11-15-2022

Training RBF Neural Networks For the Solution of Elliptic Boundary Value Problems

Andreas Karageorghis
University of Cyprus, andreask@ucy.ac.cy

C.S. Chen
University of Southern Mississippi, cs.chen@usm.edu

TRAINING RBF NEURAL NETWORKS FOR THE SOLUTION OF ELLIPTIC BOUNDARY VALUE PROBLEMS

ANDREAS KARAGEORGHIS AND C. S. CHEN

Abstract. We propose a radial basis function (RBF) neural network method for solving two- and three–dimensional second and fourth order elliptic boundary value problems (BVPs). The neural network in question is trained by minimizing a nonlinear least squares functional, thus determining the optimal values of the various RBF parameters involved. The functional minimization is carried out using standard MATLAB© software efficiently. Several numerical experiments are presented to demonstrate the efficacy of the proposed method.

1. Introduction

In recent decades, various meshless methods have been developed for solving boundary value problems (BVPs) in complicated geometries in two (2D) and three dimensions (3D). The main attraction of meshless methods is the simplicity of the solution process as no tedious domain or boundary discretization is required. Among all meshless methods, radial basis function collocation methods (RBFCMs) [4, 8, 18, 32] have become particularly popular. Traditionally, the radial basis function (RBF) centres are placed inside the domain. In [5, 25], however, the domain in which the RBF centres were placed was extended outside the domain of the BVP in question resulting in better performance of RBFCMs. In general, for elliptic partial differential equations (PDEs), in the RBFCM solution process both the PDE and the boundary conditions (BCs) are collocated at selected collocation points to form a linear system of equations $\mathbf{Ax} = \mathbf{b}$. Once the RBF weights $\mathbf{x}$ are determined by a linear solver, the approximate solution can be calculated anywhere in the domain. The above solution process seems simple and straightforward, but the determination of an appropriate value of the RBF shape parameter (or appropriate values in case shape parameters are varied), which greatly affects the accuracy of the approximation, is a challenge and needs to be addressed prior the RBFCM solution process.

In an alternative approach, RBF neural networks have, in recent years, been used extensively for function approximation and the solution of BVPs [1–3, 6, 7, 9, 10, 12, 13, 16, 17, 23, 26, 27, 30], see also [31, Sections 4.2-4.3] and [14, Section 5.3].

It is the purpose of this paper to use an RBF neural network method for the solution of second and fourth order BVPs in 2D and 3D. The formulation of the proposed method is motivated by

Date: April 27, 2022.

2010 *Mathematics Subject Classification.* Primary 65N35; Secondary 65H10.

Key words and phrases. Radial basis functions, neural networks, collocation, elliptic boundary value problems, nonlinear minimization.

the approach of Gorbachenko and his co-workers [1–3, 7, 10–12] and the training of the network is achieved by minimizing a nonlinear cost functional. The minimization is carried out using the MATLAB© optimization toolbox function `lsqnonlin` which is a nonlinear least–squares solver. In contrast to traditional RBFCMs in which the (troublesome) appropriate shape parameter value needs to be determined prior to the solution procedure, the proposed approach allows us to find the optimal locations of the RBF centres, shape parameter(s) values, and RBF centre weights simultaneously through the use of `lsqnonlin`. In the application of this function we provide the Jacobian of the corresponding nonlinear system which leads to substantial computational time savings enabling us to solve 3D problems in complex geometries efficiently and accurately.

The paper is organized as follows. In Section 2, we briefly describe how to use the nonlinear least-squares solver `lsqnonlin`. In Section 3, the formulation of the proposed neural network method for BVPs for second and fourth order PDEs is given. Four numerical examples in 2D and 3D with irregular or non–smooth domains are studied in Section 4, illustrating the effectiveness of the current approach. Finally, in Section 5, some conclusions and ideas for future work are provided.

2. Nonlinear minimization

As will be explained in the sequel, the satisfaction of the PDE and BCs of a second or fourth order BVP in 2D and 3D at a given set of sampling points by an RBF approximation yields systems of nonlinear equations

$$\mathbf{F}(\mathbf{w}) := \begin{bmatrix} \mathtt{F}_1(\mathbf{w}) \\ \mathtt{F}_2(\mathbf{w}) \\ \vdots \\ \mathtt{F}_M(\mathbf{w}) \end{bmatrix} = \mathbf{0}\,, \tag{2.1}$$

in the N unknowns $(\mathtt{w}_1, \mathtt{w}_2, \ldots, \mathtt{w}_N)^{\mathrm{T}} = \mathbf{w}$ where, clearly, $M \geq N$. The solution of system (2.1) may be recast as a nonlinear least squares minimization problem for the functional

$$\mathtt{S}(\mathbf{w}) := \sum_{m=1}^{M} \mathtt{F}_m^2, \tag{2.2}$$

which we shall solve using the MATLAB© [28] optimization toolbox function `lsqnonlin`. This uses one of two algorithms, namely, a trust–region reflective algorithm or the Levenberg–Marquardt method.

In `lsqnonlin`, the functions $\mathtt{F}_m$, $m = 1, \ldots, M$, in (2.1) must be provided. Moreover, there is the option of not providing the Jacobian of the system which is then calculated internally using finite differences. Otherwise, the user may provide the exact Jacobian $\mathtt{J} = (\mathtt{J}_{m,n})_{m=1,n=1}^{M,N}$ with

$$\mathtt{J}_{m,n} = \frac{\partial \mathtt{F}_m}{\partial \mathtt{w}_n}, \quad m = 1, \ldots, M,\ n = 1, \ldots, N.$$

When the exact Jacobian is not given by the user, the function options are defined via the call

```
options=optimoptions(@lsqnonlin,'Display','iter','MaxFunEvals',2000000, ...
'MaxIter',mm,'TolFun',TOL1,'TolX',TOL2,'SpecifyObjectiveGradient',false)
```

while the function is called as
`[w,resnorm,residual,exitflag,output] = lsqnonlin(@f1,w0,lb,ub,options)`
The vectors `lb` and `ub` are user–specified and define lower and upper bounds, respectively, for the variables `w` while the user–specified vector `w0` contains the initial values of `w`.
The solution process of `lsqnonlin` will stop if any of the following conditions is met:

(1) The number of iterations exceeds `mm`.
(2) The final change in the sum of squares relative to its initial values is less than the value of the function tolerance `TOL1`.
(3) The relative size of the current step is less than the value of the step size tolerance `TOL2`.

The `function f=f1(w)` contains the definition of the functions $\mathtt{F}_m$, $m = 1, \ldots, M$ (`f`).
When the exact Jacobian is given by the user, the option `'SpecifyObjectiveGradient'` becomes `true` and `function [f,J]=f1(w)` contains the definition of both $\mathtt{F}_m$, $m = 1, \ldots, M$ (`f`), and the exact Jacobian $\mathtt{J}_{m,n}$, $m = 1, \ldots, M$, $n = 1, \ldots, N$ (`J`).

3. RBF neural network method

3.1. **Second order problems.**

3.1.1. *The problem.* We first consider the BVP in $\mathbb{R}^2$ or $\mathbb{R}^3$ consisting of the PDE

$$\mathcal{L}u = f \quad \text{in} \quad \Omega, \tag{3.1a}$$

subject to the BC

$$\mathcal{B}u = g \quad \text{on} \quad \partial\Omega, \tag{3.1b}$$

where in (3.1a) $\mathcal{L}$ is a second order elliptic operator and the operator $\mathcal{B}$ describes the BC.

