1(\eta)] v_1(\xi) \\ + [\lambda_D F_\xi^a(-1, 1)v_0(\xi) + \lambda_C F_{\xi\xi}^g(1, 1)\omega_1(\xi) + (1 - \lambda_C)\gamma_C F^a(1, 1)\rho_1(\xi)] \rho_1(\eta) \\ + \lambda_B F_\eta^a(1, -1)v_0(\eta)\rho_1(\xi). \quad (90)$$

Here  $F_{\xi}^a(-1, 1)$  and  $F_{\eta}^a(1, -1)$  are given in (74a) and (67a), and  $F^a(1, 1)$  is given in (82b). In addition,

$$F_{\xi\eta}^g(1, 1) = Q_{BC}^a(1) - S_{BC}(1)g_{\eta\eta}(1, 1) + Q_{BC}^b(1) \left[ (1 - (1 - \lambda_C)\gamma_C) g(1, 1) + (1 - \lambda_C)\gamma_C F^a(1, 1) \right]; \quad (91a)$$

$$F_{\xi\xi}^g(1, 1) = \frac{S_{BC}(1)}{S_{CD}(1)} g_{\eta\eta}(1, 1) - \frac{1}{S_{CD}(1)} \left[ Q_{BC}^b(1) - Q_{CD}^b(1) \right] \left[ (1 - (1 - \lambda_C)\gamma_C) g(1, 1) + (1 - \lambda_C)\gamma_C F^a(1, 1) \right] - \frac{1}{S_{CD}(1)} \left[ Q_{BC}^a(1) - Q_{CD}^a(1) \right]; \quad (91b)$$

$$F_{\xi}^g(1, \eta) = T_{rBC}(\eta) - S_{BC}(\eta)V_{\eta}^g(1, \eta) - \alpha_{BC}W_{BC}(\eta)V^g(1, \eta); \quad (91c)$$

$$F_{\eta}^g(\xi, 1) = T_{rCD}(\xi) - S_{CD}V_{\xi}^g(\xi, 1) - \alpha_{CD}W_{CD}(\xi)V^g(\xi, 1); \quad (91d)$$

$$V^g(1, \eta) = g(1, \eta) - [g(1, -1) - F(1, -1)] \rho_0(\eta) - \left[ g_{\eta}(1, 1) - F_{\eta}^g(1, 1) \right] v_1(\eta) - \lambda_B \left[ g_{\eta}(1, -1) - F_{\eta}^a(1, -1) \right] v_0(\eta) - (1 - \lambda_C)\gamma_C \left[ g(1, 1) - F^a(1, 1) \right] \rho_1(\eta); \quad (91e)$$

$$V_{\eta}^g(1, \eta) = g_{\eta}(1, \eta) - [g(1, -1) - F(1, -1)] \rho'_0(\eta) - \left[ g_{\eta}(1, 1) - F_{\eta}^g(1, 1) \right] v'_1(\eta) - \lambda_B \left[ g_{\eta}(1, -1) - F_{\eta}^a(1, -1) \right] v'_0(\eta) - (1 - \lambda_C)\gamma_C \left[ g(1, 1) - F^a(1, 1) \right] \rho'_1(\eta); \quad (91f)$$

$$V^g(\xi, 1) = g(\xi, 1) - [g(-1, 1) - F(-1, 1)] \rho_0(\xi) - \left[ g_{\xi}(1, 1) - F_{\xi}^g(1, 1) \right] v_1(\xi) - \lambda_D \left[ g_{\xi}(-1, 1) - F_{\xi}^a(-1, 1) \right] v_0(\xi) - \lambda_C \left[ g_{\xi\xi}(1, 1) - F_{\xi\xi}^g(1, 1) \right] \omega_1(\xi) - (1 - \lambda_C)\gamma_C \left[ g(1, 1) - F^a(1, 1) \right] \rho_1(\xi); \quad (91g)$$

$$V_{\xi}^g(\xi, 1) = g_{\xi}(\xi, 1) - [g(-1, 1) - F(-1, 1)] \rho'_0(\xi) - \left[ g_{\xi}(1, 1) - F_{\xi}^g(1, 1) \right] v'_1(\xi) - \lambda_D \left[ g_{\xi}(-1, 1) - F_{\xi}^a(-1, 1) \right] v'_0(\xi) - \lambda_C \left[ g_{\xi\xi}(1, 1) - F_{\xi\xi}^g(1, 1) \right] \omega'_1(\xi) - (1 - \lambda_C)\gamma_C \left[ g(1, 1) - F^a(1, 1) \right] \rho'_1(\xi); \quad (91h)$$

$$F_{\eta}^g(1, 1) = F_{\eta}^a(1, 1) - F_{\eta}^b(1, 1) \left[ (1 - (1 - \lambda_C)\gamma_C) g(1, 1) + (1 - \lambda_C)\gamma_C F^a(1, 1) \right]; \quad (91i)$$

$$F_{\xi}^g(1, 1) = F_{\xi}^a(1, 1) - F_{\xi}^b(1, 1) \left[ (1 - (1 - \lambda_C)\gamma_C) g(1, 1) + (1 - \lambda_C)\gamma_C F^a(1, 1) \right]. \quad (91j)$$

This gives rise to the final form for  $V(\xi, \eta)$ ,

$$V(\xi, \eta) = g(\xi, \eta) - Pg(\xi, \eta) + PF^g(\xi, \eta), \quad (92)$$

where  $g(\xi, \eta)$  is a free (arbitrary) function,  $Pg(\xi, \eta)$  is given by (89), and  $PF^g(\xi, \eta)$  is given by (90).

**Theorem 2.4.**  $V(\xi, \eta)$  given by (92) satisfies the Dirichlet conditions (14a) and (14d) and the Robin conditions (65a) and (72a), for any  $g(\xi, \eta)$  therein that is sufficiently differentiable,

**Proof.** By verification. The verification process relies on the following relations, but is otherwise straightforward albeit a little cumbersome. These relations are,

$$\lim_{\xi \rightarrow -1} F_{\eta}^g(\xi, 1) = F_{\eta}^g(-1, 1) = F_{\eta}(-1, 1) = \lim_{\eta \rightarrow 1} F_{\eta}(-1, \eta), \quad (93a)$$

$$\lim_{\eta \rightarrow -1} F_{\xi}^g(1, \eta) = F_{\xi}^g(1, -1) = F_{\xi}(1, -1) = \lim_{\xi \rightarrow 1} F_{\xi}(\xi, -1), \quad (93b)$$

$$\lim_{\eta \rightarrow 1} F_{\xi}^g(1, \eta) = F_{\xi}^g(1, 1), \quad (93c)$$

$$\lim_{\xi \rightarrow 1} F_{\eta}^g(\xi, 1) = F_{\eta}^g(1, 1), \quad (93d)$$

$$\lim_{\xi \rightarrow 1} F_{\eta,\xi}^g(\xi, 1) = F_{\eta,\xi}^g(1, 1) = F_{\xi\eta}^g(1, 1), \quad (93e)$$

$$\lim_{\eta \rightarrow 1} F_{\xi,\eta}^g(1, \eta) = F_{\xi,\eta}^g(1, 1) = F_{\xi\eta}^g(1, 1), \quad (93f)$$

where  $F_{\xi}^g(1, \eta)$  and  $F_{\eta}^g(\xi, 1)$  are defined in (91c) and (91d),  $F_{\eta}^g(1, 1)$  and  $F_{\xi}^g(1, 1)$  are defined in (91i) and (91j), and  $F_{\xi\eta}^g(1, 1)$  is defined in (91a). These relations can be verified to be true by considering different cases such as whether adjacent boundaries are orthogonal to each other or not at a vertex.  $\square$

**Remark 2.7.** When the domain involves more than two Robin boundaries, at every vertex where two Robin boundaries meet, the compatibility constraints analogous to the aforementioned ones will apply. At every vertex where a Robin boundary and a Dirichlet boundary meet, the compatibility constraints analogous to those discussed in Section 2.4.1 will apply. The field function satisfying these boundary conditions can be formulated in an analogous way based on the four-step procedure.

**Remark 2.8.** When the domain involves a combination of Dirichlet, Neumann, and Robin boundaries, at every vertex where a Robin boundary and a Neumann boundary meet, the compatibility constraints as discussed above for two Robin boundaries will apply. In this case, the Neumann boundary involved in can be treated as a Robin one with the Robin coefficient  $\alpha$  set to zero.

## 2.5. Enforcing Dirichlet/Neumann/Robin BCs exactly for solving PDEs by extreme learning machine

Let us now assume that the function formulated in Sections 2.2, 2.3 and 2.4 represents the unknown solution field to some given PDE. Therefore, the Dirichlet, Neumann and Robin boundary conditions involved in the PDE problem will be automatically and exactly satisfied. Since the free function  $g(\xi, \eta)$  can be arbitrary, one can choose a function space or use some nonlinear representation such as artificial neural networks for  $g(\xi, \eta)$  to satisfy the PDE, thus giving rise to a specific numerical method. It should be noted that, regardless of the representation or the numerical method for computing  $g(\xi, \eta)$ , the Dirichlet/Neumann/Robin boundary conditions involved in the problem, by formulation, are exactly satisfied.

In this paper we employ the physics informed approach, and a type of randomized neural networks known as extreme learning machines (ELMs), for representing the free function  $g(\xi, \eta)$  to compute the PDE solution. We refer to [65,73,75,76,92] for more details on ELM for scientific machine learning. In the following discussion we use a second-order linear boundary value problem (BVP) to illustrate the ELM technique together with the current method for BC enforcement. A discussion on nonlinear problems with the current method is provided in a remark at the end of this section.

Specifically, we consider a general quadrilateral domain  $\Omega = \overline{ABCD}$  as illustrated in Fig. 1(a) and the following BVP on this domain,

$$\mathcal{L}u(\mathbf{x}) = f(\mathbf{x}), \quad (94a)$$

$$Bu(\mathbf{x})|_{\mathbf{x} \in \partial\Omega} = f_b(\mathbf{x}), \quad (94b)$$

where  $\mathcal{L}$  is a second-order linear differential operator,  $u(\mathbf{x})$  is the unknown field to be solved for, and  $f(\mathbf{x})$  and  $f_b(\mathbf{x})$  represent prescribed source terms.  $B$  is the boundary operator, and  $Bu(\mathbf{x})$  represents a set of Dirichlet, Neumann, or Robin type conditions imposed on different domain boundaries. Since  $u(\mathbf{x})$  needs to satisfy the second-order PDE, we assume in this section that each boundary curve of the domain ( $\mathbf{x}_{AB}(\xi)$ ,  $\mathbf{x}_{BC}(\eta)$ ,  $\mathbf{x}_{CD}(\xi)$ ,  $\mathbf{x}_{AD}(\eta)$ ) should be sufficiently differentiable.

Since the Dirichlet/Neumann/Robin conditions of (94b) are exactly enforced by formulation, we only need to focus on the PDE (94a). Employing the map  $\mathbf{x}(\xi, \eta)$ , we transform (94a) into

$$\mathcal{L}V(\xi, \eta) = f(\mathbf{x}(\xi, \eta)) = f_a(\xi, \eta), \quad (\xi, \eta) \in \Omega_{st}, \quad (95)$$

where  $V(\xi, \eta)$  is the transformed field function and is related to  $u(\mathbf{x})$  by (12). It should be noted that  $\mathcal{L}$  is a differential operator defined on the physical domain (with respect to  $\mathbf{x} = (x, y)$ ), while  $V(\xi, \eta)$  is formulated in terms of the standard domain (with respect to  $(\xi, \eta)$ ); see Remark 2.9 below for a discussion on the computation of associated terms. The formulations of  $V(\xi, \eta)$  from previous sections all have the following form,

$$V(\xi, \eta) = g(\xi, \eta) - Pg(\xi, \eta) + PF^g(\xi, \eta). \quad (96)$$

We rewrite the transfinite interpolation therein into two components,

$$PF^g(\xi, \eta) = PF^{gb}(\xi, \eta) + PF^a(\xi, \eta), \quad (97)$$

where  $PF^{gb}(\xi, \eta)$  denotes all the terms in  $PF^g(\xi, \eta)$  that involve the free function  $g(\xi, \eta)$ , and  $PF^a(\xi, \eta)$  denotes the rest of the terms. Note that  $PF^{gb}$  is linear with respect to  $g$ , and that  $PF^{gb} = 0$  if the domain involves only Dirichlet boundaries. Eq. (95) then becomes

$$\mathcal{L}V^{gb}(\xi, \eta) = f_a(\xi, \eta) - \mathcal{L}(PF^a)(\xi, \eta), \quad (98)$$

where

$$V^{gb}(\xi, \eta) = g(\xi, \eta) - Pg(\xi, \eta) + PF^{gb}(\xi, \eta). \quad (99)$$

To represent  $g(\xi, \eta)$ , we employ a randomized feedforward neural network, whose structure is characterized by an architectural vector  $\mathbf{m} = [m_0, m_1, \dots, m_L]$ . Here  $(L+1)$  (with  $L \geq 2$ ) is the depth of the network, and  $m_i$  denotes the number of nodes in the  $i$ th layer. Layer 0, with  $m_0 = 2$ , represents the input  $(\xi, \eta)$ , and the last layer, with  $m_L = 1$ , represents the output  $g(\xi, \eta)$ . Those layers in between are the hidden layers. Following the ELM convention [73,74], we assign the hidden-layer coefficients (weights/biases) by random values generated on the interval  $[-R_m, R_m]$ , where  $R_m$  is a user-prescribed constant, from a uniform distribution. Once the hidden-layer coefficients are randomly assigned, they are fixed throughout the computation. In addition, we require that the output layer contains no activation, or equivalently with the activation function  $\sigma(x) = x$ , and has zero bias. In ELM the hidden-layer coefficients are randomly assigned and non-trainable, and only the output-layer coefficients are trained [73,93].

Then the NN logic of the output layer yields the following relation,

$$g(\xi, \eta) = \sum_{j=1}^M \beta_j \varphi_j(\xi, \eta) = \Phi(\xi, \eta)\beta. \quad (100)$$

Here  $M = m_{L-1}$  is the width of the last hidden layer,  $\Phi(\xi, \eta) = (\varphi_1, \dots, \varphi_M)$  denotes the set of output fields of the last hidden layer, and  $\beta = (\beta_1, \dots, \beta_M)^T$  denotes the output-layer coefficients, which constitute the set of trainable parameters in ELM.

We train the ELM network to solve Eq. (98) based on a physics informed approach. By choosing a set of  $Q_c$  collocation points from the interior of the standard domain,  $(\xi_i, \eta_i) \in \Omega_{st}$  for  $1 \leq i \leq Q_c$ , and enforcing Eq. (98) on these collocation points, we have

$$\left[ \mathcal{L}\Phi|_{(\xi_i, \eta_i)} - \mathcal{L}(P\Phi)|_{(\xi_i, \eta_i)} + \mathcal{L}(PF\Phi^b)|_{(\xi_i, \eta_i)} \right] \beta = f_a(\xi_i, \eta_i) - \mathcal{L}(PF^a)|_{(\xi_i, \eta_i)}, \quad 1 \leq i \leq Q_c, \quad (101)$$

where we have used (100). In this equation  $P\Phi = (P\varphi_1, \dots, P\varphi_M)$ , and  $P\varphi_i(\xi, \eta)$  is defined in the same manner as  $Pg(\xi, \eta)$ .  $PF\Phi^b = (PF\varphi_1^b, \dots, PF\varphi_M^b)$ , and  $PF\varphi_i^b(\xi, \eta)$  is defined in the same manner as  $PFg^b(\xi, \eta)$ . Eq. (101) constitutes a rectangular system of linear algebraic equations about  $\beta$ , with  $Q_c$  equations and  $M$  unknowns. We seek a least squares solution and solve this system by the linear least squares method [79]. The output-layer coefficients of the ELM network are then set by this least squares solution for  $\beta$ , completing the NN training. The final solution field to the boundary value problem (94) is computed based on (12) and (96).

**Remark 2.9.** When implementing ELM for (101), one encounters derivatives like  $\frac{\partial \psi}{\partial x}$ ,  $\frac{\partial \psi}{\partial y}$ ,  $\frac{\partial^2 \psi}{\partial x^2}$ , and  $\frac{\partial^2 \psi}{\partial y^2}$ , where  $\psi$  is a function defined on the standard domain  $\Omega_{st}$ , i.e.  $\psi = \psi(\xi, \eta)$ . These terms can be computed using the Jacobian matrix  $\mathbf{J}(\xi, \eta)$  defined in (19) as follows,

$$\begin{bmatrix} \psi_x & \psi_y \end{bmatrix} = \begin{bmatrix} \psi_\xi & \psi_\eta \end{bmatrix} \mathbf{J}^{-1}(\xi, \eta), \quad \mathbf{C} = \begin{bmatrix} \psi_x & \psi_y \end{bmatrix} \frac{\partial \mathbf{J}}{\partial \xi}, \quad \mathbf{D} = \begin{bmatrix} \psi_x & \psi_y \end{bmatrix} \frac{\partial \mathbf{J}}{\partial \eta}, \quad (102a)$$

$$\begin{bmatrix} \psi_{xx} & \psi_{xy} \\ \psi_{xy} & \psi_{yy} \end{bmatrix} = \mathbf{J}^{-T} \left( \begin{bmatrix} \psi_{\xi\xi} & \psi_{\xi\eta} \\ \psi_{\xi\eta} & \psi_{\eta\eta} \end{bmatrix} - [\mathbf{C}^T \quad \mathbf{D}^T] \right) \mathbf{J}^{-1}, \quad (102b)$$

where the terms  $\psi_\xi$ ,  $\psi_\eta$ ,  $\psi_{\xi\xi}$ ,  $\psi_{\xi\eta}$  and  $\psi_{\eta\eta}$  can be computed either directly or by automatic differentiations of the neural network.