3.1.2. *The method.* We shall now follow the approach of the key papers [7,12]. In this case learning can be viewed as a regression problem and, more specifically, as an interpolation problem [14] which can be solved in a supervised fashion. The solution u of BVP (3.1) is approximated by the RBF network [18]

$$u_{\mathsf{N}}(\boldsymbol{x}) = \sum_{\mathsf{n}=1}^{\mathsf{N}} a_{\mathsf{n}}\, \Phi(c_{\mathsf{n}}, r_{\mathsf{n}}), \qquad \boldsymbol{x} \in \overline{\Omega}, \tag{3.2}$$

where $r_{\mathsf{n}} = |\boldsymbol{x} - \mathbf{x}_{\mathsf{n}}|$. Note that $\boldsymbol{x} = (x, y)$ in $\mathbb{R}^2$, $\boldsymbol{x} = (x, y, z)$ in $\mathbb{R}^3$, and $\mathbf{x}_{\mathsf{n}} = (\mathrm{x}_{\mathsf{n}}, \mathrm{y}_{\mathsf{n}})$ in $\mathbb{R}^2$ and $\mathbf{x}_{\mathsf{n}} = (\mathrm{x}_{\mathsf{n}}, \mathrm{y}_{\mathsf{n}}, \mathrm{z}_{\mathsf{n}})$ in $\mathbb{R}^3$. Each RBF $\Phi(c_{\mathsf{n}}, r_{\mathsf{n}})$ is dependent on a shape parameter c_{n}. Thus, each $\Phi(c_{\mathsf{n}}, r_{\mathsf{n}})$ is linked to a point $\mathbf{x}_{\mathsf{n}}$ and a shape parameter value c_{n}. The set of (distinct) points $\{\mathbf{x}_{\mathsf{n}}\}_{\mathsf{n}=1}^{\mathsf{N}}$ is the set of centers. Usually, the values of the shape parameters are taken to be equal a preassigned value c, i.e. $c_1 = c_2 = \ldots = c_{\mathsf{N}} = c$. The determination of the shape parameter optimal value is a major issue in RBF approximation applications. In the current approach however, as in [15, 22], we shall take distinct and unknown shape parameter values $c_{\mathsf{n}}, \mathsf{n} = 1, \ldots, \mathsf{N}$. A comparison of the performance of the single shape parameter approach versus the variable shape parameter approach for the solution on nonlinear second and fourth order BVPs can be found in [15].

The interior centres $\{\mathbf{x}_{\mathsf{n}}^0\}_{\mathsf{n}=1}^{\mathsf{N}_{\rm int}}$, and the boundary centres $\{\mathbf{x}_{\mathsf{n}}^0\}_{\mathsf{n}=\mathsf{N}_{\rm int}+1}^{\mathsf{N}_{\rm int}+\mathsf{N}_{\rm bry}}$ comprise the initial set of centres where $\mathsf{N} = \mathsf{N}_{\rm int} + \mathsf{N}_{\rm bry}$. In the current approach we shall assume that the coordinates of the centres $\{\mathbf{x}_{\mathsf{n}}\}_{\mathsf{n}=1}^{\mathsf{N}}$ are unknown. In addition to the centres, we need to provide the sampling or training (collocation) points $\{\boldsymbol{x}_{\mathsf{m}}\}_{\mathsf{m}=1}^{\mathsf{M}} \in \overline{\Omega}$. The interior collocation points $\{\boldsymbol{x}_{\mathsf{m}}\}_{\mathsf{m}=1}^{\mathsf{M}_{\rm int}}$ and the boundary collocation points $\{\boldsymbol{x}_{\mathsf{m}}\}_{\mathsf{m}=\mathsf{M}_{\rm int}+1}^{\mathsf{M}_{\rm int}+\mathsf{M}_{\rm bry}}$ comprise the set of collocation points where $\mathsf{M} = \mathsf{M}_{\rm int} + \mathsf{M}_{\rm bry}$.

The total set of network parameters consists of the coefficients $\{a_{\mathsf{n}}\}_{\mathsf{n}=1}^{\mathsf{N}}$ and the shape parameter values $\{c_{\mathsf{n}}\}_{\mathsf{n}=1}^{\mathsf{N}}$ in equation (3.2) as well as the coordinates of the centres $\{\mathbf{x}_{\mathsf{n}}\}_{\mathsf{n}=1}^{\mathsf{N}}$. These are calculated by training the network to satisfy the following equations at the M sampling or training (collocation) points

$$\mathcal{L}u_{\mathsf{N}}(\boldsymbol{x}_{\mathsf{m}}) = f(\boldsymbol{x}_{\mathsf{m}}), \quad \mathsf{m} = 1, \ldots, \mathsf{M}_{\rm int}, \tag{3.3a}$$

$$\mathcal{B}u_{\mathsf{N}}(\boldsymbol{x}_{\mathsf{m}}) = g(\boldsymbol{x}_{\mathsf{m}}), \quad \mathsf{m} = \mathsf{M}_{\rm int}+1, \ldots, \mathsf{M}_{\rm int}+\mathsf{M}_{\rm bry}. \tag{3.3b}$$

System (3.3) yields M equations in $4\mathsf{N}$ unknowns in 2D and $5\mathsf{N}$ unknowns in 3D, namely, the unknown coefficients $\boldsymbol{a} = [a_1, a_2, \ldots, a_{\mathsf{N}}]^T$, the shape parameters $\boldsymbol{c} = [c_1, c_2, \ldots, c_{\mathsf{N}}]^T$ and the centre coordinates $\mathbf{x} = [\mathrm{x}_1, \mathrm{x}_2, \ldots, \mathrm{x}_{\mathsf{N}}]^T$, $\mathbf{y} = [\mathrm{y}_1, \mathrm{y}_2, \ldots, \mathrm{y}_{\mathsf{N}}]^T$ in 2D. In 3D we have the centre coordinates $\mathbf{x} = [\mathrm{x}_1, \mathrm{x}_2, \ldots, \mathrm{x}_{\mathsf{N}}]^T$, $\mathbf{y} = [\mathrm{y}_1, \mathrm{y}_2, \ldots, \mathrm{y}_{\mathsf{N}}]^T$ and $\mathbf{z} = [\mathrm{z}_1, \mathrm{z}_2, \ldots, \mathrm{z}_{\mathsf{N}}]^T$.

$$\text{In the sequel, we shall denote by } \mathbf{X} \text{ the vector } \begin{bmatrix} \mathbf{x} \\ \mathbf{y} \end{bmatrix} \text{ in 2D and } \begin{bmatrix} \mathbf{x} \\ \mathbf{y} \\ \mathbf{z} \end{bmatrix} \text{ in 3D.} \tag{3.4}$$

For the number of equations to be at least as large as the number of unknowns, we take $\mathsf{M} \geq 4\mathsf{N}$ in 2D and $\mathsf{M} \geq 5\mathsf{N}$ in 3D.

Because all the parameters except the RBF coefficients $\boldsymbol{a}$ appear nonlinearly, the M equations (3.3a)–(3.3b) generate a nonlinear system (2.1) where

$$\mathbf{F}(\boldsymbol{a}, \boldsymbol{c}, \mathbf{X}) := \begin{bmatrix} \mathtt{F}_1 \\ \mathtt{F}_2 \\ \vdots \\ \mathtt{F}_{\mathsf{M}} \end{bmatrix} = \begin{bmatrix} \mathcal{L}u_{\mathsf{N}}(\boldsymbol{x}_1) - f(\boldsymbol{x}_1) \\ \vdots \\ \mathcal{L}u_{\mathsf{N}}(\boldsymbol{x}_{\mathsf{M}_{\rm int}}) - f(\boldsymbol{x}_{\mathsf{M}_{\rm int}}) \\ \sqrt{\lambda}\ (\mathcal{B}u_{\mathsf{N}}(\boldsymbol{x}_{\mathsf{M}_{\rm int}+1}) - g(\boldsymbol{x}_{\mathsf{M}_{\rm int}+1})) \\ \vdots \\ \sqrt{\lambda}\ (\mathcal{B}u_{\mathsf{N}}(\boldsymbol{x}_{\mathsf{M}}) - g(\boldsymbol{x}_{\mathsf{M}})) \end{bmatrix} = \mathbf{0}. \tag{3.5}$$

In (3.5) λ is a fitted penalty factor accounting for the weight of the boundary conditions [12].

The MATLAB© function `lsqnonlin` will be used to minimize the functional [12] (the details are given in Section 2),

$$\mathtt{S}(\boldsymbol{a}, \boldsymbol{c}, \mathbf{X}) := \sum_{\mathsf{m}=1}^{\mathsf{M}_{\rm int}} \left[\mathcal{L}u_{\mathsf{N}}(\boldsymbol{x}_{\mathsf{m}}) - f(\boldsymbol{x}_{\mathsf{m}})\right]^2 + \lambda \sum_{\mathsf{m}=\mathsf{M}_{\rm int}+1}^{\mathsf{M}} \left[\mathcal{B}u_{\mathsf{N}}(\boldsymbol{x}_{\mathsf{m}}) - g(\boldsymbol{x}_{\mathsf{m}})\right]^2, \tag{3.6}$$

thus training the network effectively by gradient descent learning. Lower and upper bounds on the elements of the vector of unknowns $[\boldsymbol{a}, \boldsymbol{c}, \mathbf{X}]^T$ may be imposed in `lsqnonlin`. This option allows the user to impose bounds on the shape parameter values which can be useful. For the Jacobian analytical expression, in 2D we have,

$$\mathsf{J}(\boldsymbol{a}, \boldsymbol{c}, \mathbf{X}) := (\mathsf{J}_{\mathsf{m},\mathsf{n}})_{\mathsf{m}=1,\mathsf{n}=1}^{\mathsf{M},4\mathsf{N}}, \tag{3.7}$$

where

$$(\mathsf{J}_{\mathsf{m},\mathsf{n}})_{\mathsf{m}=1,\mathsf{n}=1}^{\mathsf{M}_{\mathrm{int}},\mathsf{N}} = \frac{\partial \mathcal{L} u_{\mathsf{N}}(\boldsymbol{x}_{\mathsf{m}})}{\partial a_{\mathsf{n}}}, \qquad (\mathsf{J}_{\mathsf{m},\mathsf{N}+\mathsf{n}})_{\mathsf{m}=1,\mathsf{n}=1}^{\mathsf{M}_{\mathrm{int}},\mathsf{N}} = \frac{\partial \mathcal{L} u_{\mathsf{N}}(\boldsymbol{x}_{\mathsf{m}})}{\partial c_{\mathsf{n}}},$$

$$(\mathsf{J}_{\mathsf{m},2\mathsf{N}+\mathsf{n}})_{\mathsf{m}=1,\mathsf{n}=1}^{\mathsf{M}_{\mathrm{int}},\mathsf{N}} = \frac{\partial \mathcal{L} u_{\mathsf{N}}(\boldsymbol{x}_{\mathsf{m}})}{\partial \mathrm{x}_{\mathsf{n}}}, \qquad (\mathsf{J}_{\mathsf{m},3\mathsf{N}+\mathsf{n}})_{\mathsf{m}=1,\mathsf{n}=1}^{\mathsf{M}_{\mathrm{int}},\mathsf{N}} = \frac{\partial \mathcal{L} u_{\mathsf{N}}(\boldsymbol{x}_{\mathsf{m}})}{\partial \mathrm{y}_{\mathsf{n}}},$$

$$(\mathsf{J}_{\mathsf{m},\mathsf{n}})_{\mathsf{m}=\mathsf{M}_{\mathrm{int}}+1,\mathsf{n}=1}^{\mathsf{M},\mathsf{N}} = \sqrt{\lambda}\, \frac{\partial \mathcal{B} u_{\mathsf{N}}(\boldsymbol{x}_{\mathsf{m}})}{\partial a_{\mathsf{n}}}, \qquad (\mathsf{J}_{\mathsf{m},\mathsf{N}+\mathsf{n}})_{\mathsf{m}=\mathsf{M}_{\mathrm{int}}+1,\mathsf{n}=1}^{\mathsf{M},\mathsf{N}} = \sqrt{\lambda}\, \frac{\partial \mathcal{B} u_{\mathsf{N}}(\boldsymbol{x}_{\mathsf{m}})}{\partial c_{\mathsf{n}}}$$

$$(\mathsf{J}_{\mathsf{m},2\mathsf{N}+\mathsf{n}})_{\mathsf{m}=\mathsf{M}_{\mathrm{int}}+1,\mathsf{n}=1}^{\mathsf{M},\mathsf{N}} = \sqrt{\lambda}\, \frac{\partial \mathcal{B} u_{\mathsf{N}}(\boldsymbol{x}_{\mathsf{m}})}{\partial \mathrm{x}_{\mathsf{n}}}, \quad (\mathsf{J}_{\mathsf{m},3\mathsf{N}+\mathsf{N}+\mathsf{n}})_{\mathsf{m}=\mathsf{M}_{\mathrm{int}}+1,\mathsf{n}=1}^{\mathsf{M},\mathsf{N}} = \sqrt{\lambda}\, \frac{\partial \mathcal{B} u_{\mathsf{N}}(\boldsymbol{x}_{\mathsf{m}})}{\partial \mathrm{y}_{\mathsf{n}}}.$$

In 3D

$$\mathsf{J}(\boldsymbol{a}, \boldsymbol{c}, \mathbf{X}) := (\mathsf{J}_{\mathsf{m},\mathsf{n}})_{\mathsf{m}=1,\mathsf{n}=1}^{\mathsf{M},5\mathsf{N}},$$

where, in addition to the definitions in (3.7), we have

$$(\mathsf{J}_{\mathsf{m},4\mathsf{N}+\mathsf{n}})_{\mathsf{m}=1,\mathsf{n}=1}^{\mathsf{M}_{\mathrm{int}},\mathsf{N}} = \frac{\partial \mathcal{L} u_{\mathsf{N}}(\boldsymbol{x}_{\mathsf{m}})}{\partial \mathrm{z}_{\mathsf{n}}}, \quad (\mathsf{J}_{\mathsf{m},4\mathsf{N}+\mathsf{N}+\mathsf{n}})_{\mathsf{m}=\mathsf{M}_{\mathrm{int}}+1,\mathsf{n}=1}^{\mathsf{M},\mathsf{N}} = \sqrt{\lambda}\, \frac{\partial \mathcal{B} u_{\mathsf{N}}(\boldsymbol{x}_{\mathsf{m}})}{\partial \mathrm{z}_{\mathsf{n}}}.$$

The RBF network described above consists of two layers. In the first layer, we take the input training points $\{\boldsymbol{x}_{\mathsf{m}}\}_{\mathsf{m}=1}^{\mathsf{M}}$ and calculate the RBFs at these points which is a nonlinear transformation. In the second layer we construct the weighted sum of these RBFs (which is a linear transformation) yielding the output (3.2), see Figure 1, and also [12] and [14, Section 5.3].