**Remark 2.10.** After the NN training is complete, when using the form (96) to evaluate the solution field on the boundary test points, we have observed from numerical simulations that the cancellation error, due to subtraction of nearly equal real numbers in the terms like  $(g - Pg)$ , can be notable at isolated boundary points for some problems. For example, when evaluating the boundary-condition errors using the numerically attained solution, this can lead to errors on the order around  $10^{-10} \sim 10^{-9}$  at isolated boundary points, instead of the error levels such as  $10^{-16}$  or lower for the other boundary points. We find that a combination of the following two measures in implementation can reduce the cancellation error significantly:

- Restructure the computation for terms like  $(g - Pg)$  and  $PF$ . For example, implementing the terms in (15)–(16b) using the following equivalent forms essentially eliminates the cancellation error,

$$\begin{aligned} g(\xi, \eta) - Pg(\xi, \eta) &= [g(\xi, \eta) - g(\xi, -1)\phi_0(\eta) - g(\xi, 1)\phi_1(\eta)] \\ &\quad - [g(-1, \eta) - g(-1, -1)\phi_0(\eta) - g(-1, 1)\phi_1(\eta)] \phi_0(\xi) \\ &\quad - [g(1, \eta) - g(1, -1)\phi_0(\eta) - g(1, 1)\phi_1(\eta)] \phi_1(\xi), \end{aligned} \quad (103a)$$

$$\begin{aligned} PF(\xi, \eta) &= [F(\xi, -1)\phi_0(\eta) + F(\xi, 1)\phi_1(\eta)] \\ &\quad + [F(-1, \eta) - F(-1, -1)\phi_0(\eta) - F(-1, 1)\phi_1(\eta)] \phi_0(\xi) \\ &\quad + [F(1, \eta) - F(1, -1)\phi_0(\eta) - F(1, 1)\phi_1(\eta)] \phi_1(\xi). \end{aligned} \quad (103b)$$

- Introduce a small number of collocation points on the Dirichlet boundaries and enforce  $g(\xi, \eta) = 0$  at those points when training the neural network. Because the  $V(\xi, \eta)$  form, with arbitrary  $g(\xi, \eta)$  therein, satisfies the BC (94b) mathematically, we have only employed the PDE (94a) for NN training to determine  $g(\xi, \eta)$ . The  $g(\xi, \eta)$  determined in such a way is necessarily not unique. In practice, the function values for  $g(\xi, \eta)$  determined in this way can have large magnitudes, exacerbating the aforementioned cancellation error issue. We observe that, by additionally introducing a small number (e.g. 3 or 5) of collocation points on each of the Dirichlet boundary and enforcing  $g(\xi, \eta) = 0$  on these points during NN training, the resultant function values for  $g(\xi, \eta)$  will generally involve much smaller magnitudes. This can notably improve the cancellation error. This measure leads to the following system of equations,

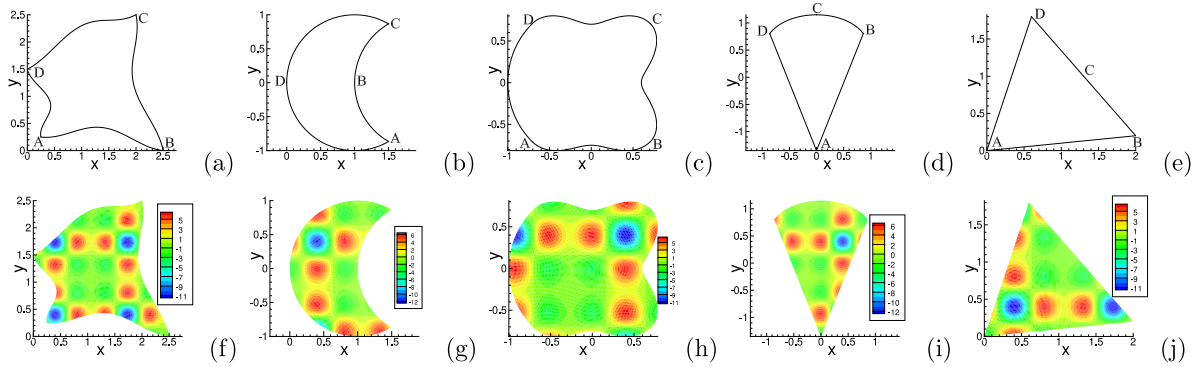
$$\Phi(\xi_j, \eta_j)\beta = 0, \quad (\xi_j, \eta_j) \in \partial\Omega_d, \quad 1 \leq j \leq Q_{db}, \quad (104)$$

where  $\partial\Omega_d$  denotes the Dirichlet boundary of the domain, and  $Q_{db}$  denotes the number of collocation points on the Dirichlet boundaries. As such, the final algebraic system for computing  $\beta$  consists of (101) and (104), with  $N_c = (Q_c + Q_{db})$  equations and  $M$  unknowns. The least squares solution to this augmented system provides the trained output-layer coefficients of the ELM network.

We have incorporated these measures into our implementation of the current method in this work. In the Appendix 5.2 we compare several algorithms for evaluating the TFC expressions to compute the NN solution and discuss the effect of  $Q_{db}$  on the NN-solution errors and the boundary-condition errors.

**Remark 2.11.** Boundary value problems consisting of (94b) (for Dirichlet, Neumann, or Robin conditions) and a nonlinear PDE,

$$\mathcal{L}u(\mathbf{x}) + \mathcal{N}(u) = f(\mathbf{x}), \quad (105)$$



**Fig. 2.** Helmholtz equation: Domain geometries (top row) and distributions (contour plots) of the exact solutions (bottom row), (a,f) domain #1, (b,g) domain #2, (c,h) domain #3, (d,i) domain #4, and (e,j) domain #5. The color bars in the plots denote the solution values. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

where  $\mathcal{N}$  denotes a nonlinear operator, can be solved using the current method and ELM in an analogous fashion. By employing the mapping function and the formulation (96) and enforcing (105) on the chosen collocation points, we get

$$\mathcal{L}V^{gb}(\xi_i, \eta_i) + \mathcal{N}(V^{gb}(\xi_i, \eta_i) + PF^a(\xi_i, \eta_i)) = f_a(\xi_i, \eta_i) - \mathcal{L}(PF^a)(\xi_i, \eta_i), \quad (\xi_i, \eta_i) \in \Omega_{st}, \quad (106)$$

$$1 \leq i \leq Q_c,$$

where  $V^{gb}$  is given by (99) and (100). This is a nonlinear algebraic system about the trainable parameters  $\beta$ , with  $Q_c$  equations and  $M$  unknowns. We seek a least squares solution and solve this system by the nonlinear least squares (Gauss-Newton) method [79], specifically by the NLLSQ-perturb (Nonlinear least squares with perturbations) algorithm from [73]. The output-layer coefficients are then updated by the least squares solution for  $\beta$  to complete the NN training.

### 3. Numerical tests

We next present several numerical examples to test the effectiveness of the method from the previous section for enforcing Dirichlet/Neumann/Robin boundary conditions (DBCs/NBCs/RBCs), using linear and nonlinear PDEs over a number of domains with complex boundary geometries. These include the 2D Helmholtz equation and the nonlinear Helmholtz equation, which are time-independent, and the heat conduction equation over space-time domains with deforming or moving boundaries. The domains involve Dirichlet boundaries, or a combination of Dirichlet boundaries with Neumann or Robin boundaries.

We define the maximum and root-mean-squares (rms) solution errors ( $e_{max}$  and  $e_{rms}$ ) as follows,

$$e_{max}(u) = \max \{ |u(\mathbf{x}_i) - u_{ex}(\mathbf{x}_i)| \}_{i=1}^{N_v}, \quad e_{rms}(u) = \sqrt{\frac{1}{N_v} \sum_{i=1}^{N_v} |u(\mathbf{x}_i) - u_{ex}(\mathbf{x}_i)|^2}, \quad (107)$$

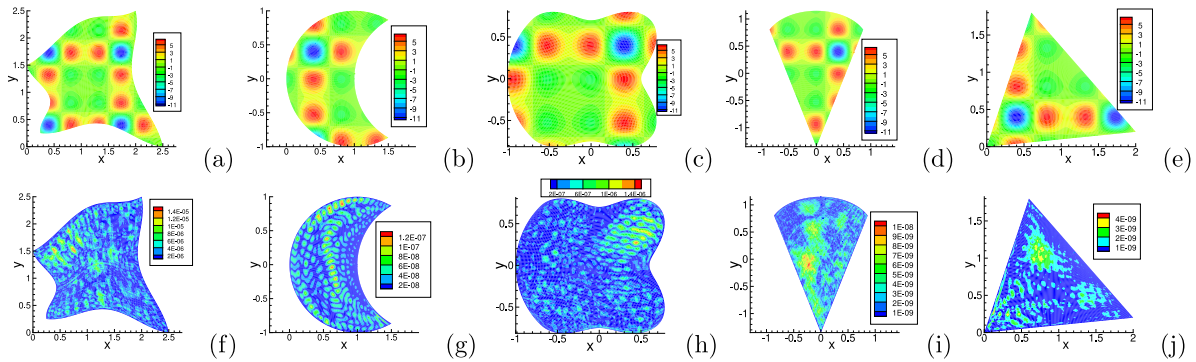
where  $u(\mathbf{x})$  and  $u_{ex}(\mathbf{x})$  denote the NN numerical solution and the exact solution, respectively,  $\mathbf{x}_i$  denotes the test points, and  $N_v$  is the number of test points. By choosing the test points over the entire domain  $\Omega$  or on a specific boundary, we can define the maximum and rms solution errors over the domain ( $e_{max}^\Omega$ ,  $e_{rms}^\Omega$ ) or on the boundaries (e.g.  $e_{max}^{AB}$ ,  $e_{rms}^{AB}$  etc.). The maximum and rms errors for the solution derivatives (e.g.  $e_{max}(\partial u/\partial x)$ ,  $e_{rms}(\partial u/\partial x)$ ,  $e_{max}(\partial u/\partial y)$ ,  $e_{rms}(\partial u/\partial y)$ ) can be defined in an analogous way. For a given quadrilateral domain, if the two adjacent boundaries join smoothly at a vertex (with a common tangent), the Jacobian matrix will become singular at that point. In this case, we exclude such a vertex from the set of test points for computing the maximum/rms errors of the solution derivatives ( $\partial u/\partial x$ ,  $\partial u/\partial y$ ).

In addition, we define the maximum/rms boundary-condition errors (see Eq. (94b)),

$$\varepsilon_{max} = \max \{ |Bu(\mathbf{x}_i) - f_b(\mathbf{x}_i)| \}_{i=1}^{N_b}, \quad \varepsilon_{rms} = \sqrt{\frac{1}{N_b} \sum_{i=1}^{N_b} |Bu(\mathbf{x}_i) - f_b(\mathbf{x}_i)|^2}, \quad (108)$$

where  $\mathbf{x}_i$  denotes the boundary test points and  $N_b$  is the number of such points. By choosing the boundary test points from a specific boundary, we can define the BC errors on specific boundaries (e.g.  $\varepsilon_{max}^{AB}$ ,  $\varepsilon_{rms}^{AB}$  etc.), which can be the errors for Dirichlet, Neumann or Robin conditions imposed there. Unless otherwise specified, we employ  $N_v = 101 \times 101$  test points (uniform grid points in the standard domain  $\Omega_{st}$ ) for computing  $e_{max}^\Omega$  and  $e_{rms}^\Omega$ , and  $N_b = 101$  or  $N_v = 101$  test points (uniform grids on each edge of  $\Omega_{st}$ ) for computing the boundary-condition errors or the boundary solution errors ( $\varepsilon_{max}^{AB}$ ,  $\varepsilon_{rms}^{AB}$ ,  $e_{max}^{AB}$ ,  $e_{rms}^{AB}$  etc.).

In all the numerical simulations of this section, we employ an ELM network architecture  $\mathbf{m} = [2, M, 1]$  for representing the free function  $g(\xi, \eta)$ , where  $M$  is the number of hidden-layer nodes, with the Gaussian activation function  $\sigma(x) = e^{-x^2}$ . The hidden-layer coefficients are assigned to uniform random values generated on  $[-R_m, R_m]$ , with the constant  $R_m$  determined by the differential



**Fig. 3.** Helmholtz equation (Dirichlet BCs on all boundaries): Distributions (contours) of the NN solutions (top row) and their point-wise absolute errors (bottom row) on the five domains. Simulation parameters: Domain #1,  $R_m = 4.62$ ,  $Q = 70$ ,  $M = 800$ ; Domain #2,  $R_m = 4.0$ ,  $Q = 65$ ,  $M = 800$ ; Domain #3,  $R_m = 4.57$ ,  $Q = 65$ ,  $M = 800$ ; Domain #4,  $R_m = 3.53$ ,  $Q = 60$ ,  $M = 800$ ; Domain #5,  $R_m = 4.17$ ,  $Q = 65$ ,  $M = 800$ . The color bars in the top-row plots denote the solution values, and those in the bottom-row plots denote values of the absolute errors. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

evolution algorithm from [75]. We employ  $N_c = (Q \times Q + Q_{db})$  collocation points (i.e.  $Q_c = Q \times Q$ ) for training the ELM. Here  $Q$  is the number of collocation points (uniform grid points) along each direction in the interior of the standard domain, and  $Q_{db}$  is the number of points on the Dirichlet boundaries for enforcing the condition  $g(\xi, \eta) = 0$  as discussed in Remark 2.10. Unless otherwise specified, we employ  $Q_{db} = 16$  if the domain has all Dirichlet boundaries (3 uniform grid points on each boundary plus 4 vertices),  $Q_{db} = 19$  if the domain has a Neumann or Robin boundary with the rest being Dirichlet boundaries (5 uniform points on each Dirichlet boundary plus 4 vertices), and  $Q_{db} = 23$  if the domain has two Neumann boundaries with the rest being Dirichlet boundaries (10 uniform grid points on each Dirichlet boundary plus 3 vertices). The Appendix 5.2 provides a study of the effect of  $Q_{db}$  on the NN solution and boundary-condition accuracies. The values for  $R_m$ ,  $Q$  and  $M$  will be provided in the following discussions. We employ (9) for the domain mapping in the numerical simulations, unless otherwise noted. Our implementation of the method and the neural network is in Python, based on the Tensorflow and Keras libraries. A fixed random seed value 10 has been used for the random number generators in all the numerical simulations.

### 3.1. Helmholtz equation

In the first test we consider the five domains depicted in Fig. 2 (top row) and investigate the boundary value problem with the Helmholtz equation on these domains,

$$\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} - 100u = f(x, y), \quad (109a)$$

$$Bu(x, y)|_{(x, y) \in \partial\Omega} = f_b(x, y), \quad (109b)$$

where  $u(x, y)$  is the unknown field to be computed,  $f$  and  $f_b$  are the source terms, and the boundary operator  $B$  denotes Dirichlet, Neumann, or Robin conditions on different boundaries. The specific geometric parameters for these domains (boundary curves, vertices) are provided in the Appendix 5.1. We choose the source terms appropriately such that the problem has the following exact solution for different boundary conditions,

$$u(x, y) = - \left[ 2 \cos \left( \frac{3}{2} \pi x + \frac{2}{5} \pi \right) + \frac{3}{2} \cos \left( 3 \pi x - \frac{\pi}{5} \right) \right] \left[ 2 \cos \left( \frac{3}{2} \pi y + \frac{2}{5} \pi \right) + \frac{3}{2} \cos \left( 3 \pi y - \frac{\pi}{5} \right) \right]. \quad (110)$$

Distributions of the exact solution on different domains are shown in Fig. 2 (bottom row).

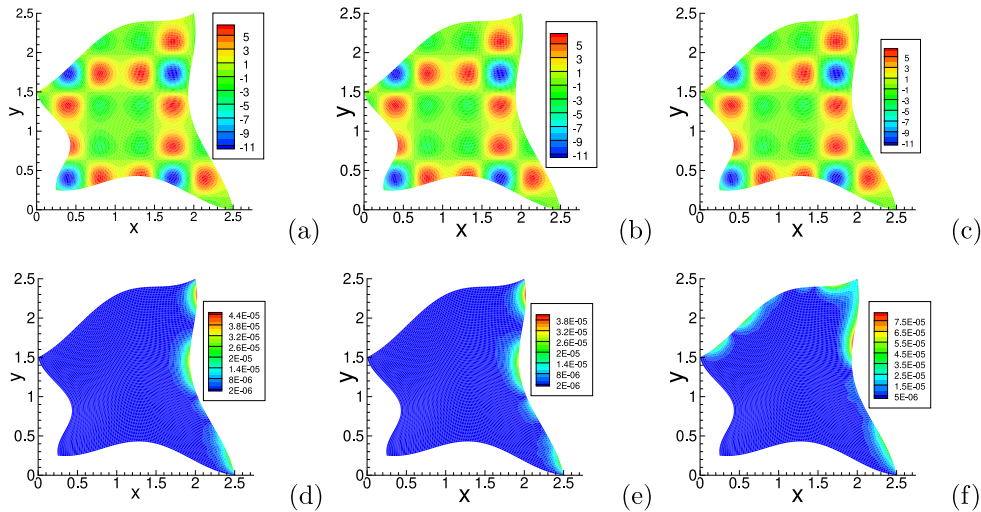
We first consider Dirichlet conditions for all boundaries of these domains. Distributions of the ELM solution over the five domains are shown in Fig. 3 (top row), and their point-wise absolute errors are also included (bottom row). The values for the simulation parameters  $R_m$ ,  $Q$  and  $M$  are provided in the figure caption. The error levels of the ELM solution differ on different domains, with the maximum error generally ranging from  $10^{-9}$  to  $10^{-5}$  in these simulations.

The boundary-condition errors of the ELM solution for the five domains are illustrated in Table 3. This table lists the maximum and rms DBC errors ( $\epsilon_{max}$ ,  $\epsilon_{rms}$ ) on different boundaries ( $AB$ ,  $BC$ ,  $CD$ ,  $AD$ ), together with the maximum/rms errors of the NN solution and its derivatives ( $u$ ,  $\partial u / \partial x$ ,  $\partial u / \partial y$ ) over the domains. The simulation parameters for this table follow those of Fig. 3. It is evident that the current method has enforced the Dirichlet BCs on these domain geometries to the machine accuracy. The error of the solution derivatives is approximately one to two orders of magnitude higher than that of the NN solution.

**Table 3**

Helmholtz equation (Dirichlet BC on all boundaries): maximum and rms errors ( $\epsilon_{max}^\Omega, \epsilon_{rms}^\Omega$ ) of the NN solution and its derivatives ( $u, \partial u/\partial x, \partial u/\partial y$ ) over the domain, and the maximum and rms DBC errors on the four boundaries ( $\epsilon_{max}^{AB}, \epsilon_{rms}^{AB}, \epsilon_{max}^{BC}, \epsilon_{rms}^{BC}, \epsilon_{max}^{CD}, \epsilon_{rms}^{CD}, \epsilon_{max}^{AD}, \epsilon_{rms}^{AD}$ ). Simulation parameters follow those of Fig. 3.

|                                                     | Domain #1               | Domain #2               | Domain #3               | Domain #4               | Domain #5               |
|-----------------------------------------------------|-------------------------|-------------------------|-------------------------|-------------------------|-------------------------|
| max-error- $u$ (domain)                             | $1.622 \times 10^{-5}$  | $1.298 \times 10^{-7}$  | $1.463 \times 10^{-6}$  | $1.122 \times 10^{-8}$  | $4.443 \times 10^{-9}$  |
| rms-error- $u$ (domain)                             | $3.297 \times 10^{-6}$  | $2.987 \times 10^{-8}$  | $2.776 \times 10^{-7}$  | $3.038 \times 10^{-9}$  | $1.060 \times 10^{-9}$  |
| max-error- $\frac{\partial u}{\partial x}$ (domain) | $3.515 \times 10^{-3}$  | $7.239 \times 10^{-5}$  | $9.867 \times 10^{-4}$  | $1.415 \times 10^{-6}$  | $3.475 \times 10^{-6}$  |
| rms-error- $\frac{\partial u}{\partial x}$ (domain) | $2.265 \times 10^{-4}$  | $5.400 \times 10^{-6}$  | $4.600 \times 10^{-5}$  | $1.172 \times 10^{-7}$  | $2.320 \times 10^{-7}$  |
| max-error- $\frac{\partial u}{\partial y}$ (domain) | $3.169 \times 10^{-3}$  | $6.733 \times 10^{-5}$  | $1.318 \times 10^{-3}$  | $1.589 \times 10^{-6}$  | $1.783 \times 10^{-6}$  |
| rms-error- $\frac{\partial u}{\partial y}$ (domain) | $2.448 \times 10^{-4}$  | $4.323 \times 10^{-6}$  | $8.316 \times 10^{-5}$  | $1.208 \times 10^{-7}$  | $1.421 \times 10^{-7}$  |
| max DBC-error ( $\overline{AB}$ )                   | 0.0                     | 0.0                     | 0.0                     | 0.0                     | 0.0                     |
| rms DBC-error ( $\overline{AB}$ )                   | 0.0                     | 0.0                     | 0.0                     | 0.0                     | 0.0                     |
| max DBC-error ( $\overline{BC}$ )                   | $4.441 \times 10^{-16}$ | $4.441 \times 10^{-16}$ | $8.882 \times 10^{-16}$ | $4.441 \times 10^{-16}$ | $1.776 \times 10^{-15}$ |
| rms DBC-error ( $\overline{BC}$ )                   | $1.302 \times 10^{-16}$ | $1.305 \times 10^{-16}$ | $2.242 \times 10^{-16}$ | $1.968 \times 10^{-16}$ | $3.513 \times 10^{-16}$ |
| max DBC-error ( $\overline{CD}$ )                   | 0.0                     | 0.0                     | 0.0                     | 0.0                     | 0.0                     |
| rms DBC-error ( $\overline{CD}$ )                   | 0.0                     | 0.0                     | 0.0                     | 0.0                     | 0.0                     |
| max DBC-error ( $\overline{AD}$ )                   | $8.882 \times 10^{-16}$ | $1.776 \times 10^{-15}$ | $1.776 \times 10^{-15}$ | $8.882 \times 10^{-16}$ | $8.882 \times 10^{-16}$ |
| rms DBC-error ( $\overline{AD}$ )                   | $3.418 \times 10^{-16}$ | $3.955 \times 10^{-16}$ | $4.159 \times 10^{-16}$ | $2.661 \times 10^{-16}$ | $1.768 \times 10^{-16}$ |



**Fig. 4.** Helmholtz equation (Neumann or Robin BC on boundaries): Distributions of the NN solutions (top row) and their point-wise absolute errors (bottom row) on domain #1. (a,d) Case #1: NBC on  $\overline{BC}$ , and DBCs on the other boundaries. (b,e) Case #2: RBC ( $\alpha = 1.0$ ) on  $\overline{BC}$ , and DBCs on the other boundaries. (c,f) Case #3: NBCs on  $\overline{BC}$  and  $\overline{CD}$ , and DBCs on the other boundaries. Simulation parameters:  $R_m = 5.0$ ,  $Q = 70$ ,  $M = 950$  for all cases. The color bars in the top-row plots denote the solution values, and those in the bottom-row plots denote values of the absolute errors. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

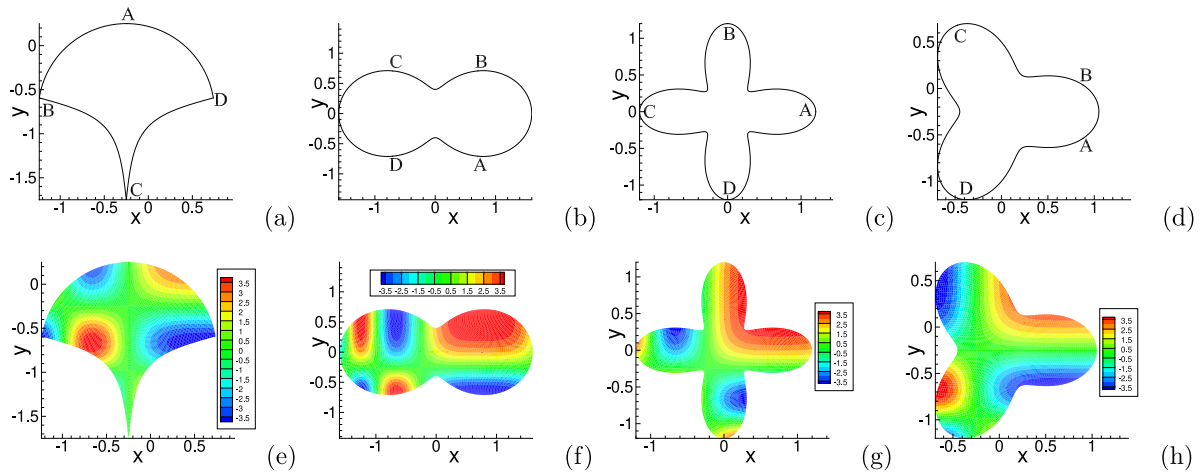
We next consider Neumann and Robin conditions on the domain. Fig. 4 and Table 4 illustrate the ELM results obtained for a combination of Dirichlet conditions with Neumann or Robin conditions. Fig. 4 shows the NN solutions (top row) and their point-wise errors (bottom row) for three cases. In case #1 (plots (a,d)), Neumann condition is imposed on the boundary  $\overline{BC}$  and Dirichlet conditions are imposed on the other boundaries. In case #2 (plots (b,e)), we impose the Robin condition (64b) with  $\alpha_{BC} = 1$  on  $\overline{BC}$ , with the rest being Dirichlet boundaries. In case #3 (plots (c,f)), we impose Neumann conditions on the boundaries  $\overline{BC}$  and  $\overline{CD}$ , and Dirichlet conditions on the other boundaries. The simulation parameter values are specified in the figure caption. The plots indicate that the largest solution errors generally occur on or near the Neumann or Robin boundaries, while the NN solution error is generally much smaller on the Dirichlet boundaries and in the interior of the domain. The maximum solution error over the domain is on the order of  $10^{-5}$  for these cases.

Table 4 lists the maximum and rms boundary-condition errors for different boundaries of these three cases ( $\epsilon_{max}, \epsilon_{rms}$ ), together with the maximum/rms errors ( $\epsilon_{max}^\Omega, \epsilon_{rms}^\Omega$ ) of the NN solution and its derivatives ( $u, \partial u/\partial x, \partial u/\partial y$ ) over the domain. Note that the error on  $\overline{BC}$  stands for the RBC error for case #2 and the NBC error for cases #1 and #3, and that the error on  $\overline{CD}$  stands for the

**Table 4**

Helmholtz equation on domain #1 (Neumann or Robin BCs): maximum and rms errors of the NN solution and its derivatives ( $u$ ,  $\partial u/\partial x$ ,  $\partial u/\partial y$ ) over the domain, and the maximum and rms boundary-condition (DBC, NBC, RBC) errors on the boundaries. Different cases correspond to those in Fig. 4.

|                                                     | Case #1                 | Case #2                 | Case #3                 |
|-----------------------------------------------------|-------------------------|-------------------------|-------------------------|
| max-error- $u$ (domain)                             | $4.536 \times 10^{-5}$  | $4.217 \times 10^{-5}$  | $8.295 \times 10^{-5}$  |
| rms-error- $u$ (domain)                             | $3.775 \times 10^{-6}$  | $3.420 \times 10^{-6}$  | $8.818 \times 10^{-6}$  |
| max-error- $\frac{\partial u}{\partial x}$ (domain) | $5.336 \times 10^{-4}$  | $4.981 \times 10^{-4}$  | $9.780 \times 10^{-4}$  |
| rms-error- $\frac{\partial u}{\partial x}$ (domain) | $4.964 \times 10^{-5}$  | $4.698 \times 10^{-5}$  | $8.371 \times 10^{-5}$  |
| max-error- $\frac{\partial u}{\partial y}$ (domain) | $1.236 \times 10^{-3}$  | $1.255 \times 10^{-3}$  | $9.092 \times 10^{-4}$  |
| rms-error- $\frac{\partial u}{\partial y}$ (domain) | $8.272 \times 10^{-5}$  | $8.462 \times 10^{-5}$  | $7.955 \times 10^{-5}$  |
| max DBC-error ( $\overline{AB}$ )                   | 0.0                     | 0.0                     | 0.0                     |
| rms DBC-error ( $\overline{AB}$ )                   | 0.0                     | 0.0                     | 0.0                     |
| max NBC- or RBC-error ( $\overline{BC}$ )           | $1.421 \times 10^{-14}$ | $1.421 \times 10^{-14}$ | $1.421 \times 10^{-14}$ |
| rms NBC- or RBC-error ( $\overline{BC}$ )           | $3.614 \times 10^{-15}$ | $4.180 \times 10^{-15}$ | $4.154 \times 10^{-15}$ |
| max DBC- or NBC-error ( $\overline{CD}$ )           | 0.0                     | 0.0                     | $1.421 \times 10^{-14}$ |
| rms DBC- or NBC-error ( $\overline{CD}$ )           | 0.0                     | 0.0                     | $3.065 \times 10^{-15}$ |
| max DBC-error ( $\overline{AD}$ )                   | $8.882 \times 10^{-16}$ | $8.882 \times 10^{-16}$ | $1.776 \times 10^{-15}$ |
| rms DBC-error ( $\overline{AD}$ )                   | $2.947 \times 10^{-16}$ | $2.947 \times 10^{-16}$ | $5.738 \times 10^{-16}$ |



**Fig. 5.** Nonlinear Helmholtz equation: Domain geometries (top row) and distributions of the exact solutions (bottom row), (a,e) domain #1, (b,f) domain #2, (c,g) domain #3, and (d,h) domain #4. The color bars denote the solution values. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

NBC error for case #3 and the DBC error for cases #1 and #2. The data show that the current method has enforced the Dirichlet, Neumann, and Robin boundary conditions to the machine accuracy.

### 3.2. Nonlinear Helmholtz equation

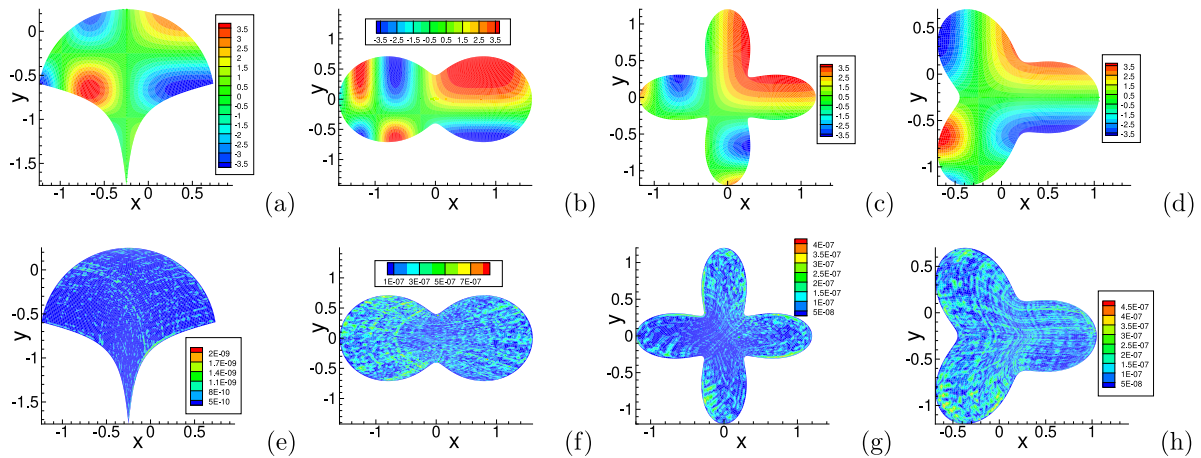
We next investigate the boundary value problem with the 2D nonlinear Helmholtz equation on the four domains  $\Omega$  as depicted in Fig. 5 (top row),

$$\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} - 20u + 10 \cos(2u) = f(x, y), \quad (111a)$$

$$\mathcal{B}u(x, y)|_{(x,y) \in \partial\Omega} = f_b(x, y), \quad (111b)$$

where  $u(x, y)$  is the field function to be computed,  $f$  and  $f_b$  are source terms for the PDE and the boundary conditions, and  $\mathcal{B}$  again denotes the Dirichlet, Neumann or Robin boundary conditions. The geometric parameters for these domains are specified in the Appendix 5.1. We set the source terms appropriately for different boundary conditions such that the problem has the following exact solution,

$$u(x, y) = 4 \cos \left[ \frac{\pi}{2} \left( x - \frac{3}{4} \right)^2 \right] \cos \left[ \frac{\pi}{2} \left( y - \frac{3}{4} \right)^2 \right]. \quad (112)$$



**Fig. 6.** Nonlinear Helmholtz equation (Dirichlet BC on all boundaries): Distributions of the NN solutions (top row) and their point-wise absolute errors (bottom row) on the four domains. Simulation parameters: (a,e)  $R_m = 4.0$ ,  $Q = 50$ ,  $M = 1000$ . (b,f)  $R_m = 4.5$ ,  $Q = 55$ ,  $M = 1000$ . (c,g,d,h)  $R_m = 5.5$ ,  $Q = 60$ ,  $M = 1000$ .  $Q_{db} = 8$  (see Remark 2.10) for all domains. The color bars in the top-row plots denote the solution values, and those in the bottom-row plots denote values of the absolute errors. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

**Table 5**

Nonlinear Helmholtz equation (Dirichlet BC on all boundaries): maximum and rms NN-solution errors over the domain, and the maximum and rms DBC errors on the four boundaries. Simulation parameters follow those of Fig. 6.

|                                   | Domain #1               | Domain #2               | Domain #3               | Domain #4               |
|-----------------------------------|-------------------------|-------------------------|-------------------------|-------------------------|
| max solution-error (domain)       | $2.544 \times 10^{-9}$  | $8.948 \times 10^{-7}$  | $4.475 \times 10^{-7}$  | $4.808 \times 10^{-7}$  |
| rms solution-error (domain)       | $4.359 \times 10^{-10}$ | $2.099 \times 10^{-7}$  | $7.637 \times 10^{-8}$  | $1.118 \times 10^{-7}$  |
| max DBC-error ( $\overline{AB}$ ) | 0.0                     | 0.0                     | 0.0                     | 0.0                     |
| rms DBC-error ( $\overline{AB}$ ) | 0.0                     | 0.0                     | 0.0                     | 0.0                     |
| max DBC-error ( $\overline{BC}$ ) | $8.882 \times 10^{-16}$ | $4.441 \times 10^{-16}$ | $5.551 \times 10^{-16}$ | $4.441 \times 10^{-16}$ |
| rms DBC-error ( $\overline{BC}$ ) | $2.303 \times 10^{-16}$ | $1.662 \times 10^{-16}$ | $2.075 \times 10^{-16}$ | $1.615 \times 10^{-16}$ |
| max DBC-error ( $\overline{CD}$ ) | 0.0                     | 0.0                     | 0.0                     | 0.0                     |
| rms DBC-error ( $\overline{CD}$ ) | 0.0                     | 0.0                     | 0.0                     | 0.0                     |
| max DBC-error ( $\overline{AD}$ ) | $4.441 \times 10^{-16}$ | $4.441 \times 10^{-16}$ | $4.996 \times 10^{-16}$ | $4.441 \times 10^{-16}$ |
| rms DBC-error ( $\overline{AD}$ ) | $2.169 \times 10^{-16}$ | $1.171 \times 10^{-16}$ | $1.995 \times 10^{-16}$ | $2.014 \times 10^{-16}$ |

Fig. 5 (bottom row) shows distributions of the exact solution over these domains.