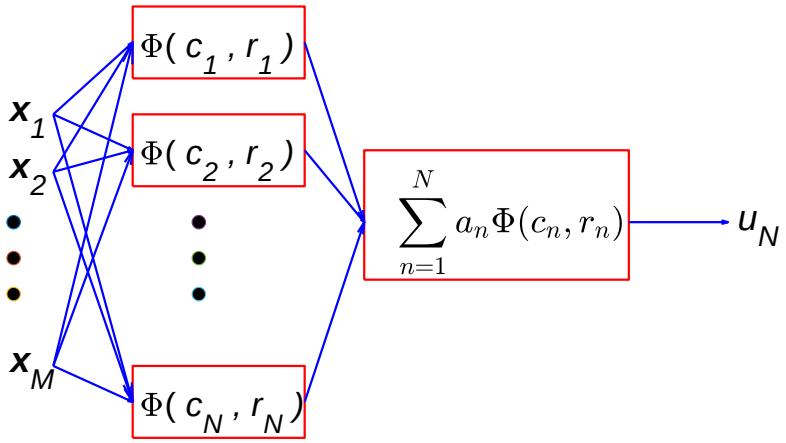


FIGURE 1. Structure of RBF network.

3.1.3. *Initial centre locations.* Like the locations of the collocation points, the initial locations of the centres are usually taken to be in $\overline{\Omega}$. However, it has been observed that taking the initial locations of the centres in a magnification of $\overline{\Omega}$ by a factor $\eta > 1$ (see e.g. [5, 22]) produced more accurate results. For 2D star–shaped domains whose boundary $\partial\Omega$ is described parametrically by

$$x = r(\vartheta)\cos\vartheta,\ y = r(\vartheta)\sin\vartheta, \quad 0 \le \vartheta \le 2\pi,$$

the centres are taken as

$$\tilde{\mathbf{x}}_{\mathsf{n}}^0 = \eta\,\mathbf{x}_{\mathsf{n}}^0,\ \mathsf{n} = 1, \ldots, \mathsf{N}. \tag{3.8}$$

In a general domain Ω with centre $\mathbf{x}_{\rm c}$, the centres are taken as

$$\tilde{\mathbf{x}}_{\mathsf{n}}^0 = \eta\,(\mathbf{x}_{\mathsf{n}}^0 - \mathbf{x}_{\rm c}) + \mathbf{x}_{\rm c},\ \mathsf{n} = 1, \ldots, \mathsf{N}. \tag{3.9}$$

3.2. **Fourth order problems.**

3.2.1. *The problem.* We also consider the BVP in $\mathbb{R}^2$ or $\mathbb{R}^3$ consisting of the PDE

$$\mathcal{L}u \;=\; f \quad \text{in} \quad \Omega, \tag{3.10a}$$

and the BCs

$$\mathcal{B}_1\,u \;=\; g_1 \quad \text{and} \quad \mathcal{B}_2\,u \;=\; g_2 \quad \text{on} \quad \partial\Omega, \tag{3.10b}$$

where in (3.10a) $\mathcal{L}$ is a fourth order elliptic operator and the operators $\mathcal{B}_1, \mathcal{B}_2$ in (3.10b) describe the BCs.

3.2.2. *The method.* The solution u of BVP (3.10) is again approximated by (3.2). The selection of the sampling points $\{\boldsymbol{x}_{\mathsf{m}}\}_{\mathsf{m}=1}^{\mathsf{M}_{\rm int}+\mathsf{M}_{\rm bry}} \in \overline{\Omega}$ is carried out as in second order problems, see Section 3.1.2, with $\mathsf{M}_{\rm int}$ interior and $\mathsf{M}_{\rm bry}$ boundary points. In a similar fashion, we select the initial locations of the interior centres $\{\mathbf{x}_{\mathsf{n}}^0\}_{\mathsf{n}=1}^{\mathsf{N}_{\rm int}}$ and the boundary centres $\{\mathbf{x}_{\mathsf{n}}^0\}_{\mathsf{n}=\mathsf{N}_{\rm int}+1}^{\mathsf{N}_{\rm int}+\mathsf{N}_{\rm bry}}$. However, as in [22], we select a second set of distinct initial boundary centres $\{\mathbf{x}_{\mathsf{n}}^0\}_{\mathsf{n}=\mathsf{N}_{\rm int}+1}^{\mathsf{N}_{\rm int}+\mathsf{N}_{\rm bry}}$. This set is taken on a magnification of $\partial\Omega$ about the centre of Ω. We therefore now we define $\mathsf{N} = \mathsf{N}_{\rm int} + 2\mathsf{N}_{\rm bry}$, i.e. N is different than in the second order case.

The parameters now consist of the coefficients $\{a_{\mathsf{n}}\}_{\mathsf{n}=1}^{\mathsf{N}}$, the shape parameters $\{c_{\mathsf{n}}\}_{\mathsf{n}=1}^{\mathsf{N}}$ and the centre coordinates $\mathbf{x} = [\mathrm{x}_1, \mathrm{x}_2, \ldots, \mathrm{x}_{\mathsf{N}}]^T$, $\mathbf{y} = [\mathrm{y}_1, \mathrm{y}_2, \ldots, \mathrm{y}_{\mathsf{N}}]^T$ in 2D and $\mathbf{x} = [\mathrm{x}_1, \mathrm{x}_2, \ldots, \mathrm{x}_{\mathsf{N}}]^T$, $\mathbf{y} = [\mathrm{y}_1, \mathrm{y}_2, \ldots, \mathrm{y}_{\mathsf{N}}]^T$, $\mathbf{z} = [\mathrm{z}_1, \mathrm{z}_2, \ldots, \mathrm{z}_{\mathsf{N}}]^T$ in 3D. We shall use the same notation as in (3.4) with $\mathbf{X}$ denoting the centre coordinates. The parameters are found by solving the collocation equations at the sampling points

$$\mathcal{L}u_{\mathsf{N}}(\boldsymbol{x}_{\mathsf{m}}) \;=\; f(\boldsymbol{x}_{\mathsf{m}}), \quad \mathsf{m} = 1, \ldots, \mathsf{M}_{\rm int}, \tag{3.11a}$$

$$\mathcal{B}_1 u_{\mathsf{N}}(\boldsymbol{x}_{\mathsf{m}}) \;=\; g_1(\boldsymbol{x}_{\mathsf{m}}), \quad \mathsf{m} = \mathsf{M}_{\rm int}+1, \ldots, \mathsf{M}_{\rm int}+\mathsf{M}_{\rm bry}, \tag{3.11b}$$

$$\mathcal{B}_2 u_{\mathsf{N}}(\boldsymbol{x}_{\mathsf{m}}) \;=\; g_2(\boldsymbol{x}_{\mathsf{m}}), \quad \mathsf{m} = \mathsf{M}_{\rm int}+1, \ldots, \mathsf{M}_{\rm int}+\mathsf{M}_{\rm bry}. \tag{3.11c}$$

To have sufficiently many equations we take $\mathsf{M}_{\rm int} + 2\mathsf{M}_{\rm bry} \ge 4\,(\mathsf{N}_{\rm int} + 2\,\mathsf{N}_{\rm bry})$ in 2D and $\mathsf{M}_{\rm int} + 2\mathsf{M}_{\rm bry} \ge 5\,(\mathsf{N}_{\rm int} + 2\,\mathsf{N}_{\rm bry})$ in 3D.

The $\mathsf{M}_{\rm int} + 2\mathsf{M}_{\rm bry}$ equations (3.11a), (3.11b)–(3.11c) give rise the nonlinear system (2.1), namely,

$$\mathbf{F}(\boldsymbol{a}, \boldsymbol{c}, \mathbf{X}) := \begin{bmatrix} \mathtt{F}_1 \\ \mathtt{F}_2 \\ \vdots \\ \mathtt{F}_{\mathsf{M}_{\rm int}+2\mathsf{M}_{\rm bry}} \end{bmatrix}$$

$$= \begin{bmatrix} \mathcal{L}u_{\mathsf{N}}(\boldsymbol{x}_1) - f(\boldsymbol{x}_1) \\ \vdots \\ \mathcal{L}u_{\mathsf{N}}(\boldsymbol{x}_{\mathsf{M}_{\rm int}}) - f(\boldsymbol{x}_{\mathsf{M}_{\rm int}}) \\ \sqrt{\lambda_1}\ (\mathcal{B}_1 u_{\mathsf{N}}(\boldsymbol{x}_{\mathsf{M}_{\rm int}+1}) - g_1(\boldsymbol{x}_{\mathsf{M}_{\rm int}+1})) \\ \vdots \\ \sqrt{\lambda_1}\ \left(\mathcal{B}_1 u_{\mathsf{N}}(\boldsymbol{x}_{\mathsf{M}_{\rm int}+\mathsf{M}_{\rm bry}}) - g_1(\boldsymbol{x}_{\mathsf{M}_{\rm int}+\mathsf{M}_{\rm bry}})\right) \\ \sqrt{\lambda_2}\ (\mathcal{B}_2 u_{\mathsf{N}}(\boldsymbol{x}_{\mathsf{M}_{\rm int}+1}) - g_2(\boldsymbol{x}_{\mathsf{M}_{\rm int}+1})) \\ \vdots \\ \sqrt{\lambda_2}\ \left(\mathcal{B}_2 u_{\mathsf{N}}(\boldsymbol{x}_{\mathsf{M}_{\rm int}+\mathsf{M}_{\rm bry}}) - g_2(\boldsymbol{x}_{\mathsf{M}_{\rm int}+\mathsf{M}_{\rm bry}})\right) \end{bmatrix} = \mathbf{0}. \tag{3.12}$$

In (3.12) λ_1, λ_2 are fitted penalty factors accounting for the weight of the boundary conditions [12]. As in the second order case, the network will be trained by using `lsqnonlin` to minimize the functional

$$\begin{aligned} \mathtt{S}(\boldsymbol{a}, \boldsymbol{c}, \mathbf{X} \ := \ & \sum_{\mathsf{m}=1}^{\mathsf{M}_{\rm int}} \left[\mathcal{L}u_{\mathsf{N}}(\boldsymbol{x}_{\mathsf{m}}) - f(\boldsymbol{x}_{\mathsf{m}})\right]^2 + \lambda_1 \sum_{\mathsf{m}=\mathsf{M}_{\rm int}+1}^{\mathsf{M}_{\rm int}+\mathsf{M}_{\rm bry}} \left[\mathcal{B}_1 u_{\mathsf{N}}(\boldsymbol{x}_{\mathsf{m}}) - g_1(\boldsymbol{x}_{\mathsf{m}})\right]^2 \\ & + \lambda_2 \sum_{\mathsf{m}=\mathsf{M}_{\rm int}+1}^{\mathsf{M}_{\rm int}+\mathsf{M}_{\rm bry}} \left[\mathcal{B}_2 u_{\mathsf{N}}(\boldsymbol{x}_{\mathsf{m}}) - g_2(\boldsymbol{x}_{\mathsf{m}})\right]^2. \end{aligned} \tag{3.13}$$

Let $\mathcal{M} = \mathsf{M}_{\rm int+M_{bry}}$, $\mathsf{N} = \mathsf{N}_{\rm int} + 2\mathsf{N}_{\rm bry}$ and also $\mathsf{M} = \mathsf{M}_{\rm int} + 2\mathsf{M}_{\rm bry}$, the Jacobian in 2D is defined by

$$(\mathtt{J}_{\mathsf{m},\mathsf{n}})_{\mathsf{m}=1,\mathsf{n}=1}^{\mathsf{M}_{\rm int},\mathsf{N}} = \frac{\partial \mathcal{L}u_{\mathsf{N}}(\boldsymbol{x}_{\mathsf{m}})}{\partial a_{\mathsf{n}}}, \qquad (\mathtt{J}_{\mathsf{m},\mathsf{N}+\mathsf{n}})_{\mathsf{m}=1,\mathsf{n}=1}^{\mathsf{M}_{\rm int},\mathsf{N}} = \frac{\partial \mathcal{L}u_{\mathsf{N}}(\boldsymbol{x}_{\mathsf{m}})}{\partial c_{\mathsf{n}}},$$

$$(\mathtt{J}_{\mathsf{m},2N+\mathsf{n}})_{\mathsf{m}=1,\mathsf{n}=1}^{\mathsf{M}_{\rm int},\mathsf{N}} = \frac{\partial \mathcal{L}u_{\mathsf{N}}(\boldsymbol{x}_{\mathsf{m}})}{\partial \mathrm{x}_{\mathsf{n}}}, \qquad (\mathtt{J}_{\mathsf{m},3N+\mathsf{n}})_{\mathsf{m}=1,\mathsf{n}=1}^{\mathsf{M}_{\rm int},\mathsf{N}} = \frac{\partial \mathcal{L}u_{\mathsf{N}}(\boldsymbol{x}_{\mathsf{m}})}{\partial \mathrm{y}_{\mathsf{n}}},$$

$$(\mathtt{J}_{\mathsf{m},\mathsf{n}})_{\mathsf{m}=\mathsf{M}_{\rm int}+1,\mathsf{n}=1}^{\mathcal{M},\mathsf{N}} = \sqrt{\lambda_1}\,\frac{\partial \mathcal{B}_1 u_{\mathsf{N}}(\boldsymbol{x}_{\mathsf{m}})}{\partial a_{\mathsf{n}}}, \qquad (\mathtt{J}_{\mathsf{m},\mathsf{N}+\mathsf{n}})_{\mathsf{m}=\mathsf{M}_{\rm int}+1,\mathsf{n}=1}^{\mathcal{M},\mathsf{N}} = \sqrt{\lambda_1}\,\frac{\partial \mathcal{B}_1 u_{\mathsf{N}}(\boldsymbol{x}_{\mathsf{m}})}{\partial c_{\mathsf{n}}},$$