For domains #1 and #4 (Fig. 5a and d), we employ the function in (9) to map the problem domain  $\Omega$  to the standard domain  $\Omega_{st}$ . This function, however, fails to produce a univalent map for domains #2 and #3. In the following simulations we employ the following mapping function for domains #2 and #3,

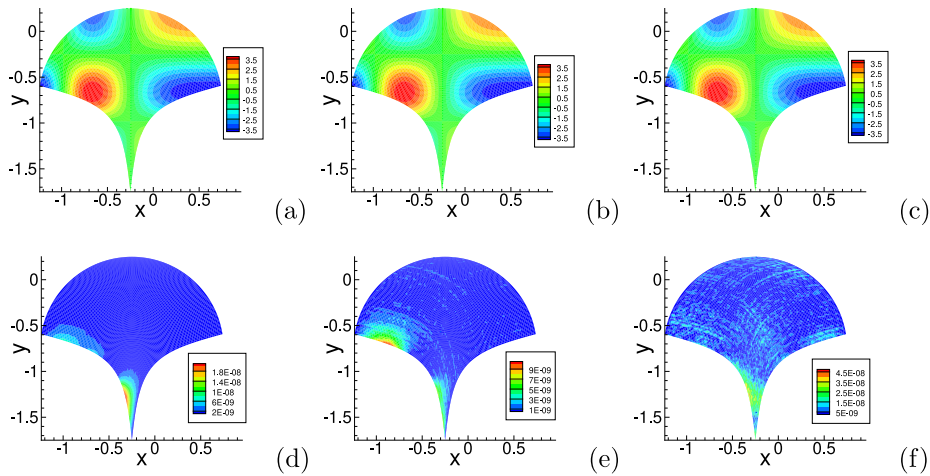
$$\mathbf{x}(\xi, \eta) = \mathbf{x}_{AD}(\eta)\varpi_0(\xi) + \mathbf{x}_{BC}(\eta)\varpi_2(\xi) + \mathbf{x}_{AB}(\xi)\varpi_0(\eta) + \mathbf{x}_{CD}(\xi)\varpi_2(\eta) + \mathbf{x}_I\varpi_1(\xi)\varpi_1(\eta) - [\mathbf{x}_A\varpi_0(\xi) + \mathbf{x}_B\varpi_2(\xi)]\varpi_0(\eta) - [\mathbf{x}_D\varpi_0(\xi) + \mathbf{x}_C\varpi_2(\xi)]\varpi_2(\eta). \quad (113)$$

Here  $\mathbf{x}_I = (0, 0)$  denotes the geometric center of the domains #2 and #3, and  $\varpi_i(\xi)$  ( $i = 0, 1, 2$ ) denote the Lagrange polynomials defined on the points  $\{-1, 0, 1\}$ , as given by

$$\varpi_0(\xi) = \frac{1}{2}\xi(\xi - 1), \quad \varpi_1(\xi) = (1 + \xi)(1 - \xi), \quad \varpi_2(\xi) = \frac{1}{2}\xi(\xi + 1), \quad \xi \in [-1, 1]. \quad (114)$$

The ELM simulation results with Dirichlet conditions on all domain boundaries are illustrated in Fig. 6 and Table 5 for different domains. Fig. 6 shows distributions of the NN solutions (top row) and their point-wise absolute errors (bottom row) for these four domains. These results are obtained using the simulation parameter values as given in the figure caption. The results indicate that the current method has captured the solution accurately on these domain geometries, with the maximum ELM error on the order of  $10^{-7}$  or  $10^{-9}$  for different domains.

Table 5 is an assessment of the boundary-condition errors on different boundaries, as well as the solution errors, for these four domains. The DBC error is exactly zero on some boundaries ( $\overline{AB}$  and  $\overline{CD}$ ), and has a maximum on the order of  $10^{-16}$  on the other boundaries ( $\overline{BC}$  and  $\overline{AD}$ ). Our method has evidently enforced the boundary condition to the machine accuracy on these complex



**Fig. 7.** Nonlinear Helmholtz equation (Neumann or Robin BCs): Distributions of the NN solutions (top row) and their point-wise absolute errors (bottom row) on domain #1. (a,d) Case #1: Neumann condition on  $\overline{BC}$  and Dirichlet condition on the other boundaries. (b,e) Case #2: Robin condition ( $\alpha = 1.0$ ) on  $\overline{BC}$  and Dirichlet condition on the other boundaries. (c,f) Case #3: Neumann condition on  $\overline{BC}$  and  $\overline{CD}$ , and Dirichlet condition on the other boundaries. Simulation parameters: (a,d)  $R_m = 4.0$ ,  $Q = 60$ ,  $M = 1000$ . (b,e)  $R_m = 3.5$ ,  $Q = 60$ ,  $M = 1000$ . (c,f)  $R_m = 2.5$ ,  $Q = 60$ ,  $M = 1000$ . The color bars in the top-row plots denote the solution values, and those in the bottom-row plots denote values of the absolute errors. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

**Table 6**

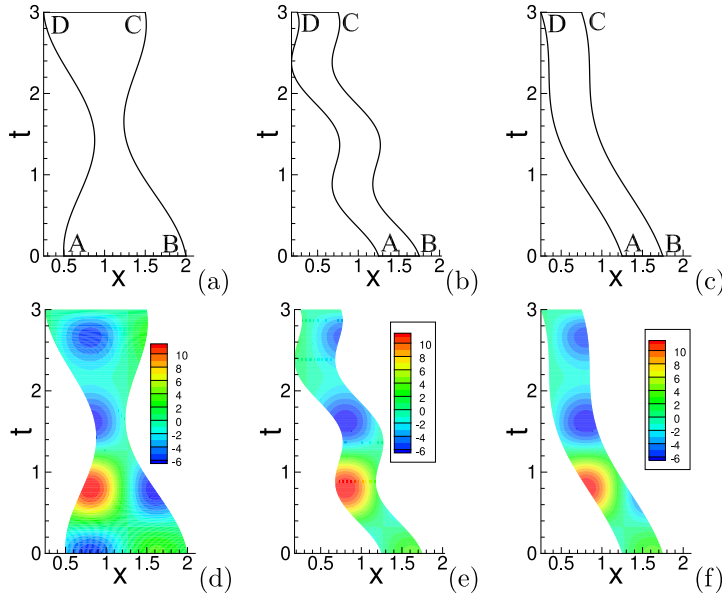
Nonlinear Helmholtz equation on domain #1 (Neumann or Robin BCs): maximum and rms NN-solution errors over the domain, and the maximum and rms boundary-condition (DBC, NBC, RBC) errors on different boundaries. The three cases correspond to those in Fig. 7 for different types of boundary conditions.

|                                           | Case #1                 | Case #2                 | Case #3                 |
|-------------------------------------------|-------------------------|-------------------------|-------------------------|
| max solution-error (domain)               | $2.133 \times 10^{-8}$  | $1.045 \times 10^{-8}$  | $4.709 \times 10^{-8}$  |
| rms solution-error (domain)               | $2.498 \times 10^{-9}$  | $1.367 \times 10^{-9}$  | $7.885 \times 10^{-9}$  |
| max DBC-error ( $\overline{AB}$ )         | 0.0                     | $1.388 \times 10^{-16}$ | 0.0                     |
| rms DBC-error ( $\overline{AB}$ )         | 0.0                     | $6.418 \times 10^{-17}$ | 0.0                     |
| max NBC- or RBC-error ( $\overline{BC}$ ) | $5.329 \times 10^{-15}$ | $6.217 \times 10^{-15}$ | $7.105 \times 10^{-15}$ |
| rms NBC- or RBC-error ( $\overline{BC}$ ) | $1.540 \times 10^{-15}$ | $2.198 \times 10^{-15}$ | $1.994 \times 10^{-15}$ |
| max DBC- or NBC-error ( $\overline{CD}$ ) | 0.0                     | $1.110 \times 10^{-16}$ | $6.217 \times 10^{-15}$ |
| rms DBC- or NBC-error ( $\overline{CD}$ ) | 0.0                     | $4.893 \times 10^{-17}$ | $1.702 \times 10^{-15}$ |
| max DBC-error ( $\overline{AD}$ )         | $6.661 \times 10^{-16}$ | $6.661 \times 10^{-16}$ | $4.441 \times 10^{-16}$ |
| rms DBC-error ( $\overline{AD}$ )         | $2.534 \times 10^{-16}$ | $2.534 \times 10^{-16}$ | $1.524 \times 10^{-16}$ |

boundaries. The NN solution error ranges from  $10^{-9}$  to  $10^{-7}$  on these domains. The error for the solution derivatives is generally one to two orders of magnitude larger than the NN-solution error.

In Fig. 7 and Table 6 we demonstrate the ELM simulation results obtained with Neumann or Robin boundary conditions. Fig. 7 shows the ELM solution and its point-wise absolute error on domain #1 for three cases: (i) Neumann condition imposed on  $\overline{BC}$  and Dirichlet conditions imposed on the rest of the boundaries (plots (a,d)), (ii) Robin condition with  $\alpha_{BC} = 1$  imposed on  $\overline{BC}$  and Dirichlet conditions imposed on the rest of the boundaries (plots (b,e)), and (iii) Neumann conditions imposed on  $\overline{BC}$  and  $\overline{CD}$  and Dirichlet conditions imposed on the rest of the boundaries. The simulation parameter values for each case are provided in the figure caption. It is observed that the ELM solution is highly accurate, with the maximum errors on the order of  $10^{-8}$  for all cases.

Table 6 demonstrates the accuracy of the current method for enforcing different types of boundary conditions. Here we list the boundary-condition errors ( $\epsilon_{max}$ ,  $\epsilon_{rms}$ ) on different boundaries for the three cases in Fig. 7, together with the maximum/rms NN-solution errors over the domain. The maximum boundary-condition error is on the order of  $10^{-15}$  or  $10^{-16}$ , and on some boundaries it is exactly zero. These results demonstrate that our method has enforced the Dirichlet, Neumann, and Robin conditions to the machine accuracy for this nonlinear problem.



**Fig. 8.** Heat equation on deforming/moving domains: space–time domain geometries (top row) and distributions (contours) of the exact solutions (bottom row). (a,d) Domain #1 (deforming spatial domain); (b,e) Domain #2 (moving spatial domain); (c,f) Domain #3 (another moving spatial domain). The color bars denote the solution values. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

### 3.3. Heat conduction on moving/deforming domains

In the next example we investigate the heat conduction problem on a spatial domain that deforms or moves over time. Specifically, we consider a time-dependent domain in 1D,  $\Omega(t) = [a(t), b(t)]$ , and the heat conduction equation on  $\Omega(t)$ ,

$$\frac{\partial u}{\partial t} - v \frac{\partial^2 u}{\partial x^2} = f(x, t), \quad x \in \Omega(t) = [a(t), b(t)], \quad t \in [0, t_f], \quad (115a)$$

$$u(a(t), t) = u_a(t), \quad t \in [0, t_f], \quad (115b)$$

$$u(b(t), t) = u_b(t), \quad t \in [0, t_f], \quad (115c)$$

$$u(x, 0) = u_{in}(x), \quad x \in [a(0), b(0)] = [x_a, x_b]. \quad (115d)$$

In the above equations,  $u(x, t)$  is the field function to be computed,  $v = 0.005$  is the diffusion coefficient (thermal diffusivity),  $t_f$  denotes the time horizon of the problem, and  $f(x, t)$  is a source term.  $u_a(t)$  and  $u_b(t)$  are prescribed boundary conditions, and  $u_{in}(x)$  denotes the initial condition.  $[x_a, x_b]$  denotes the initial domain (at  $t = 0$ ). It is assumed that the prescribed boundary and initial conditions are compatible, namely,  $u_{in}(x_a) = u_a(0)$  and  $u_{in}(x_b) = u_b(0)$ . We choose the source term  $f(x, t)$  and the boundary/initial conditions appropriately such that this problem has the following exact solution,

$$u(x, t) = [2 \cos(0.75\pi x + 0.42\pi) + 1.5 \cos(1.5\pi x - 0.22\pi)] \cdot [2 \cos(0.75\pi t + 0.42\pi) + 1.5 \cos(1.5\pi t - 0.22\pi)]. \quad (116)$$

We consider the following three specific domains for this problem:

- Domain #1 is defined by

$$a(t) = x_a(1 - t/t_f) + x_d(t/t_f) + 0.25 [1 - \cos(2\pi t/t_f)], \quad (117a)$$

$$b(t) = x_b(1 - t/t_f) + x_c(t/t_f) - 0.25 [1 - \cos(2\pi t/t_f)], \quad (117b)$$

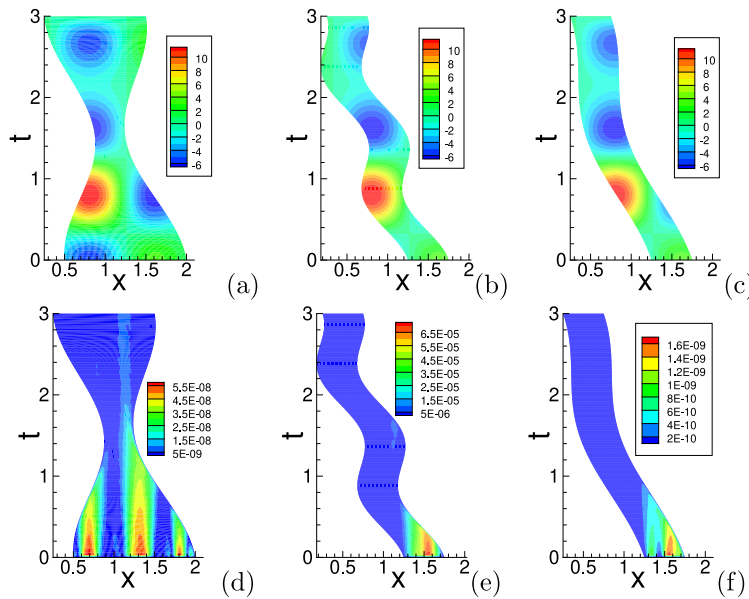
with the parameter values

$$x_a = 0.5, \quad x_b = 2.0, \quad x_c = 1.5, \quad x_d = 0.25, \quad t_f = 3.0. \quad (118)$$

This is a deforming spatial domain, and is shown in Fig. 8(a) as a space–time domain.

- Domain #2 is defined by

$$a(t) = x_a(1 - t/t_f) + x_d(t/t_f) + 0.15 [\cos(4\pi t/t_f) - 1], \quad (119a)$$



**Fig. 9.** Heat equation on deforming/moving domains: Distributions of the NN solutions (top row), and their point-wise absolute errors (bottom row) for domains #1 (a,d), #2 (b,e) and #3 (c,f). Simulation parameters: Domains #1 and #3,  $R_m = 4.0$ ,  $Q = 100$ ,  $Q_{db} = 7$ ,  $M = 1000$ ; Domain #2,  $R_m = 5.5$ ,  $Q = 100$ ,  $Q_{db} = 7$ ,  $M = 1000$ . The color bars in the top-row plots denote the solution values, and those in the bottom-row plots denote values of the absolute errors. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

**Table 7**

Heat equation on deforming/moving domains: the maximum and rms NN solution errors over the space-time domain and on the boundary  $\overline{CD}$ , and the maximum and rms boundary/initial condition errors on  $\overline{AB}$ ,  $\overline{BC}$ , and  $\overline{AD}$ . Note that Dirichlet conditions are imposed on  $\overline{AB}$ ,  $\overline{BC}$  and  $\overline{AD}$  of the space-time domain, and that no boundary condition is imposed on  $\overline{CD}$ . Simulation parameters follow those of Fig. 9.

|                                        | Domain #1               | Domain #2               | Domain #3               |
|----------------------------------------|-------------------------|-------------------------|-------------------------|
| max solution-error (domain)            | $6.083 \times 10^{-8}$  | $7.503 \times 10^{-5}$  | $1.736 \times 10^{-9}$  |
| rms solution-error (domain)            | $1.712 \times 10^{-8}$  | $1.401 \times 10^{-6}$  | $3.208 \times 10^{-10}$ |
| max BC-error ( $\overline{AB}$ )       | 0.0                     | 0.0                     | 0.0                     |
| rms BC-error ( $\overline{AB}$ )       | 0.0                     | 0.0                     | 0.0                     |
| max BC-error ( $\overline{BC}$ )       | $8.882 \times 10^{-16}$ | $8.882 \times 10^{-16}$ | $8.882 \times 10^{-16}$ |
| rms BC-error ( $\overline{BC}$ )       | $1.983 \times 10^{-16}$ | $3.436 \times 10^{-16}$ | $3.743 \times 10^{-16}$ |
| max solution-error ( $\overline{CD}$ ) | $6.537 \times 10^{-9}$  | $3.875 \times 10^{-6}$  | $3.662 \times 10^{-11}$ |
| rms solution-error ( $\overline{CD}$ ) | $2.765 \times 10^{-9}$  | $1.960 \times 10^{-6}$  | $1.711 \times 10^{-11}$ |
| max BC-error ( $\overline{AD}$ )       | $1.776 \times 10^{-15}$ | $2.220 \times 10^{-16}$ | $4.441 \times 10^{-16}$ |
| rms BC-error ( $\overline{AD}$ )       | $4.842 \times 10^{-16}$ | $7.842 \times 10^{-17}$ | $5.925 \times 10^{-17}$ |

$$b(t) = a(t) + (x_b - x_a), \quad (119b)$$

with the parameter values

$$x_a = 1.25, \quad x_b = 1.75, \quad x_d = 0.25, \quad t_f = 3.0. \quad (120)$$

This is a moving spatial domain, and is shown in Fig. 8(b) as a space-time domain.

- Domain #3 is defined by

$$a(t) = x_a(1 - t/t_f) + x_d(t/t_f) - 0.15 [1 - \cos(2\pi t/t_f)], \quad (121a)$$

$$b(t) = a(t) + (x_b - x_a), \quad (121b)$$

with the same parameter values as given in (120). This is another moving spatial domain and is shown in Fig. 8(c).

Distributions of the exact solution (116) over these domains are included in Fig. 8 (bottom row).

We solve this problem by a space–time approach and treat the time variable  $t$  on the same footing as the space variable  $x$ . The boundary conditions (115b)–(115c) and the initial condition (115d) all become Dirichlet type conditions imposed on the boundaries  $\overline{AB}$ ,  $\overline{BC}$  and  $\overline{AD}$  of the space–time domain  $D = \overline{ABCD} = [a(t), b(t)] \times [0, t_f]$  (see Fig. 8). No condition is imposed on the boundary  $\overline{CD}$ . These boundary conditions are enforced exactly using the method from Section 2.2, with a modification by removing the Dirichlet condition on  $\overline{CD}$  from the formulation therein.

Fig. 9 shows distributions of the ELM solutions (top row) and their point-wise absolute errors (bottom row) in the space–time plane for these three domains. The simulation parameter values are provided in the figure caption. The NN solutions on domains #1 and #3 are more accurate, with the maximum errors on the order of  $10^{-8}$  and  $10^{-9}$ , respectively, compared with that on domain #2, with the maximum error around  $10^{-5}$ .

Table 7 demonstrates the accuracy of the current method for enforcing the boundary conditions with this problem. Here we list the maximum/rms boundary-condition errors on  $\overline{AB}$ ,  $\overline{BC}$  and  $\overline{AD}$  of the space–time domain, together with the NN solution errors on  $\overline{CD}$  and over the entire domain. It is evident that our method has enforced the boundary conditions to the machine accuracy for this problem with moving/deforming boundaries. The NN has captured the solution in the space–time domain accurately. The errors of the solution derivatives ( $\partial u / \partial x$ ,  $\partial u / \partial t$ ) are at a level one to two orders of magnitude higher than that of the NN solution ( $u$ ).