$$(\mathsf{J}_{\mathsf{m},2\mathsf{N}+\mathsf{n}})_{\mathsf{m}=\mathsf{M}_{\rm int}+1,\mathsf{n}=1}^{\mathcal{M},\mathsf{N}} = \sqrt{\lambda_1}\,\frac{\partial \mathcal{B}_1 u_{\mathsf{N}}(\boldsymbol{x}_{\mathsf{m}})}{\partial \mathrm{x}_{\mathsf{n}}}, \qquad (\mathsf{J}_{\mathsf{m},3\mathsf{N}+\mathsf{n}})_{\mathsf{m}=\mathsf{M}_{\rm int}+1,\mathsf{n}=1}^{\mathcal{M},\mathsf{N}} = \sqrt{\lambda_1}\,\frac{\partial \mathcal{B}_1 u_{\mathsf{N}}(\boldsymbol{x}_{\mathsf{m}})}{\partial \mathrm{y}_{\mathsf{n}}},$$

$$(\mathsf{J}_{\mathsf{m},\mathsf{n}})_{\mathsf{m}=\mathcal{M}+1,\mathsf{n}=1}^{\mathsf{M},\mathsf{N}} = \sqrt{\lambda_2}\,\frac{\partial \mathcal{B}_2 u_{\mathsf{N}}(\boldsymbol{x}_{\mathsf{m}})}{\partial a_{\mathsf{n}}}, \qquad (\mathsf{J}_{\mathsf{m},\mathsf{N}+\mathsf{n}})_{\mathsf{m}=\mathcal{M}+1,\mathsf{n}=1}^{\mathsf{M},\mathsf{N}} = \sqrt{\lambda_2}\,\frac{\partial \mathcal{B}_2 u_{\mathsf{N}}(\boldsymbol{x}_{\mathsf{m}})}{\partial c_{\mathsf{n}}},$$

$$(\mathsf{J}_{\mathsf{m},2\mathsf{N}+\mathsf{n}})_{\mathsf{m}=\mathcal{M}+1,\mathsf{n}=1}^{\mathsf{M},\mathsf{N}} = \sqrt{\lambda_2}\,\frac{\partial \mathcal{B}_2 u_{\mathsf{N}}(\boldsymbol{x}_{\mathsf{m}})}{\partial \mathrm{x}_{\mathsf{n}}}, \qquad (\mathsf{J}_{\mathsf{m},3\mathsf{N}+\mathsf{n}})_{\mathsf{m}=\mathcal{M}+1,\mathsf{n}=1}^{\mathsf{M},\mathsf{N}} = \sqrt{\lambda_2}\,\frac{\partial \mathcal{B}_2 u_{\mathsf{N}}(\boldsymbol{x}_{\mathsf{m}})}{\partial \mathrm{y}_{\mathsf{n}}}.$$

In 3D we need, in addition to the above,

$$(\mathsf{J}_{\mathsf{m},4N+\mathsf{n}})_{\mathsf{m}=1,\mathsf{n}=1}^{\mathsf{M}_{\rm int},\mathsf{N}} = \frac{\partial \mathcal{L} u_{\mathsf{N}}(\boldsymbol{x}_{\mathsf{m}})}{\partial \mathrm{z}_{\mathsf{n}}}, \quad (\mathsf{J}_{\mathsf{m},4\mathsf{N}+\mathsf{n}})_{\mathsf{m}=\mathcal{M}+1,\mathsf{n}=1}^{\mathsf{M},\mathsf{N}} = \sqrt{\lambda_2}\,\frac{\partial \mathcal{B}_2 u_{\mathsf{N}}(\boldsymbol{x}_{\mathsf{m}})}{\partial \mathrm{z}_{\mathsf{n}}}.$$

3.2.3. *Initial centre locations.* As suggested in Section 3.1.3 (see (3.8)–(3.9)) taking the initial locations of the centres in a magnification of Ω yields improved results. In this case, we place the second set of boundary centres $\{\mathbf{x}_{\mathsf{n}}^0\}_{\mathsf{n}=\mathsf{N}_{\rm int}+\mathsf{N}_{\rm bry}+1}^{\mathsf{N}_{\rm int}+2\mathsf{N}_{\rm bry}}$ on a magnification ξ of the (already magnified) first set of boundary centres $\{\mathbf{x}_{\mathsf{n}}^0\}_{\mathsf{n}=\mathsf{N}_{\rm int}+1}^{\mathsf{N}_{\rm int}+\mathsf{N}_{\rm bry}}$. For star–shaped boundaries (cf. (3.8)) we take

$$\tilde{\mathbf{x}}_{\mathsf{N}_{\rm bry}+\mathsf{n}}^0 = \xi\,\tilde{\mathbf{x}}_{\mathsf{n}}^0,\ \mathsf{n} = \mathsf{N}_{\rm int}+1,\ldots,\mathsf{N}_{\rm int}+\mathsf{N}_{\rm bry}, \tag{3.14}$$

and for general domains with centre $\mathbf{x}_{\rm c}$, we have (cf. (3.9))

$$\tilde{\mathbf{x}}_{\mathsf{N}_{\rm bry}+\mathsf{n}}^0 = \xi\,\left(\tilde{\mathbf{x}}_{\mathsf{n}}^0 - \mathrm{x}_{\rm c}\right) + \mathbf{x}_{\rm c},\ \mathsf{n} = \mathsf{N}_{\rm int}+1,\ldots,\mathsf{N}_{\rm int}+\mathsf{N}_{\rm bry}. \tag{3.15}$$

With this choice we avoid singular square matrices, see [21].

4. Numerical examples

We calculated the approximation u_{N} at L test data points $\{\mathrm{x}_\ell\}_{\ell=1}^{L}$ in $\overline{\Omega}$ from which we computed the maximum relative error

$$E = \frac{||u - u_{\mathsf{N}}||_{\infty,\overline{\Omega}}}{||u||_{\infty,\overline{\Omega}}} \tag{4.1}$$

and the root mean square error $\mathcal{E}$

$$\mathcal{E} = \left(\frac{1}{L}\sum_{\ell=1}^{L}\left[u(\mathrm{x}_\ell) - u_{\mathsf{N}}(\mathrm{x}_\ell)\right]^2\right)^{1/2}. \tag{4.2}$$

We used the 2D and 3D Gaussian RBFs

$$\Phi(c_{\mathsf{n}}, r_{\mathsf{n}}) = \mathrm{e}^{-c_{\mathsf{n}}\, r_{\mathsf{n}}^2}, \tag{4.3}$$

the derivatives of which required in the implementation of method are given in the Appendix. The initial values of the shape parameters were uniformly distributed on an interval $[d_{\min}, d_{\max}]$ using the formula (see [29])

$$\boldsymbol{c}_0(\ell) = d_{\min} + (d_{\max} - d_{\min})\frac{(\ell-1)}{(\mathsf{N}-1)}, \quad \ell = 1,\ldots,\mathsf{N}. \tag{4.4}$$

The selection of the initial shape parameter values is crucial to the accuracy and efficiency of the method. We have conducted intensive tests and obtained good results for $2 < d_{\max} < 3$ (the range in which the initial shape parameter values are distributed) and $2 < \eta < 3$ (the magnification factor of the enlarged domain in which the centres are initially placed) in 2D cases. For 3D cases, due to the complexity of the solids, the initial location of the sources and η were selected differently to 2D cases. We will further explain this in Examples 3 and 4. In [1, 7, 12], random numbers were assigned for the initial values of the RBF weights. In this work, the initial values of the RBF weights were all taken to be equal to zero. We also experimented with assigning random initial weight values and found it made little difference to our results.