#### 4. Concluding remarks

We have developed a systematic method for enforcing exactly the Dirichlet, Neumann, and Robin type boundary conditions on general quadrilateral domains with arbitrary curved boundaries. The method consists of two components, an exact mapping of general quadrilateral domains to the standard domain and a four-step procedure to systematically formulate the general forms of trial functions that exactly satisfy the imposed Dirichlet/Neumann/Robin conditions, both utilizing transfinite interpolations and TFC constrained expressions in their construction.

The constructed general forms of trial functions satisfying the imposed boundary conditions are in parametric forms, expressed with respect to the standard domain. When only Dirichlet boundaries are involved, the formulation is conceptually straightforward by leveraging the domain mapping. When Neumann or Robin type boundaries are present, the formulation becomes significantly more challenging. We formulate the general forms of trial functions for exact BC enforcement through a procedure consisting of four steps: (i) Identify the set of variables that the transformed boundary conditions and the compatibility constraints induced by these conditions are imposed on; (ii) Construct the transfinite interpolation for the types of identified variables; (iii) Formulate the preliminary TFC constrained expression based on this transfinite interpolation; (iv) Update terms of the transfinite interpolation by corresponding terms involving the free function from the preliminary TFC expression, thus giving rise to the final TFC form as the constructed trial function.

When Neumann (or Robin) boundaries are present, employing the four-step procedure, we have analyzed and presented in detail the formulation for two types of situations: (i) when a Neumann (or Robin) boundary only intersects with Dirichlet boundaries, and (ii) when two Neumann (or Robin) boundaries intersect with each other. When the quadrilateral domain involves a combination of Dirichlet, Neumann, and Robin boundaries and if multiple Neumann or Robin boundaries are present, the formulation either falls into or can be constructed based on the two aforementioned situations. In this case, at every vertex where two Neumann (or Robin) boundaries meet or where a Neumann (or Robin) boundary and a Dirichlet boundary meet, the compatibility constraints and the corresponding constructions for handling such constraints analyzed herein will apply. The overall formulation can be constructed analogously based on the four-step procedure.

The presented method for exact BC enforcement has been implemented together with the ELM technique for physics-informed machine learning. Extensive numerical experiments are conducted for several linear or nonlinear, stationary or dynamic, boundary/initial value problems on a variety of domains with complex geometries. The numerical results demonstrate that the current method has enforced the Dirichlet, Neumann, and Robin conditions to machine precision on curved domain boundaries.

How to enforce Dirichlet, Neumann, and Robin type conditions exactly on complex domain geometries is a central challenge in scientific machine learning. The method developed in this work constitutes only a preliminary step toward that objective. The present formulations are based on two-dimensional quadrilateral domains and thus inevitably entail certain geometric limitations. More specifically, the current method applies to two-dimensional simply-connected domains with piecewise smooth (or sufficiently smooth) boundaries, provided that the number of boundary segments does not exceed four. When mixed boundary conditions are imposed in a piecewise manner along a single smooth boundary segment, the number of admissible pieces is further reduced. As a result, the method faces difficulties for more general two-dimensional geometries, such as general polygons or multiply connected domains. The extension of the formulation to non-regular geometries in three dimensions is relatively straightforward for Dirichlet boundary conditions, but considerably more challenging for derivative-based boundary conditions, such as Neumann and Robin types. We are actively pursuing research along these lines and expect to report our findings in due course through other venues.

It is instructive to compare the present method with other exact or hard-constraint BC enforcement techniques in the literature. Table 8 provides such a comparison, summarizing representative existing strategies for exact or hard BC enforcement, together with the method developed here, across several criteria, including BC type, domain complexity, boundary regularity requirements, corner compatibility treatment, implementation complexity, and other notable strengths or limitations. Hard-constraint techniques differ mainly in how they trade off exactness, geometric flexibility, regularity requirements, and implementation effort. Exact trial-space methods, including classical liftings, hard ansatz embeddings, and distance-function constructions, obviate BC loss terms and usually deliver the best Dirichlet accuracy. But they do so by shifting difficulty to the design of the admissible representation. In

**Table 8**

Comparison between the current method and other representative exact or hard-constraint BC-enforcement strategies from the literature in terms of the BC focus, domain complexity, boundary regularity requirement, corner compatibility handling, implementation complexity, and other notable properties or limitations.

| Method                                   | BC focus                                   | Domain complexity                                      | Boundary regularity                                         | Corner compatibility                      | Implement complexity                                  | Notable property                                    | Limitations, other notes                     |
|------------------------------------------|--------------------------------------------|--------------------------------------------------------|-------------------------------------------------------------|-------------------------------------------|-------------------------------------------------------|-----------------------------------------------------|----------------------------------------------|
| Classical lifting/trial functions [9,45] | Mainly DBC; some mixed BCs                 | Simple domains; some extension to irregular boundaries | Requirement moderate for DBC, higher for derivative BCs     | Not built in                              | Low to moderate                                       | Foundational                                        | Limited geometry generality                  |
| Hard ansatz embedding [47,48]            | Mainly DBC; some other BC types            | Moderate; depends on output transform                  | Problem-dependent; Requirement grows with PDE order         | Not built in                              | Moderate                                              | Clean hard-embedding philosophy                     | Best when ansatz construction is simple      |
| Penalty-free or weak approach [20,46]    | Mainly DBC; weak treatment for other types | Complex geometries; weak form settings                 | Requirement lower than strong-form methods                  | Not built in                              | Moderate to high                                      | Weak form lowers derivative burden                  | Specialized to second-order problems         |
| Distance function methods [52–54]        | Strongest for DBC; with NBC and RBC        | Moderate to complex geometries                         | Requirement higher for derivative BCs and higher order PDEs | Not built in, delicate                    | Moderate to high, problem dependent                   | Geometry aware, approximate distance function based | Nontrivial geometry preprocessing            |
| TFC constrained expressions [60–62]      | Linear equality BCs                        | Mainly regular geometries                              | Requirement moderate to high, depends on PDE order          | Not built in; requires separate treatment | Moderate to high; algebraically heavy                 | Exact analytic BC embedding                         | Regular domain geometry                      |
| Specialized techniques [49,70]           | Mainly NBC; also DBC/RBC                   | Regular or complex domains                             | Requirement high                                            | Not built in, delicate                    | Moderate to high                                      | Promising derivative BC enforcement                 | Less mature than exact DBC methods           |
| Current method                           | DBC, NBC, RBC                              | Moderately complex geometries                          | Requirement moderate to high, depends on PDE order          | Built-in corner compatibility             | Higher, algebraically heavier than regular geometries | Numerical machine accuracy for complex geometries   | Limited to curvilinear quadrilateral domains |

particular, distance-function based methods are the most geometry-aware and offer a strong option for exact Dirichlet enforcement on smooth to moderately complex domains, whereas TFC constrained expressions are analytically elegant but most natural on regular domains. Across these methods, corner compatibility and derivative-based constraints remain delicate, since exact embedding does not eliminate singularities or incompatibilities already present in the boundary data. By contrast, the current method incorporates corner-compatibility constraints directly into the construction of the exact trial ansatz for Dirichlet, Neumann, and Robin conditions on moderately complex geometries, and enforces these BCs to numerical machine precision. At the same time, the formulation also reveals its own limitation: the method is restricted to curvilinear quadrilateral domains.

This table also shows that weak-form constrained formulations occupy a complementary regime. Penalty-free or weak variational approaches, such as PFNN [46] and related methods, reduce regularity demands and are attractive for second-order problems on complex geometries, although they are typically more specialized and less universal than exact strong-form embeddings. Overall, the most established hard-constraint results remain those for Dirichlet boundary conditions, while exact treatment of Neumann and Robin conditions is advancing rapidly. Corner handling and robust deployment on genuinely complex geometries remain the principal open challenges in the current literature.

#### CRedit authorship contribution statement

**Suchuan Dong:** Writing – review & editing, Writing – original draft, Visualization, Validation, Supervision, Software, Methodology, Data curation, Conceptualization. **Yuchuan Zhang:** Visualization, Validation, Software, Data curation, Writing – original draft, Writing – review & editing.

#### Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

## Appendix 5

### 5.1. Geometric domain parameters

#### Section 3.1: Helmholtz Equation

Domain #1:

Vertices:  $\mathbf{x}_A = (0.25, 0.25)$ ,  $\mathbf{x}_B = (2.5, 0.0)$ ,  $\mathbf{x}_C = (2.0, 2.5)$ ,  $\mathbf{x}_D = (0.0, 1.5)$ .

Edges:

$$\mathbf{x}_{AB}(\xi) = \mathbf{x}_A \phi_0(\xi) + \mathbf{x}_B \phi_1(\xi) - (0, h(\xi)), \quad \xi \in [-1, 1], \quad (122a)$$

$$\mathbf{x}_{BC}(\eta) = \mathbf{x}_B \phi_0(\eta) + \mathbf{x}_C \phi_1(\eta) + (h(\eta), 0), \quad \eta \in [-1, 1], \quad (122b)$$

$$\mathbf{x}_{CD}(\xi) = \mathbf{x}_D \phi_0(\xi) + \mathbf{x}_C \phi_1(\xi) - (0, h(\xi)), \quad \xi \in [-1, 1], \quad (122c)$$

$$\mathbf{x}_{AD}(\eta) = \mathbf{x}_A \phi_0(\eta) + \mathbf{x}_D \phi_1(\eta) - (h(\eta), 0), \quad \eta \in [-1, 1], \quad (122d)$$

where  $\phi_0(\xi)$  and  $\phi_1(\xi)$  are defined in (7), and  $h(\xi) = -0.15 [1 + \cos(\pi\xi)]$  for  $\xi \in [-1, 1]$ .

Domain #2:

This domain is formed by a unit circle centered at (1, 0), subtracting a second unit circle centered at (2, 0).

Vertices:  $\mathbf{x}_A = \left(\frac{3}{2}, -\frac{\sqrt{3}}{2}\right)$ ,  $\mathbf{x}_B = (1, 0)$ ,  $\mathbf{x}_C = \left(\frac{3}{2}, \frac{\sqrt{3}}{2}\right)$ ,  $\mathbf{x}_D = (0, 0)$ .

Edges:

$$\mathbf{x}_{AB}(\xi) = (2, 0) + (\cos \theta_{AB}(\xi), \sin \theta_{AB}(\xi)), \quad \theta_{AB}(\xi) = -\frac{2\pi}{3} \phi_0(\xi) - \pi \phi_1(\xi), \quad \xi \in [-1, 1]; \quad (123a)$$

$$\mathbf{x}_{BC}(\eta) = (2, 0) + (\cos \theta_{BC}(\eta), \sin \theta_{BC}(\eta)), \quad \theta_{BC}(\eta) = \pi \phi_0(\eta) + \frac{2\pi}{3} \phi_1(\eta), \quad \eta \in [-1, 1]; \quad (123b)$$

$$\mathbf{x}_{CD}(\xi) = (1, 0) + (\cos \theta_{CD}(\xi), \sin \theta_{CD}(\xi)), \quad \theta_{CD}(\xi) = \pi \phi_0(\xi) + \frac{\pi}{3} \phi_1(\xi), \quad \xi \in [-1, 1]; \quad (123c)$$

$$\mathbf{x}_{AD}(\eta) = (1, 0) + (\cos \theta_{AD}(\eta), \sin \theta_{AD}(\eta)), \quad \theta_{AD}(\eta) = -\frac{\pi}{3} \phi_0(\eta) - \pi \phi_1(\eta), \quad \eta \in [-1, 1]. \quad (123d)$$

Domain #3:

Vertices:  $\mathbf{x}_A = \left(-\frac{\sqrt{2}}{2}, -\frac{\sqrt{2}}{2}\right)$ ,  $\mathbf{x}_B = \left(\frac{\sqrt{2}}{2}, -\frac{\sqrt{2}}{2}\right)$ ,  $\mathbf{x}_C = \left(\frac{\sqrt{2}}{2}, \frac{\sqrt{2}}{2}\right)$ ,  $\mathbf{x}_D = \left(-\frac{\sqrt{2}}{2}, \frac{\sqrt{2}}{2}\right)$ .

Edges:

$$\begin{aligned} \mathbf{x}_{AB}(\xi) &= \left[1 - \frac{a_{AB}}{2} (1 + \cos(\pi\xi))\right] (\cos \theta_{AB}(\xi), \sin \theta_{AB}(\xi)), \\ \theta_{AB}(\xi) &= -\frac{3\pi}{4} \phi_0(\xi) - \frac{\pi}{4} \phi_1(\xi), \quad a_{AB} = 0.25, \quad \xi \in [-1, 1]; \end{aligned} \quad (124a)$$

$$\begin{aligned} \mathbf{x}_{BC}(\eta) &= \left[1 - \frac{a_{BC}}{2} (1 + \cos(\pi\eta))\right] (\cos \theta_{BC}(\eta), \sin \theta_{BC}(\eta)), \\ \theta_{BC}(\eta) &= -\frac{\pi}{4} \phi_0(\eta) + \frac{\pi}{4} \phi_1(\eta), \quad a_{BC} = 0.4, \quad \eta \in [-1, 1]; \end{aligned} \quad (124b)$$

$$\begin{aligned} \mathbf{x}_{CD}(\xi) &= \left[1 - \frac{a_{CD}}{2} (1 + \cos(\pi\xi))\right] (\cos \theta_{CD}(\xi), \sin \theta_{CD}(\xi)), \\ \theta_{CD}(\xi) &= \frac{3\pi}{4} \phi_0(\xi) + \frac{\pi}{4} \phi_1(\xi), \quad a_{CD} = 0.3, \quad \xi \in [-1, 1]; \end{aligned} \quad (124c)$$

$$\mathbf{x}_{AD}(\eta) = (\cos \theta_{AD}(\eta), \sin \theta_{AD}(\eta)), \quad \theta_{AD}(\eta) = \frac{5\pi}{4} \phi_0(\eta) + \frac{3\pi}{4} \phi_1(\eta), \quad \eta \in [-1, 1]. \quad (124d)$$

Domain #4:

This domain is formed by two straight sides ( $\overline{AB}$  and  $\overline{AD}$ ) and an elliptic arc ( $\overline{BCD}$ ).

Vertices:

$$\begin{cases} \mathbf{x}_A = (0, -1.35), & \mathbf{x}_B = \left(0.95 \cos\left(\frac{5\pi}{36}\right), 0.55 + 0.6 \sin\left(\frac{5\pi}{36}\right)\right), \\ \mathbf{x}_C = (0, 1.15), & \mathbf{x}_D = \left(-0.95 \cos\left(\frac{5\pi}{36}\right), 0.55 + 0.6 \sin\left(\frac{5\pi}{36}\right)\right). \end{cases} \quad (125)$$

Edges:

$$\mathbf{x}_{AB}(\xi) = \mathbf{x}_A \phi_0(\xi) + \mathbf{x}_B \phi_1(\xi), \quad \xi \in [-1, 1]; \quad (126a)$$

$$\mathbf{x}_{BC}(\eta) = (0.95 \cos \theta_{BC}(\eta), 0.55 + 0.6 \sin \theta_{BC}(\eta)), \quad \theta_{BC}(\eta) = \frac{5\pi}{36} \phi_0(\eta) + \frac{\pi}{2} \phi_1(\eta), \quad \eta \in [-1, 1]; \quad (126b)$$

$$\mathbf{x}_{CD}(\xi) = (0.95 \cos \theta_{CD}(\xi), 0.55 + 0.6 \sin \theta_{CD}(\xi)), \quad \theta_{CD}(\xi) = \frac{31\pi}{36} \phi_0(\xi) + \frac{\pi}{2} \phi_1(\xi), \quad \xi \in [-1, 1]; \quad (126c)$$

$$\mathbf{x}_{AD}(\eta) = \mathbf{x}_A \phi_0(\eta) + \mathbf{x}_D \phi_1(\eta), \quad \eta \in [-1, 1]. \quad (126d)$$

Domain #5:

This is a triangle with vertices at  $A$ ,  $B$  and  $D$ .  $C$  is the mid-point of  $\overline{BD}$ .

Vertices:  $\mathbf{x}_A = (0, 0)$ ,  $\mathbf{x}_B = (2, 0.2)$ ,  $\mathbf{x}_C = (1.3, 1)$ ,  $\mathbf{x}_D = (0.6, 1.8)$ .

Edges:

$$\mathbf{x}_{AB}(\xi) = \mathbf{x}_A \phi_0(\xi) + \mathbf{x}_B \phi_1(\xi), \quad \xi \in [-1, 1]; \quad (127a)$$

$$\mathbf{x}_{BC}(\eta) = \mathbf{x}_B \phi_0(\eta) + \mathbf{x}_C \phi_1(\eta), \quad \eta \in [-1, 1]; \quad (127b)$$

$$\mathbf{x}_{CD}(\xi) = \mathbf{x}_D \phi_0(\xi) + \mathbf{x}_C \phi_1(\xi), \quad \xi \in [-1, 1]; \quad (127c)$$

$$\mathbf{x}_{AD}(\eta) = \mathbf{x}_A \phi_0(\eta) + \mathbf{x}_D \phi_1(\eta), \quad \eta \in [-1, 1]. \quad (127d)$$

### Section 3.2: Nonlinear Helmholtz Equation

Domain #1:

Vertices:

$$\begin{cases} \mathbf{x}_A = (-0.25, 0.25), & \mathbf{x}_B = \left( \cos\left(\frac{19\pi}{20}\right) - 0.25, \sin\left(\frac{19\pi}{20}\right) - 0.75 \right), \\ \mathbf{x}_C = (-0.25, -1.75), & \mathbf{x}_D = \left( \cos\left(\frac{\pi}{20}\right) - 0.25, \sin\left(\frac{\pi}{20}\right) - 0.75 \right). \end{cases} \quad (128)$$

Edges:

$$\begin{aligned} \mathbf{x}_{AB}(\xi) &= (-0.25, -0.75) + (\cos \theta_{AB}(\xi), \sin \theta_{AB}(\xi)), \\ \theta_{AB}(\xi) &= \frac{\pi}{2} \phi_0(\xi) + \frac{19\pi}{20} \phi_1(\xi), \quad \xi \in [-1, 1]; \end{aligned} \quad (129a)$$

$$\begin{aligned} \mathbf{x}_{BC}(\eta) &= 2 \left[ \left( \cos\left(\frac{19\pi}{20}\right), \sin\left(\frac{19\pi}{20}\right) \right) \phi_0(\eta) + (0, -1) \phi_1(\eta) \right] - (\cos \theta_{BC}(\eta), \sin \theta_{BC}(\eta)) \\ &\quad + (-0.25, -0.75), \quad \theta_{BC}(\eta) = \frac{19\pi}{20} \phi_0(\eta) + \frac{3\pi}{2} \phi_1(\eta), \quad \eta \in [-1, 1]; \end{aligned} \quad (129b)$$

$$\begin{aligned} \mathbf{x}_{CD}(\xi) &= 2 \left[ \left( \cos\left(\frac{\pi}{20}\right), \sin\left(\frac{\pi}{20}\right) \right) \phi_0(\xi) + (0, -1) \phi_1(\xi) \right] - (\cos \theta_{CD}(\xi), \sin \theta_{CD}(\xi)) \\ &\quad + (-0.25, -0.75), \quad \theta_{CD}(\xi) = \frac{\pi}{20} \phi_0(\xi) - \frac{\pi}{2} \phi_1(\xi), \quad \xi \in [-1, 1]; \end{aligned} \quad (129c)$$

$$\begin{aligned} \mathbf{x}_{AD}(\eta) &= (-0.25, -0.75) + (\cos \theta_{AD}(\eta), \sin \theta_{AD}(\eta)), \\ \theta_{AD}(\eta) &= \frac{\pi}{2} \phi_0(\eta) + \frac{\pi}{20} \phi_1(\eta), \quad \eta \in [-1, 1]. \end{aligned} \quad (129d)$$

Domain #2:

Vertices:  $\mathbf{x}_A = \left( \frac{\sqrt{2}}{2}, -\frac{\sqrt{2}}{2} \right)$ ,  $\mathbf{x}_B = \left( \frac{\sqrt{2}}{2}, \frac{\sqrt{2}}{2} \right)$ ,  $\mathbf{x}_C = \left( -\frac{\sqrt{2}}{2}, \frac{\sqrt{2}}{2} \right)$ ,  $\mathbf{x}_D = \left( -\frac{\sqrt{2}}{2}, -\frac{\sqrt{2}}{2} \right)$ .