As mentioned in Section 2, we need to set the maximum number of iterations `MaxIter` and the tolerances `TolFun` and `TolX` to prevent the solution process to run indefinitely. In the numerical tests presented in this section, we set `MaxIter` = 500 for 2D cases and 50 for 3D cases and the tolerances `TolFun` = `TolX` = `Tol`. In all our numerical tests, we noticed that the pre–assigned value of the tolerance in `lsqnonlin` has little impact on the accuracy but could affect the efficiency (number of iterations) if not chosen properly. In general, we would suggest taking tolerance values between 1.0(-6) to 1.0(-12).

The numerical computations in this section were carried out using MATLAB© R2020b on x64-based processor, Intel(R) Core(TM) i7-3820 CPU @ 3.60GHz, 64 GB memory.

4.1. **Example 1.** Consider the 2D BVP

$$\Delta u = f(x, y) \quad \text{in} \quad \Omega, \tag{4.5a}$$

$$u = g(x, y) \quad \text{on} \quad \partial\Omega, \tag{4.5b}$$

where f and g are derived from the exact solution

$$u(x, y) = -\frac{1}{2\pi^2} \sin(\pi x) \sin(\pi y), \quad (x, y) \in \overline{\Omega}. \tag{4.6}$$

This BVP was considered in [1, 7, 12] when Ω is the unit square (and clearly $g = 0$). We thus first consider the unit square domain and then the five–star domain shown in Figure 3. The polar coordinates equation of the five–star boundary $\partial\Omega$ is

$$r(\vartheta) = 1 + \cos^2(5\vartheta/2), \quad 0 \le \vartheta \le 2\pi.$$

In both cases, the interior collocation points are taken to be a distribution of Halton points and the boundary collocation points are distributed uniformly. The initial coordinates of the centres are constructed in a similar way and we take a Halton distribution of L=101 and 300 interior test data points for the square and five–star domains, respectively.

4.1.1. *Implementation details.* The vector $\mathbf{F}$ (3.5) and Jacobian $\mathtt{J}$ (3.7) are

$$\mathsf{F_i} = \sum_{\mathsf{n}=1}^{\mathsf{N}} a_{\mathsf{n}}\, \Delta\Phi(c_{\mathsf{n}}, r_{\mathsf{i,n}}) - \Delta u(\boldsymbol{x}_{\mathsf{i}}), \quad \mathsf{i} = 1, \ldots, \mathsf{M}_{\text{int}},$$

$$\mathsf{F_i} = \sqrt{\lambda}\, \left(\sum_{\mathsf{n}=1}^{\mathsf{N}} a_{\mathsf{n}}\, \Phi(c_{\mathsf{n}}, r_{\mathsf{i,n}}) - u(\boldsymbol{x}_{\mathsf{i}}) \right), \quad \mathsf{i} = \mathsf{M}_{\text{int}} + 1, \ldots, \mathsf{M},$$

For $\mathsf{i} = 1, \ldots, \mathsf{M}_{\text{int}},\ \mathsf{j} = 1, \ldots, \mathsf{N}$,

$$\mathsf{J_{i,j}} = \Delta\Phi(c_{\mathsf{j}}, r_{\mathsf{i,j}}), \qquad \mathsf{J_{i,N+j}} = a_{\mathsf{j}} \frac{\partial \Delta\Phi(c_{\mathsf{j}}, r_{\mathsf{i,j}})}{\partial c_{\mathsf{j}}},$$

$$\mathsf{J_{i,2N+j}} = a_{\mathsf{j}} \frac{\partial \Delta\Phi(c_{\mathsf{j}}, r_{\mathsf{i,j}})}{\partial \mathrm{x}_{\mathsf{j}}}, \qquad \mathsf{J_{i,3N+j}} = a_{\mathsf{j}} \frac{\partial \Delta\Phi(c_{\mathsf{j}}, r_{\mathsf{i,j}})}{\partial \mathrm{y}_{\mathsf{j}}},$$

For $\mathsf{i} = \mathsf{M}_{\text{int}} + 1, \ldots, \mathsf{M},\ \mathsf{j} = 1, \ldots, \mathsf{N}$,

$$\mathsf{J_{i,j}} = \sqrt{\lambda}\, \Phi(c_{\mathsf{j}}, r_{\mathsf{i,j}}), \qquad \mathsf{J_{i,N+j}} = \sqrt{\lambda}\, a_{\mathsf{j}} \frac{\partial \Phi(c_{\mathsf{j}}, r_{\mathsf{i,j}})}{\partial c_{\mathsf{j}}},$$

$$\mathsf{J_{i,2N+j}} = \sqrt{\lambda}\, a_{\mathsf{j}} \frac{\partial \Phi(c_{\mathsf{j}}, r_{\mathsf{i,j}})}{\partial \mathrm{x}_{\mathsf{j}}}, \qquad \mathsf{J_{i,3N+j}} = \sqrt{\lambda}\, a_{\mathsf{j}} \frac{\partial \Phi(c_{\mathsf{j}}, r_{\mathsf{i,j}})}{\partial \mathrm{y}_{\mathsf{j}}}.$$

4.1.2. *Results.* We first present the results for the square domain. For training, we chose 275 collocation points of which 200 Halton points were placed in the interior and 76 uniformly distributed points on the boundary. In a similar way, we chose 61 centres of which 45 were placed in the interior and 16 on the boundary. We also took a magnification factor $\eta = 2.2$. The initial shape parameter values were evenly distributed in the interval $[d_{\min}, d_{\max}] = [0.5, 2.5]$. In Figure 2 we show the initial distribution of sources and collocation points. In Table 1 we present some results using various values of `Tol` and λ from which we observe that for larger values of λ, the accuracy improves slightly. Little difference in accuracy is observed for various `Tol` values, but smaller values can lead to a large number of iterations (and high CPU times).

Note that `lsqnonlin` offers the option of not providing the Jacobian. In Table 2, we present the results obtained with the same parameters as those in Table 1 without providing the Jacobian. Comparing the results with and without the Jacobian, we notice that the accuracy of these two approaches is similar except the CPU time is much higher when the Jacobian is not given. For 3D problems, the difference is even more pronounced. For the rest of this section, we will only present results obtained when providing the Jacobian. Comparing our results with those obtained in [12] for the square domain, we found that our approach is far more accurate. As no CPU times were reported in [12] we cannot compare the two approaches in terms of efficiency.

Next, we present the results for the five–star domain. In Figure 3 we present the initial locations of the sources and collocation points for $\mathsf{M}_{\text{int}} = 400$, $\mathsf{M}_{\text{bry}} = 160$, $\mathsf{N}_{\text{int}} = 50$, $\mathsf{N}_{\text{bry}} = 40$. Some results for various values of `Tol` and λ with $\eta = 2.8$, $[d_{\min}, d_{\max}] = [0.5, 2.9]$ are depicted in Table 3

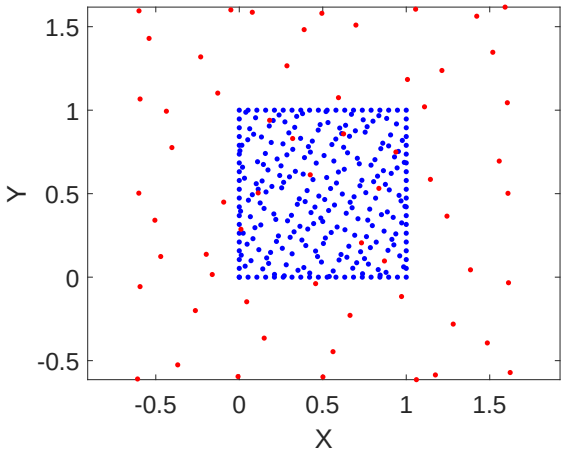


FIGURE 2. Example 1: Initial positions of sources (•) and collocation points (•).