Edges:

$$\begin{aligned} \mathbf{x}_{AB}(\xi) &= [1 + 0.6 \cos(2\theta_{AB}(\xi))] (\cos \theta_{AB}(\xi), \sin \theta_{AB}(\xi)), \\ \theta_{AB}(\xi) &= -\frac{\pi}{4} \phi_0(\xi) + \frac{\pi}{4} \phi_1(\xi), \quad \xi \in [-1, 1]; \end{aligned} \quad (130a)$$

$$\begin{aligned} \mathbf{x}_{BC}(\eta) &= [1 + 0.6 \cos(2\theta_{BC}(\eta))] (\cos \theta_{BC}(\eta), \sin \theta_{BC}(\eta)), \\ \theta_{BC}(\eta) &= \frac{\pi}{4} \phi_0(\eta) + \frac{3\pi}{4} \phi_1(\eta), \quad \eta \in [-1, 1]; \end{aligned} \quad (130b)$$

$$\begin{aligned} \mathbf{x}_{CD}(\xi) &= [1 + 0.6 \cos(2\theta_{CD}(\xi))] (\cos \theta_{CD}(\xi), \sin \theta_{CD}(\xi)), \\ \theta_{CD}(\xi) &= \frac{5\pi}{4} \phi_0(\xi) + \frac{3\pi}{4} \phi_1(\xi), \quad \xi \in [-1, 1]; \end{aligned} \quad (130c)$$

$$\begin{aligned} \mathbf{x}_{AD}(\eta) &= [1 + 0.6 \cos(2\theta_{AD}(\eta))] (\cos \theta_{AD}(\eta), \sin \theta_{AD}(\eta)), \\ \theta_{AD}(\eta) &= -\frac{3\pi}{4} \phi_0(\eta) - \frac{\pi}{4} \phi_1(\eta), \quad \eta \in [-1, 1]. \end{aligned} \quad (130d)$$

**Algorithm 1:** Computing the NN solution (DBC on all boundaries) – Form #1

---

**input :**  $g(\xi, \eta)$ ,  $g(\pm 1, \eta)$ ,  $g(\xi, \pm 1)$ ,  $g(\pm 1, \pm 1)$ ;  $F(\pm 1, \eta)$ ,  $F(\xi, \pm 1)$ ,  $F(\pm 1, \pm 1)$   
**output:**  $V(\xi, \eta)$

- 1 calculate  $A(\xi, \eta) = P_g(\xi, \eta)$  based on equation (16a), i.e.  

$$A(\xi, \eta) = g(-1, \eta)\phi_0(\xi) + g(1, \eta)\phi_1(\xi) + g(\xi, -1)\phi_0(\eta) + g(\xi, 1)\phi_1(\eta) \\ - [g(-1, -1)\phi_0(\eta) + g(-1, 1)\phi_1(\eta)]\phi_0(\xi) - [g(1, -1)\phi_0(\eta) + g(1, 1)\phi_1(\eta)]\phi_1(\xi)$$
- 2 calculate  $B(\xi, \eta) = P_F(\xi, \eta)$  based on equation (16b), i.e.  

$$B(\xi, \eta) = F(-1, \eta)\phi_0(\xi) + F(1, \eta)\phi_1(\xi) + F(\xi, -1)\phi_0(\eta) + F(\xi, 1)\phi_1(\eta) \\ - [F(-1, -1)\phi_0(\eta) + F(-1, 1)\phi_1(\eta)]\phi_0(\xi) - [F(1, -1)\phi_0(\eta) + F(1, 1)\phi_1(\eta)]\phi_1(\xi)$$
- 3 calculate  $V(\xi, \eta) = g(\xi, \eta) - A(\xi, \eta) + B(\xi, \eta)$

---

**Domain #3:**

Vertices:  $\mathbf{x}_A = (1.2, 0)$ ,  $\mathbf{x}_B = (0, 1.2)$ ,  $\mathbf{x}_C = (-1.2, 0)$ ,  $\mathbf{x}_D = (0, -1.2)$ .

Edges:

$$\begin{aligned} \mathbf{x}_{AB}(\xi) &= [0.8 + 0.4 \cos(4\theta_{AB}(\xi))] (\cos \theta_{AB}(\xi), \sin \theta_{AB}(\xi)), \\ \theta_{AB}(\xi) &= \frac{\pi}{2} \phi_1(\xi), \quad \xi \in [-1, 1]; \end{aligned} \quad (131a)$$

$$\begin{aligned} \mathbf{x}_{BC}(\eta) &= [0.8 + 0.4 \cos(4\theta_{BC}(\eta))] (\cos \theta_{BC}(\eta), \sin \theta_{BC}(\eta)), \\ \theta_{BC}(\eta) &= \frac{\pi}{2} \phi_0(\eta) + \pi \phi_1(\eta), \quad \eta \in [-1, 1]; \end{aligned} \quad (131b)$$

$$\begin{aligned} \mathbf{x}_{CD}(\xi) &= [0.8 + 0.4 \cos(4\theta_{CD}(\xi))] (\cos \theta_{CD}(\xi), \sin \theta_{CD}(\xi)), \\ \theta_{CD}(\xi) &= \frac{3\pi}{2} \phi_0(\xi) + \pi \phi_1(\xi), \quad \xi \in [-1, 1]; \end{aligned} \quad (131c)$$

$$\begin{aligned} \mathbf{x}_{AD}(\eta) &= [0.8 + 0.4 \cos(4\theta_{AD}(\eta))] (\cos \theta_{AD}(\eta), \sin \theta_{AD}(\eta)), \\ \theta_{AD}(\eta) &= -\frac{\pi}{2} \phi_1(\eta), \quad \eta \in [-1, 1]. \end{aligned} \quad (131d)$$

**Domain #4:**

Vertices:

$$\begin{cases} \mathbf{x}_A = (0.75 + 0.3 \cos(3\theta_A)) (\cos \theta_A, \sin \theta_A), & \mathbf{x}_B = (0.75 + 0.3 \cos(3\theta_B)) (\cos \theta_B, \sin \theta_B), \\ \mathbf{x}_C = (0.75 + 0.3 \cos(3\theta_C)) (\cos \theta_C, \sin \theta_C), & \mathbf{x}_D = (0.75 + 0.3 \cos(3\theta_D)) (\cos \theta_D, \sin \theta_D), \\ \theta_A = -\frac{\pi}{10}, \quad \theta_B = \frac{\pi}{10}, \quad \theta_C = \frac{13\pi}{20}, \quad \theta_D = \frac{27\pi}{20}. \end{cases} \quad (132)$$

Edges:

$$\begin{aligned} \mathbf{x}_{AB}(\xi) &= [0.75 + 0.3 \cos(3\theta_{AB}(\xi))] (\cos \theta_{AB}(\xi), \sin \theta_{AB}(\xi)), \\ \theta_{AB}(\xi) &= \theta_A \phi_0(\xi) + \theta_B \phi_1(\xi), \quad \xi \in [-1, 1]; \end{aligned} \quad (133a)$$

$$\begin{aligned} \mathbf{x}_{BC}(\eta) &= [0.75 + 0.3 \cos(3\theta_{BC}(\eta))] (\cos \theta_{BC}(\eta), \sin \theta_{BC}(\eta)), \\ \theta_{BC}(\eta) &= \theta_B \phi_0(\eta) + \theta_C \phi_1(\eta), \quad \eta \in [-1, 1]; \end{aligned} \quad (133b)$$

$$\begin{aligned} \mathbf{x}_{CD}(\xi) &= [0.75 + 0.3 \cos(3\theta_{CD}(\xi))] (\cos \theta_{CD}(\xi), \sin \theta_{CD}(\xi)), \\ \theta_{CD}(\xi) &= \theta_D \phi_0(\xi) + \theta_C \phi_1(\xi), \quad \xi \in [-1, 1]; \end{aligned} \quad (133c)$$

$$\begin{aligned} \mathbf{x}_{AD}(\eta) &= [0.75 + 0.3 \cos(3\theta_{AD}(\eta))] (\cos \theta_{AD}(\eta), \sin \theta_{AD}(\eta)), \\ \theta_{AD}(\eta) &= \theta_A \phi_0(\eta) + (\theta_D - 2\pi) \phi_1(\eta), \quad \eta \in [-1, 1]. \end{aligned} \quad (133d)$$

## 5.2. Evaluation of TFC expressions and effect of $Q_{db}$ on NN solution

We have touched on the cancellation error in Remark 2.10 in evaluating the TFC expression and its effect on the solution accuracy on the boundary test points. In this appendix, we compare several different algorithms for evaluating the TFC expression and illustrate the effect of  $Q_{db}$ , the number of boundary collocation points on which to enforce  $g(\xi, \eta) = 0$  as discussed in Remark 2.10, on the NN solution.

We focus on the Dirichlet conditions first, and will extend the discussions to other types of boundary conditions afterwards. To compute the NN solution  $V(\xi, \eta)$  (Eq. (15)) for Dirichlet BCs, we consider three forms for evaluating the TFC expression, as listed

**Algorithm 2:** Computing the NN solution (DBC on all boundaries) – Form #2**input :**  $g(\xi, \eta)$ ,  $g(\pm 1, \eta)$ ,  $g(\xi, \pm 1)$ ,  $g(\pm 1, \pm 1)$ ;  $F(\pm 1, \eta)$ ,  $F(\xi, \pm 1)$ ,  $F(\pm 1, \pm 1)$ **output:**  $V(\xi, \eta)$ 1 calculate  $A(\xi, \eta) = Pg(\xi, \eta)$  by

$$A(\xi, \eta) = [g(-1, \eta) - g(-1, -1)\phi_0(\eta)]\phi_0(\xi) + [g(1, \eta) - g(1, 1)\phi_1(\eta)]\phi_1(\xi) \\ + [g(\xi, -1) - g(1, -1)\phi_1(\xi)]\phi_0(\eta) + [g(\xi, 1) - g(-1, 1)\phi_0(\xi)]\phi_1(\eta) \quad (134)$$

2 calculate  $B(\xi, \eta) = PF(\xi, \eta)$  by

$$B(\xi, \eta) = [F(-1, \eta) - F(-1, -1)\phi_0(\eta)]\phi_0(\xi) + [F(1, \eta) - F(1, 1)\phi_1(\eta)]\phi_1(\xi) \\ + [F(\xi, -1) - F(1, -1)\phi_1(\xi)]\phi_0(\eta) + [F(\xi, 1) - F(-1, 1)\phi_0(\xi)]\phi_1(\eta) \quad (135)$$

3 calculate  $V(\xi, \eta) = g(\xi, \eta) - A(\xi, \eta) + B(\xi, \eta)$ **Algorithm 3:** Computing the NN solution (DBC on all boundaries) – Form #3**input :**  $g(\xi, \eta)$ ,  $g(\pm 1, \eta)$ ,  $g(\xi, \pm 1)$ ,  $g(\pm 1, \pm 1)$ ;  $F(\pm 1, \eta)$ ,  $F(\xi, \pm 1)$ ,  $F(\pm 1, \pm 1)$ **output:**  $V(\xi, \eta)$ 1 calculate  $A(\xi, \eta) = g(\xi, \eta) - Pg(\xi, \eta)$  based on equation (103a), i.e.

$$A(\xi, \eta) = [g(\xi, \eta) - g(\xi, -1)\phi_0(\eta) - g(\xi, 1)\phi_1(\eta)] - [g(-1, \eta) - g(-1, -1)\phi_0(\eta) - g(-1, 1)\phi_1(\eta)]\phi_0(\xi) \\ - [g(1, \eta) - g(1, -1)\phi_0(\eta) - g(1, 1)\phi_1(\eta)]\phi_1(\xi)$$

2 calculate  $B(\xi, \eta) = PF(\xi, \eta)$  based on equation (103b), i.e.

$$B(\xi, \eta) = [F(\xi, -1)\phi_0(\eta) + F(\xi, 1)\phi_1(\eta)] + [F(-1, \eta) - F(-1, -1)\phi_0(\eta) - F(-1, 1)\phi_1(\eta)]\phi_0(\xi) \\ + [F(1, \eta) - F(1, -1)\phi_0(\eta) - F(1, 1)\phi_1(\eta)]\phi_1(\xi)$$

3 calculate  $V(\xi, \eta) = A(\xi, \eta) + B(\xi, \eta)$ **Table 9**

Helmholtz equation (domain #1, DBC on all boundaries): Effect of  $Q_{db}$  on the free-function magnitude  $\max_{(\xi, \eta) \in \Omega_{st}} \{|g(\xi, \eta)|\}$ , the max solution error  $e_{max}^Q(u)$ , and the max boundary-condition errors on the four boundaries ( $\epsilon_{max}^{AB}$ ,  $\epsilon_{max}^{BC}$ ,  $\epsilon_{max}^{CD}$ ,  $\epsilon_{max}^{AD}$ ), with the NN solution computed by Algorithms 1, 2 and 3 (Alg #1 to #3).  $Q_{db}$  is the number of boundary points on which to enforce  $g(\xi, \eta) = 0$  as defined in Remark 2.10. Simulation parameters follow those of Fig. 3(a,f) with  $R_m = 4.62$ ,  $Q = 70$  and  $M = 800$ .

| Alg | $Q_{db}/4$ | $\max\{ g(\xi, \eta) \}$ | $e_{max}^Q(u)$        | $\epsilon_{max}^{AB}$  | $\epsilon_{max}^{BC}$  | $\epsilon_{max}^{CD}$  | $\epsilon_{max}^{AD}$  |
|-----|------------|--------------------------|-----------------------|------------------------|------------------------|------------------------|------------------------|
| #1  | 0          | $6.58 \times 10^9$       | $1.37 \times 10^{-5}$ | $9.54 \times 10^{-7}$  | $9.54 \times 10^{-7}$  | $9.54 \times 10^{-7}$  | $9.54 \times 10^{-7}$  |
|     | 1          | $1.57 \times 10^9$       | $1.33 \times 10^{-5}$ | $1.19 \times 10^{-7}$  | $2.38 \times 10^{-7}$  | $2.38 \times 10^{-7}$  | $2.38 \times 10^{-7}$  |
|     | 4          | $2.68 \times 10^8$       | $1.46 \times 10^{-5}$ | $7.45 \times 10^{-9}$  | $2.98 \times 10^{-8}$  | $7.45 \times 10^{-9}$  | $2.98 \times 10^{-8}$  |
|     | 6          | $4.33 \times 10^7$       | $1.26 \times 10^{-5}$ | $1.16 \times 10^{-10}$ | $7.45 \times 10^{-9}$  | $4.66 \times 10^{-10}$ | $7.45 \times 10^{-9}$  |
|     | 11         | $2.96 \times 10^5$       | $1.60 \times 10^{-5}$ | $1.46 \times 10^{-11}$ | $2.91 \times 10^{-11}$ | $2.91 \times 10^{-11}$ | $1.46 \times 10^{-11}$ |
|     | 16         | $1.78 \times 10^3$       | $2.36 \times 10^{-5}$ | $2.28 \times 10^{-13}$ | $2.27 \times 10^{-13}$ | $5.68 \times 10^{-14}$ | $2.82 \times 10^{-14}$ |
|     | 21         | $5.65 \times 10^1$       | $3.44 \times 10^{-5}$ | $1.78 \times 10^{-15}$ | $8.33 \times 10^{-16}$ | $3.55 \times 10^{-15}$ | $1.78 \times 10^{-15}$ |
|     | 26         | $2.36 \times 10^1$       | $3.47 \times 10^{-5}$ | $1.78 \times 10^{-15}$ | $8.88 \times 10^{-16}$ | $1.78 \times 10^{-15}$ | $1.78 \times 10^{-15}$ |
| #2  | 0          | $6.58 \times 10^9$       | $1.37 \times 10^{-5}$ | $2.38 \times 10^{-7}$  | $2.38 \times 10^{-7}$  | $4.77 \times 10^{-7}$  | $4.77 \times 10^{-7}$  |
|     | 1          | $1.57 \times 10^9$       | $1.33 \times 10^{-5}$ | $2.98 \times 10^{-8}$  | $2.22 \times 10^{-16}$ | $1.11 \times 10^{-16}$ | $8.88 \times 10^{-16}$ |
|     | 4          | $2.68 \times 10^8$       | $1.47 \times 10^{-5}$ | $8.88 \times 10^{-16}$ | $2.22 \times 10^{-16}$ | $1.11 \times 10^{-16}$ | $8.88 \times 10^{-16}$ |
|     | 6          | $4.33 \times 10^7$       | $1.26 \times 10^{-5}$ | $8.88 \times 10^{-16}$ | $2.22 \times 10^{-16}$ | $1.11 \times 10^{-16}$ | $8.88 \times 10^{-16}$ |
|     | 11         | $2.96 \times 10^5$       | $1.60 \times 10^{-5}$ | $8.88 \times 10^{-16}$ | $2.22 \times 10^{-16}$ | $1.11 \times 10^{-16}$ | $8.88 \times 10^{-16}$ |
| #3  | 0          | $6.58 \times 10^9$       | $1.14 \times 10^{-5}$ | 0.0                    | $4.44 \times 10^{-16}$ | 0.0                    | $8.88 \times 10^{-16}$ |
|     | 1          | $1.56 \times 10^9$       | $1.11 \times 10^{-5}$ | 0.0                    | $4.44 \times 10^{-16}$ | 0.0                    | $8.88 \times 10^{-16}$ |
|     | 4          | $2.68 \times 10^8$       | $1.62 \times 10^{-5}$ | 0.0                    | $4.44 \times 10^{-16}$ | 0.0                    | $8.88 \times 10^{-16}$ |
|     | 6          | $7.82 \times 10^7$       | $1.17 \times 10^{-5}$ | 0.0                    | $4.44 \times 10^{-16}$ | 0.0                    | $8.88 \times 10^{-16}$ |
|     | 11         | $2.95 \times 10^5$       | $1.57 \times 10^{-5}$ | 0.0                    | $4.44 \times 10^{-16}$ | 0.0                    | $8.88 \times 10^{-16}$ |

in Algorithms 1 to 3. Algorithm 1 follows the form given in Eqs. (15) and (16a)–(16b). Algorithm 2 employs a modified form for computing  $Pg(\xi, \eta)$  and  $PF(\xi, \eta)$ . Algorithm 3 adopts the form as given in Eqs. (103a)–(103b).

Table 9 shows the maximum absolute NN-solution errors and the maximum absolute boundary-condition errors, as well as the largest magnitude of the free function ( $\max_{(\xi, \eta) \in \Omega_{st}} \{|g(\xi, \eta)|\}$ ), for the Helmholtz equation on domain #1 from Section 3.1 (see Fig. 2(a)), with the NN solution computed using these three algorithms. These results are obtained using different  $Q_{db}$  (see Remark 2.10) in NN training, while the other parameter values follow those of Fig. 3(a,f).  $Q_{db} = 0$  stands for the case in which  $g(\xi, \eta) = 0$  is ignored.  $Q_{db}/4 = n_{db}$  ( $n_{db} \geq 1$ ) corresponds to a case in which  $g(\xi, \eta) = 0$  is enforced on  $(n_{db} - 1)$  uniform grid points on the interior of each boundary (in standard domain) plus the four vertices.

We can make the following observations from Table 9:

- $Q_{db}$  influences significantly the magnitude of the obtained free function  $g(\xi, \eta)$ , which decreases dramatically as  $Q_{db}$  increases.
- $Q_{db}$  has little (or minimum) influence on the NN-solution error over the domain.

**Algorithm 4:** Computing the NN solution (NBC on  $\overline{BC}$ , DBCs on other boundaries)**input :**  $g(\xi, \eta)$ ,  $g(\pm 1, \eta)$ ,  $g(\xi, \pm 1)$ ,  $g(\pm 1, \pm 1)$ ,  $F(\pm 1, \eta)$ ,  $F(\xi, \pm 1)$ ,  $F(\pm 1, \pm 1)$ , and their derivatives**output:**  $V(\xi, \eta)$ 

$$\begin{aligned}
1 \text{ calculate } A(\xi, \eta) &= g(\xi, \eta) - P g(\xi, \eta) \text{ by} \\
A(\xi, \eta) &= [g(\xi, \eta) - g(\xi, -1)\varphi_0(\eta) - g(\xi, 1)\varphi_1(\eta)] \\
&\quad - [g(-1, \eta) - g(-1, -1)\varphi_0(\eta) - g(-1, 1)\varphi_1(\eta)] \varphi_0(\xi) \\
&\quad - [g_\xi(1, \eta) - g_\xi(1, -1)\varphi_0(\eta) - g_\xi(1, 1)\varphi_1(\eta)] \varphi_1(\xi) \\
&\quad - [\lambda_B g_\eta(1, -1)\psi_0(\eta) + \lambda_C g_\eta(1, 1)\psi_1(\eta)] \varphi_1(\xi)
\end{aligned} \tag{136}$$

$$\begin{aligned}
2 \text{ calculate } F_\xi^g(1, \eta) &\text{ based on equation (34a)} \\
3 \text{ calculate } B(\xi, \eta) &= P F^g(\xi, \eta) \text{ by} \\
B(\xi, \eta) &= [F(\xi, -1)\varphi_0(\eta) + F(\xi, 1)\varphi_1(\eta)] + [F(-1, \eta) - F(-1, -1)\varphi_0(\eta) - F(-1, 1)\varphi_1(\eta)] \varphi_0(\xi) \\
&\quad + [F_\xi^g(1, \eta) - F_\xi^g(1, -1)\varphi_0(\eta) - F_\xi^g(1, 1)\varphi_1(\eta)] \varphi_1(\xi) \\
&\quad + [\lambda_B F_\eta^a(1, -1)\psi_0(\eta) + \lambda_C F_\eta^a(1, 1)\psi_1(\eta)] \varphi_1(\xi)
\end{aligned} \tag{137}$$

$$4 \text{ calculate } V(\xi, \eta) = A(\xi, \eta) + B(\xi, \eta)$$

**Algorithm 5:** Computing the NN solution (NBCs on  $\overline{BC}$  and  $\overline{CD}$ , DBCs on other boundaries)**input :**  $g(\xi, \eta)$ ,  $g(\pm 1, \eta)$ ,  $g(\xi, \pm 1)$ ,  $g(\pm 1, \pm 1)$ ,  $F(\pm 1, \eta)$ ,  $F(\xi, \pm 1)$ ,  $F(\pm 1, \pm 1)$ , and their derivatives**output:**  $V(\xi, \eta)$ 

$$\begin{aligned}
1 \text{ calculate } A(\xi, \eta) &= g(\xi, \eta) - P g(\xi, \eta) \text{ by} \\
A(\xi, \eta) &= [g(\xi, \eta) - g(\xi, -1)\rho_0(\eta) - g_\eta(\xi, 1)v_1(\eta)] \\
&\quad - [g(-1, \eta) - g(-1, -1)\rho_0(\eta) - g_\eta(-1, 1)v_1(\eta)] \rho_0(\xi) \\
&\quad - [g_\xi(1, \eta) - g_\xi(1, -1)\rho_0(\eta) - g_{\xi\eta}(1, 1)v_1(\eta)] v_1(\xi) \\
&\quad - \lambda_B g_\eta(1, -1)v_0(\eta)\rho_1(\xi) - [\lambda_D g_\xi(-1, 1)v_0(\xi) + \lambda_C g_{\xi\xi}(1, 1)\omega_1(\xi)] \rho_1(\eta)
\end{aligned} \tag{138}$$

$$\begin{aligned}
2 \text{ calculate } F_\xi^g(1, 1), F_\xi^g(1, -1), F_\xi^g(1, \eta) \text{ and } F_\eta^g(\xi, 1) &\text{ based on equations (62a)–(62d)} \\
3 \text{ calculate } B(\xi, \eta) &= P F^g(\xi, \eta) \text{ by} \\
B(\xi, \eta) &= [F(\xi, -1)\rho_0(\eta) + F_\eta^g(\xi, 1)v_1(\eta)] + [F(-1, \eta) - F(-1, -1)\rho_0(\eta) - F_\eta(-1, 1)v_1(\eta)] \rho_0(\xi) \\
&\quad + [F_\xi^g(1, \eta) - F_\xi^g(1, -1)\rho_0(\eta) - F_{\xi\eta}^g(1, 1)v_1(\eta)] v_1(\xi) \\
&\quad + \lambda_B F_\eta^a(1, -1)v_0(\eta)\rho_1(\xi) + [\lambda_D F_\xi^a(-1, 1)v_0(\xi) + \lambda_C F_{\xi\xi}^g(1, 1)\omega_1(\xi)] \rho_1(\eta)
\end{aligned} \tag{139}$$

$$4 \text{ calculate } V(\xi, \eta) = A(\xi, \eta) + B(\xi, \eta)$$

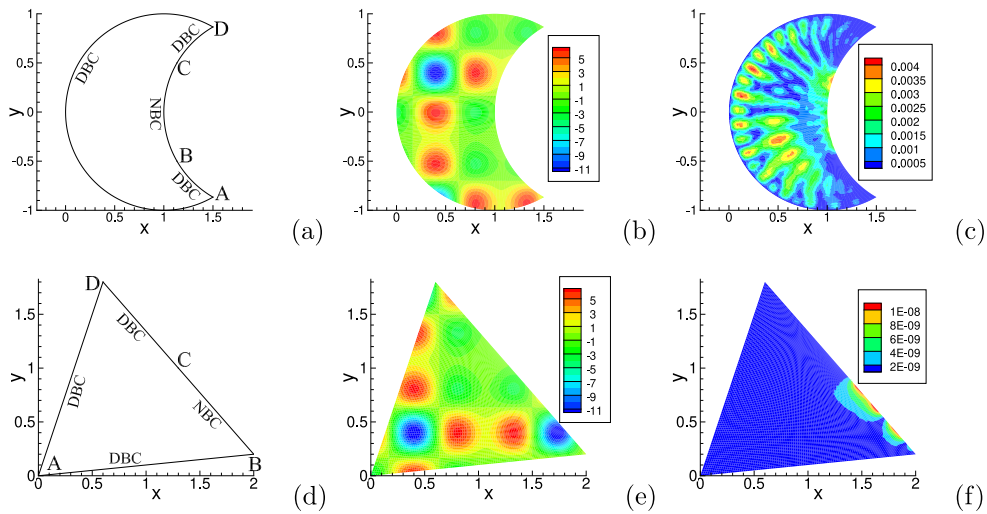
- $Q_{db}$  tends to notably influence the boundary-condition errors, depending on the algorithm used for computing the NN solution.
- For Algorithms 1 and 2, if  $g(\xi, \eta) = 0$  is not enforced when training the NN ( $Q_{db} = 0$ ), the resultant DBC error can be quite notable (e.g. max error  $\sim 10^{-7}$ , far from machine precision). This error decreases with increasing  $Q_{db}$ , and reaches the machine precision as  $Q_{db}$  becomes sufficiently large. Algorithm 2 requires a far smaller  $Q_{db}$  for the boundary-condition error to achieve machine accuracy, compared with Algorithm 1.
- For Algorithm 3,  $Q_{db}$  appears to have no effect on the boundary-condition error, which uniformly achieves the machine accuracy for all the  $Q_{db}$  values tested.

The results signify the importance of the numerical algorithm for evaluating the TFC expressions. Algorithm 3 is clearly the best for computing the NN solution among the three forms. The data also underscore the significance of  $Q_{db}$  when the optimal algorithm is unknown and only a suboptimal algorithm is available. While Algorithms 1 and 2 are suboptimal, thanks to the use of  $Q_{db}$ , they can nonetheless produce NN solutions with the boundary-condition errors at the machine precision.

When Neumann or Robin boundaries are involved, the TFC expressions can be evaluated in a manner analogous to Algorithm 3 to compute the NN solution. Algorithms 4 and 5 summarize the forms we have used in this paper to compute the NN solutions when NBC is imposed on  $\overline{BC}$  (Section 2.3.1) and when NBC is imposed on  $\overline{BC}$  and  $\overline{CD}$  (Section 2.3.2), respectively. These algorithms, with slight modification, can be used to compute the NN solutions when RBCs are involved.

### 5.3. Numerical tests with piecewise mixed BCs on a single boundary

The method developed in this work can handle certain mixed boundary conditions imposed piecewise on a single smooth boundary. In this case two different types of boundaries join smoothly at a junction, where the Jacobian matrix of the mapping



**Fig. 10.** Helmholtz equation (piecewise mixed BCs on a single boundary): Top row: crescent domain (Case #1), with piecewise Dirichlet/Neumann BCs imposed on the circular arc  $\overline{ABCD}$ . Bottom row: triangular domain (Case #2), with piecewise Dirichlet/Neumann BCs imposed on the edge  $\overline{BD}$ . Domain geometries and BC settings (left column), NN solution distributions (middle column), and point-wise absolute NN-solution errors (right column). Simulation parameters:  $R_m = 5.0$ ,  $Q = 80$  and  $M = 900$  for Case #1;  $R_m = 3.6$ ,  $Q = 70$  and  $M = 900$  for Case #2. The color bars in (b,e) denote the solution values, and those in (c,f) denote values of the absolute errors. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

**Table 10**

Helmholtz equation (piecewise mixed BCs on a single boundary): maximum and rms NN-solution errors over the domain, and the maximum and rms boundary-condition errors on different boundaries. The two cases correspond to those of Fig. 10.

|                                   | Case #1                 | Case #2                 |
|-----------------------------------|-------------------------|-------------------------|
| max solution-error (domain)       | $4.603 \times 10^{-3}$  | $1.209 \times 10^{-8}$  |
| rms solution-error (domain)       | $1.448 \times 10^{-3}$  | $1.996 \times 10^{-9}$  |
| max DBC-error ( $\overline{AB}$ ) | 0.0                     | 0.0                     |
| rms DBC-error ( $\overline{AB}$ ) | 0.0                     | 0.0                     |
| max NBC-error ( $\overline{BC}$ ) | $5.329 \times 10^{-15}$ | $6.040 \times 10^{-14}$ |
| rms NBC-error ( $\overline{BC}$ ) | $1.545 \times 10^{-15}$ | $9.945 \times 10^{-15}$ |
| max DBC-error ( $\overline{CD}$ ) | 0.0                     | 0.0                     |
| rms DBC-error ( $\overline{CD}$ ) | 0.0                     | 0.0                     |
| max DBC-error ( $\overline{AD}$ ) | $1.776 \times 10^{-15}$ | $8.882 \times 10^{-16}$ |
| rms DBC-error ( $\overline{AD}$ ) | $3.233 \times 10^{-16}$ | $4.060 \times 10^{-16}$ |

function becomes singular. This situation can be handled using the method discussed in Remark 2.3. In this appendix we use two examples of piecewise mixed BCs imposed on a single domain boundary to demonstrate the performance of the current method. We consider the Helmholtz problem from Section 3.1 in these examples.

The first example is for a crescent domain (Fig. 10a), which is the same as domain #2 from Section 3.1 but with a different configuration for the vertices and the imposed boundary conditions. This domain consists of the arcs of two unit circles centered at (1,0) and (2,0), respectively, and the four vertices are located at

$$\mathbf{x}_A = \left( \frac{3}{2}, -\frac{\sqrt{3}}{2} \right), \quad \mathbf{x}_B = \left( 2 - \frac{\sqrt{3}}{2}, -\frac{1}{2} \right), \quad \mathbf{x}_C = \left( 2 - \frac{\sqrt{3}}{2}, \frac{1}{2} \right), \quad \mathbf{x}_D = \left( \frac{3}{2}, \frac{\sqrt{3}}{2} \right).$$

Piecewise Dirichlet and Neumann BCs are imposed on the inner arc  $\overline{ABCD}$  and Dirichlet BC is imposed on the outer arc, as illustrated in Fig. 10(a).

The second example is for a triangular domain (Fig. 10d), which is the same as domain #5 from Section 3.1 but with a different configuration for the imposed boundary conditions. Neumann and Dirichlet BCs are imposed on the edge  $\overline{BD}$  in a piecewise fashion (C being the midpoint of  $\overline{BD}$ ), as illustrated in Fig. 10(d), while Dirichlet BCs are imposed on the other edges of the triangle.

The NN solutions for these two problems are illustrated by the other plots in Fig. 10, whose caption lists the simulation parameters used to obtain the results. These plots consist of distributions (contours) of the NN solutions (middle column) and distributions of the point-wise absolute errors (right column) in the domains. Table 10 provides the maximum/rms errors of the NN solution over

the domain and the maximum/rms BC errors on the domain boundaries for these two problems. The current method has evidently enforced the mixed BCs imposed piecewise on a single boundary to the machine precision.

We would like to mention a technical detail for computing the NN solution for these cases after the ELM network is trained. Since a Neumann (or Robin) boundary may join a Dirichlet boundary smoothly with the piecewise mixed BC configuration, the Jacobian matrix of the mapping function is singular at the junction, as discussed previously. This causes some subtlety and requires care when computing the NN solution on the test points located on the joining Dirichlet boundary. Let us use the crescent domain as an example to make the discussion concrete. The Jacobian matrix is singular at the junctions  $B$  and  $C$  (Fig. 10a) because the boundary is smooth there. As a result, when computing the NN solution on the test points located on  $\overline{AB}$  or  $\overline{CD}$ , the TFC constrained expression (Eq. (35) for this case) cannot be directly used, because the term  $PF^g(\xi, \eta)$  therein (specifically  $T_{BC}$  or  $S_{BC}$  in (34a)) requires the evaluation of the singular Jacobian matrix at  $B$  or  $C$ . This is not an issue in practice, however, because on  $\overline{AB}$  and  $\overline{CD}$  the TFC expression (35) is mathematically reduced to the Dirichlet conditions (Eqs. (28d) and (28e)). So when computing the NN solution on a given set of test points for these cases, one can first separate those boundary points located on  $\overline{AB}$  or  $\overline{CD}$  from the rest, and then compute the solution on these points using (28d) or (28e) directly. There is no such subtlety if the Neumann (or Robin boundary) joins a Dirichlet boundary at an angle.

## Data availability

Data will be made available on request.

## References

- [1] Yann LeCun, Yoshua Bengio, Geoffrey Hinton, Deep learning, *Nature* 521 (2015) 436–444.
- [2] I. Goodfellow, Y. Bengio, A. Courville, *Deep Learning*, The MIT Press, 2016.
- [3] G.E. Karniadakis, G. Kevrekidis, L. Lu, P. Perdikaris, S. Wang, L. Yang, Physics-informed machine learning, *Nat. Rev. Phys.* 3 (2021) 422–440.
- [4] H. Lee, I. Kang, Neural algorithms for solving differential equations, *J. Comput. Phys.* 91 (1990) 110–117.
- [5] A.J. Meade, A.A. Fernandez, The numerical solution of linear ordinary differential equations by feedforward neural networks, *Math. Comput. Model.* 19 (12) (1994) 1–25.
- [6] A.J. Meade, A.A. Fernandez, Solution of nonlinear ordinary differential equations by feedforward neural networks, *Math. Comput. Model.* 20 (9) (1994) 19–44.
- [7] 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.
- [8] R. Yentis, M.E. Zaghoul, VLSI implementation of locally connected neural network for solving partial differential equations, *IEEE Trans. Circuits Syst. I* 43 (1996) 687–690.
- [9] I.E. Lagaris, A.C. Likas, D.I. Fotiadis, Artificial neural networks for solving ordinary and partial differential equations, *IEEE Trans. Neural Netw.* 9 (1998) 987–1000.
- [10] G. Cybenko, Approximation by superpositions of a sigmoidal function, *Math. Control Signals Systems* 2 (1989) 303–314.
- [11] K. Hornik, M. Stinchcombe, H. White, Multilayer feedforward networks are universal approximators, *Neural Netw.* 2 (1989) 359–366.
- [12] K. Hornik, M. Stinchcombe, H. White, Universal approximation of an unknown mapping and its derivatives using multilayer feedforward networks, *Neural Netw.* 3 (1990) 551–560.
- [13] B.-N. Jiang, On the least squares method, *Comput. Methods Appl. Mech. Engrg.* 152 (1998) 239–257.
- [14] E.D. Eason, A review of least-squares methods for solving partial differential equations, *Internat. J. Numer. Methods Engrg.* 10 (1976) 1021–1046.
- [15] M. Raissi, P. Perdikaris, G.E. Karniadakis, Physics-informed neural networks: a deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations, *J. Comput. Phys.* 378 (2019) 686–707.
- [16] J. Sirignano, K. Spiliopoulos, DGM: A deep learning algorithm for solving partial differential equations, *J. Comput. Phys.* 375 (2018) 1339–1364.
- [17] W. E, B. Yu, The deep Ritz method: a deep learning-based numerical algorithm for solving variational problems, *Commun. Math. Stat.* 6 (2018) 1–12.
- [18] Yaohua Zang, Gang Bao, Xiaojing Ye, Haomin Zhou, Weak adversarial networks for high-dimensional partial differential equations, *J. Comput. Phys.* 411 (2020) 109409.
- [19] M. Ainsworth, J. Dong, Galerkin neural networks: a framework for approximating variational equations with error control, *SIAM J. Sci. Comput.* 43 (2021) A2474–A2501.
- [20] Y. Liao, P. Ming, Deep Nitsche method: deep ritz method with essential boundary conditions, *Commun. Comput. Phys.* 29 (2021) 1365–1384.
- [21] L. Lyu, Z. Zhang, M. Chen, J. Chen, MIM: a deep mixed residual method for solving high-order partial differential equations, *J. Comput. Phys.* 452 (2022) 110930.
- [22] K. Li, K. Tang, T. Wu, Q. Liao, D3M: A deep domain decomposition method for partial differential equations, *IEEE Access* 8 (2020) 5283–5294.
- [23] A.D. Jagtap, E. Kharazmi, G.E. Karniadakis, Conservative physics-informed neural networks on discrete domains for conservation laws: applications to forward and inverse problems, *Comput. Methods Appl. Mech. Engrg.* 365 (2020) 113028.
- [24] E.C. Cyr, M.A. Gulian, R.G. Patel, M. Perego, N.A. Trask, Robust training and initialization of deep neural networks: an adaptive basis viewpoint, *Proc. Mach. Learn. Res.* 107 (2020) 1–26.
- [25] Y. Wang, G. Lin, Efficient deep learning techniques for multiphase flow simulation in heterogeneous porous media, *J. Comput. Phys.* 401 (2020) 108968.
- [26] S. Wang, X. Yu, P. Perdikaris, When and why PINNs fail to train: a neural tangent kernel perspective, *J. Comput. Phys.* 449 (2022) 110768.
- [27] Lu Lu, Xuhui Meng, Zhiping Mao, George Em Karniadakis, DeepXDE: A deep learning library for solving differential equations, *SIAM Rev.* 63 (1) (2021) 208–228.
- [28] A.S. Krishnapriyan, A. Gholami, S. Zhe, R.M. Kirby, M.W. Mahoney, Characterizing possible failure modes in physics-informed neural networks, 2021, arXiv:2109.01050.
- [29] Y. Du, T.A. Zaki, Evolutional deep neural network, *Phys. Rev. E* 104 (2021) 045303.
- [30] K. Tang, X. Wan, Q. Liao, Adaptive deep density estimation for Fokker–Planck equations, *J. Comput. Phys.* 457 (2022) 111080.
- [31] H. Gao, M.J. Zahr, J.-X. Wang, Physics-informed graph neural Galerkin networks: A unified framework for solving PDE-governed forward and inverse problems, *Comput. Methods Appl. Mech. Engrg.* 390 (2022) 114502.
- [32] M. Penwarden, A.D. Jagtap, S. Zhe, G.E. Karniadakis, R.M. Kirby, A unified scalable framework for causal sweeping strategies for physics-informed neural networks (PINNs) and their temporal decompositions, *J. Comput. Phys.* 493 (2023) 112464.

- [33] L.D. McCleenny, U.M. Braga-Neto, Self-adaptive physics-informed neural networks, *J. Comput. Phys.* 474 (2023) 111722.
- [34] S. Zhang, H. Zhao, Y. Zhong, H. Zhou, Why shallow networks struggle with approximating and learning high frequency: a numerical study, *Inf. Inference: J. IMA* 14 (2025) iaaf022.
- [35] J. Bruna, B. Peherstorfer, E. Vanden-Eijnden, Neural Galerkin schemes with active learning for high-dimensional evolution equations, *J. Comput. Phys.* 496 (2024) 112588.
- [36] Y. Wang, C.-Y. Lai, Multi-stage neural networks: Function approximator of machine precision, *J. Comput. Phys.* 504 (2024) 112865.
- [37] Z. Aldirany, R. Cottetereau, M. Laforest, S. Prudhomme, Multi-level neural networks for accurate solutions of boundary-value problems, *Comput. Methods Appl. Mech. Engrg.* 419 (2024) 116666.
- [38] S. Wang, S. Sankaran, P. Perdikaris, Respecting causality for training physics-informed neural networks, *Comput. Methods Appl. Mech. Engrg.* 421 (2024) 116813.
- [39] S. Cuomo, V.S. Di Cola, F. Giampaolo, G. Rozza, M. Raissi, F. Piccialli, Scientific machine learning through physics-informed neural networks: Where we are and what's next, *J. Sci. Comput.* 92 (2022) 88.
- [40] C. Beck, M. Hutzenthaler, A. Jentzen, B. Kuckuck, An overview on deep learning-based approximation methods for partial differential equations, *Discret. Contin. Dyn. Syst. - B* 28 (6) (2023) 3697–3746.
- [41] E. Weinan, Jiequn Han, Arnulf Jentzen, Algorithms for solving high dimensional PDEs: from nonlinear Monte Carlo to machine learning, *Nonlinearity* 35 (1) (2021) 278.
- [42] Sifan Wang, Yujun Teng, Paris Perdikaris, Understanding and mitigating gradient flow pathologies in physics-informed neural networks, *SIAM J. Sci. Comput.* 43 (5) (2021) A3055–A3081, <http://dx.doi.org/10.1137/20M1318043>.
- [43] C. Rowan, K. Hampleman, K. Maute, A. Doostan, Boundary condition enforcement with PINNs: a comparative study and verification on 3D geometries, 2025, [arXiv:2512.14941](https://arxiv.org/abs/2512.14941).
- [44] I.E. Lagaris, A.C. Likas, D.G. Papageorgiou, Neural-network methods for boundary value problems with irregular boundaries, *IEEE Trans. Neural Netw.* 11 (2000) 1041–1049.
- [45] 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 (2009) 1221–1233.
- [46] H. Sheng, C. Yang, PFNN: A penalty-free neural network method for solving a class of second-order boundary-value problems on complex geometries, *J. Comput. Phys.* 428 (2021) 110085.
- [47] L. Lyu, K. Wu, R. Du, J. Chen, Enforcing exact boundary and initial conditions in the deep mixed residual method, *CSIAM Trans. Appl. Math.* 2 (4) (2021) 748–775, <http://dx.doi.org/10.4208/csiam-am.2020-0089>.
- [48] Lu Lu, Raphaël Pestourie, Wenjie Yao, Zhiping Wang, Francisco Verdugo, Steven G. Johnson, Physics-informed neural networks with hard constraints for inverse design, *SIAM J. Sci. Comput.* 43 (6) (2021) B1105–B1132, <http://dx.doi.org/10.1137/21M1397908>.
- [49] S. Liu, Z. Hao, C. Ying, H. Su, J. Zhu, Z. Cheng, A unified hard-constraint framework for solving geometrically complex PDEs, 2022, [arXiv:2210.03526](https://arxiv.org/abs/2210.03526).
- [50] P. Roy, S.T. Castonguay, Exact enforcement of temporal continuity in sequential physics-informed neural networks, *Comput. Methods Appl. Mech. Engrg.* 430 (2024) 117197.
- [51] M.-C. Lai, Y. Song, X. Yuan, H. Yue, T. Zeng, The hard-constraint PINNs for interface optimal control problems, *SIAM J. Sci. Comput.* 47 (2025) C601–C629.
- [52] N. Sukumar, A. Srivastava, Exact imposition of boundary conditions with distance functions in physics-informed deep neural networks, *Comput. Methods Appl. Mech. Engrg.* 389 (2022) 114333, <http://dx.doi.org/10.1016/j.cma.2021.114333>.
- [53] Stefano Berrone, Claudio Canuto, Manuela Pintore, N. Sukumar, Enforcing Dirichlet boundary conditions in physics-informed neural networks and variational physics-informed neural networks, *Heliyon* 9 (8) (2023) e18820, <http://dx.doi.org/10.1016/j.heliyon.2023.e18820>.
- [54] J. Wang, Y.L. Mo, B. Izzuddin, C.-W. Kim, Exact Dirichlet boundary physics-informed neural network EPINN for solid mechanics, *Comput. Methods Appl. Mech. Engrg.* 414 (2023) 116184.
- [55] Xi'an Li, Jiaxin Deng, Jinran Wu, Shaotong Zhang, Weide Li, You-Gan Wang, Physical informed neural networks with soft and hard boundary constraints for solving advection-diffusion equations using Fourier expansions, *Comput. Math. Appl.* 159 (2024) 60–75, <http://dx.doi.org/10.1016/j.camwa.2024.01.021>.
- [56] R.J. Gladstone, M.A. Nabian, N. Sukumar, A. Srivastava, FO-PINN: A first-order formulation for physics-informed neural networks, *Eng. Anal. Bound. Elem.* 174 (2025) 106161.
- [57] N. Goschel, S. Goschel, D. Ruprecht, Enforcing boundary conditions for physics-informed neural operators, 2025, [arXiv:2510.24557](https://arxiv.org/abs/2510.24557).
- [58] N. Sukumar, R. Roy, A wachspress-based transfinite formulation for exactly enforcing Dirichlet boundary conditions on convex polygonal domains in physics-informed neural networks, 2026, [arXiv:2601.01756](https://arxiv.org/abs/2601.01756).
- [59] D. Mortari, The theory of connections: connecting points, *Mathematics* 5 (2017) 57.
- [60] D. Mortari, C. Leake, The multivariate theory of connections, *Mathematics* 7 (2019) 296.
- [61] Carl Leake, Daniele Mortari, Deep theory of functional connections: A new method for estimating the solutions of partial differential equations, *Mach. Learn. Knowl. Extr.* 2 (1) (2020) 37–55, <http://dx.doi.org/10.3390/make2010003>.
- [62] E. Schiassi, R. Furfaro, C. Leake, M. De Florio, H. Johnson, D. Mortari, Extreme theory of functional connections: a fast physics-informed neural network method for solving ordinary and partial differential equations, *Neurocomputing* 457 (2021) 334–356.
- [63] D. Mortari, D. Arnas, Bijective mapping analysis to extend the theory of functional connections to non-rectangular 2-dimensional domains, *Mathematics* 8 (2020) 1593.
- [64] A.A. Thiruthummal, S. Shelyag, E.-J. Kim, Extremization to fine tune physics informed neural networks for solving boundary value problems, *Commun. Nonlinear Sci. Numer. Simul.* 137 (2024) 108129.
- [65] Y. Wang, S. Dong, An extreme learning machine based method for computational PDEs in higher dimensions, *Comput. Methods Appl. Mech. Engrg.* 418 (2024) 116578.
- [66] S. Dong, N. Ni, A method for representing periodic functions and enforcing exactly periodic boundary conditions with deep neural networks, *J. Comput. Phys.* 435 (2021) 110242.
- [67] Meng Li, You Yang, A structure-preserving PINN with embedded periodic boundary layer and adaptively enforced initial conditions for geometric flows, *Comput. Phys. Comm.* 311 (2025) 109762, <http://dx.doi.org/10.1016/j.cpc.2025.109762>.
- [68] Ehsan Kharazmi, Zhongqiang Zhang, George Em Karniadakis, hp-VPINNs: Variational physics-informed neural networks with domain decomposition, *Comput. Methods Appl. Mech. Engrg.* 374 (2021) 113547, <http://dx.doi.org/10.1016/j.cma.2020.113547>.
- [69] S.G. Makridakis, A. Pim, T. Pryer, A deep Uzawa–Lagrange multiplier approach for boundary conditions in PINNs and deep Ritz methods, *J. Machine Learn.* 4 (3) (2025) 169–191.
- [70] C. Straub, P. Brendel, V. Medvedev, A. Roskopf, Hard-constraining Neumann boundary conditions in physics-informed neural networks via Fourier feature embedding, 2025, [arXiv:2504.01093](https://arxiv.org/abs/2504.01093).
- [71] W.J. Gordon, Blending-function methods of bivariate and multivariate interpolation and approximation, *SIAM J. Numer. Anal.* 8 (1971) 158–177.
- [72] W.J. Gordon, C.A. Hall, Construction of curvilinear coordinate systems and applications to mesh generation, *Internat. J. Numer. Methods Engrg.* 7 (1973) 461–477.
- [73] S. Dong, Z. Li, Local extreme learning machines and domain decomposition for solving linear and nonlinear partial differential equations, *Comput. Methods Appl. Mech. Engrg.* 387 (2021) 114129, (also [arXiv:2012.02895](https://arxiv.org/abs/2012.02895)).

- [74] S. Dong, Z. Li, A modified batch intrinsic plasticity method for pre-training the random coefficients of extreme learning machines, *J. Comput. Phys.* 445 (2021) 110585, (also [arXiv:2103.08042](#)).
- [75] S. Dong, J. Yang, On computing the hyperparameter of extreme learning machines: algorithms and applications to computational PDEs, and comparison with classical and high-order finite elements, *J. Comput. Phys.* 463 (2022) 111290, (also [arXiv:2110.14121](#)).
- [76] N. Ni, S. Dong, Numerical computation of partial differential equations by hidden-layer concatenated extreme learning machine, *J. Sci. Comput.* 95 (2023) 35.
- [77] S. Dong, Y. Wang, A method for computing inverse parametric PDEs with random-weight neural networks, *J. Comput. Phys.* 489 (2023) 112263.
- [78] S. Dong, N. Ni, Learning the exact time integration algorithm for initial value problems by randomized neural networks, *Next Res.* 2 (2025) 100623.
- [79] A. Bjorck, *Numerical Methods for Least Squares Problems*, SIAM, 1996.
- [80] S. Panghal, M. Kumar, Optimization free neural network approach for solving ordinary and partial differential equations, *Eng. Comput.* 37, 2989–3002.
- [81] V. Dwivedi, B. Srinivasan, Physics informed extreme learning machine (PIELM) – a rapid method for the numerical solution of partial differential equations, *Neurocomputing* 391 (2020) 96–118.
- [82] F. Calabro, G. Fabiani, C. Siettos, Extreme learning machine collocation for the numerical solution of elliptic PDEs with sharp gradients, *Comput. Methods Appl. Mech. Engrg.* 387 (2021) 114188.
- [83] G. Fabiani, F. Calabro, L. Russo, C. Siettos, Numerical solution and bifurcation analysis of nonlinear partial differential equations with extreme learning machines, *J. Sci. Comput.* 89 (2021) 44.
- [84] J. Chen, X. Chi, W. E, Z. Yang, Bridging traditional and machine learning-based algorithms for solving PDEs: the random feature method, *J. Mach. Learn.* 1 (3) (2022) 268–298.
- [85] H.D. Quan, H.T. Huynh, Solving partial differential equation based on extreme learning machine, *Math. Comput. Simul.* 205 (2023) 697–708.
- [86] G. Fabiani, E. Galaris, L. Russo, C. Siettos, Parsimonious physics-informed random projection neural networks for initial value problems of ODEs and index-1 DAEs, *Chaos* 33 (2023) 043128.
- [87] J. Sun, S. Dong, F. Wang, Local randomized neural networks with discontinuous Galerkin methods for partial differential equations, *J. Comput. Appl. Math.* 445 (2024) 115830.
- [88] Z. Zhang, F. Bao, L. Ju, G. Zhang, Transferable neural networks for partial differential equations, *J. Sci. Comput.* 99 (2024) 2.
- [89] G. Fiabiani, I.G. Kevrekidis, C. Siettos, A.N. Yannacopoulos, RandONets: shallow networks with random projections for learning linear and nonlinear operators, *J. Comput. Phys.* 520 (2025) 113433.
- [90] D.E. De Falco, E. Schiassi, F. Calabro, Least squares with equality constraints extreme learning machines for the resolution of PDEs, *J. Comput. Phys.* 547 (2026) 114553.
- [91] W. Fleming, *Functions of Several Variables*, second ed., Springer-Verlag, 1977.
- [92] S. Dong, J. Yang, Numerical approximation of partial differential equations by a variable projection method with artificial neural networks, *Comput. Methods Appl. Mech. Engrg.* 398 (2022) 115284, (also [arXiv:2201.09989](#)).
- [93] G.-B. Huang, Q.-Y. Zhu, C.-K. Siew, Extreme learning machine: theory and applications, *Neurocomputing* 70 (2006) 489–501.