	Tol	E	$\mathcal{E}$	niter	CPU (secs)
$\lambda = 10000$					
	1.0(-8)	5.971(-6)	1.463(-7)	6	0.06
	1.0(-10)	3.057(-6)	5.624(-8)	61	0.31
	1.0(-12)	2.123(-6)	4.286(-8)	500	2.36
$\lambda = 1000$					
	1.0(-8)	7.044(-6)	9.542(-8)	5	0.06
	1.0(-10)	7.197(-6)	9.669(-8)	6	0.06
	1.0(-12)	5.709(-6)	9.082(-8)	500	2.37
$\lambda = 100$					
	1.0(-8)	7.826(-5)	8.261(-7)	11	0.09
	1.0(-10)	3.149(-5)	4.415(-7)	71	0.34
	1.0(-12)	1.883(-5)	2.716(-7)	500	2.37

TABLE 1. Example 1: Results for the square domain with various Tol and λ values when providing the Jacobian.

from which we observe the solution process converges rapidly (in few iterations) with little change for various decreasing values of Tol. For larger values of λ, higher accuracy was obtained. This means that we get better results when putting more weight on the BC in the solution process.

Next, we examine how the accuracy is affected as we increase the number of collocation points and centres. We consider the star–shaped domain with $\lambda = 10000$ with the same initial parameters as before, i.e. with $\eta = 2.8$, $[d_{\min}, d_{\max}] = [0.5, 2.9]$. The obtained accuracy employing different numbers of training points and centres is depicted in Table 4. We observe that the accuracy improves rapidly, proportionally to the number of training points and centres until it reaches

	Tol	E	$\mathcal{E}$	niter	CPU (secs)
$\lambda = 10000$					
	1.0(-8)	6.553(-6)	1.962(-7)	9	1.21
	1.0(-10)	3.089(-6)	6.520(-8)	48	5.57
	1.0(-12)	2.291(-6)	4.423(-8)	500	63.99
$\lambda = 1000$					
	1.0(-8)	1.631(-5)	3.050(-7)	9	1.15
	1.0(-10)	7.147(-6)	1.190(-7)	34	3.98
	1.0(-12)	5.751(-6)	8.739(-8)	500	64.07
$\lambda = 100$					
	1.0(-8)	1.985(-5)	3.582(-7)	5	0.76
	1.0(-10)	2.036(-5)	3.557(-7)	12	1.51
	1.0(-12)	1.650(-5)	2.289(-7)	500	63.96

TABLE 2. Example 1: Results for the square domain with various Tol and λ values without providing the Jacobian.

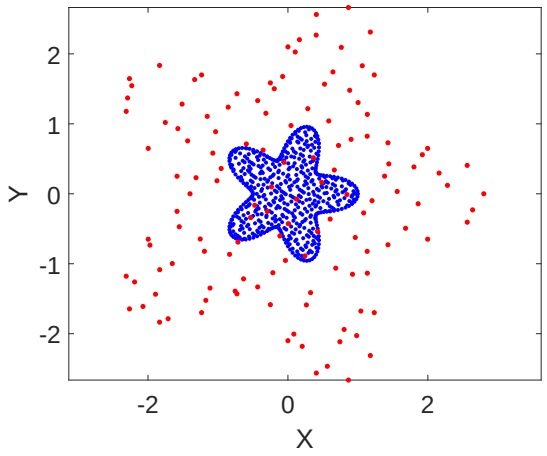


FIGURE 3. Example 1: Initial positions of sources (●) and collocation points (●).

a certain limit, which is consistent with the standard accuracy of RBFCMs indicated in the literature. Furthermore, in this table we also compare the results obtained with the proposed method and the traditional Kansa method [18]. Overall, we notice that, for this example, the proposed method performs better than the Kansa method in terms of accuracy. However, we would like to emphasize that the primary goal of the current approach is to render the solution procedure as simple as possible by letting the weighting coefficients, shape parameters, and centres be determined simultaneously.

	Tol	E	$\mathcal{E}$	niter	CPU (secs)
$\lambda = 10000$					
	1.0(-8)	6.249(-8)	8.854(-10)	4	0.15
	1.0(-10)	6.538(-8)	8.940(-10)	5	0.19
	1.0(-12)	6.538(-8)	8.940(-10)	5	0.18
$\lambda = 1000$					
	1.0(-8)	4.509(-7)	3.528(-9)	4	0.17
	1.0(-10)	3.892(-7)	3.408(-9)	5	0.16
	1.0(-12)	3.892(-7)	3.408(-9)	5	0.19
$\lambda = 100$					
	1.0(-8)	2.655(-6)	3.077(-8)	6	0.20
	1.0(-10)	2.655(-6)	3.077(-8)	6	0.19
	1.0(-12)	1.186(-6)	1.236(-8)	21	0.59

TABLE 3. Example 1: Results for various Tol and λ values for the five–star domain.

M_{int}	M_{bry}	N_{int}	N_{bry}	current method		Kansa method	
				E	$\mathcal{E}$	E	$\mathcal{E}$
200	120	40	40	5.512(-5)	9.011(-7)	6.185(-5)	4.940(-7)
300	160	50	40	1.322(-5)	3.017(-7)	8.795(-6)	6.113(-8)
400	160	90	40	6.244(-8)	9.175(-10)	3.291(-6)	2.771(-8)
500	250	80	50	5.334(-8)	1.120(-9)	3.946(-6)	4.484(-8)
600	250	90	40	5.765(-8)	8.892(-10)	2.102(-6)	2.923(-8)

TABLE 4. Example 1: Results for various numbers of training points and centres for the current method and the Kansa method.

4.2. **Example 2.** We next consider the 2D fourth order PDE

$$\mathcal{L}u = \Delta^2 u = f(x,y) = -2\pi^2 \sin(\pi x)\sin(\pi y) \quad \text{in} \quad \Omega, \tag{4.7}$$

subject to BCs for u and its normal derivative $\partial u/\partial n$, calculated from the exact solution (4.6). The domain Ω is the L–shaped domain shown in Figure 4. The coordinates of the six vertices of the L-shaped domain are $\{(-0.2,-0.2),(0.8,-0.2),(0.8,0.3),(0.3,0.3),(0.3,0.8),(-0.2,0.8)\}$. The interior collocation points are taken to be a distribution of Halton points and the boundary collocation points are distributed uniformly. The boundary and interior centres are distributed in a similar fashion and a set of $L = 300$ interior Halton points composes the set of test points.

4.2.1. *Implementation details.* The vector $\mathbf{F}$ (3.12) and corresponding Jacobian $\mathtt{J}$ are

$$\mathtt{F_i} = \sum_{\mathtt{n}=1}^{\mathtt{N}} a_{\mathtt{n}}\, \Delta^2\Phi(c_{\mathtt{n}}, r_{\mathtt{i,n}}) - \Delta^2 u(\boldsymbol{x}_{\mathtt{i}}), \quad \mathtt{i} = 1, \ldots, \mathtt{M}_{\mathrm{int}},$$

$$\mathtt{F_i} = \sqrt{\lambda_1}\left(\sum_{\mathtt{n}=1}^{\mathtt{N}} a_{\mathtt{n}}\, \Phi(c_{\mathtt{n}}, r_{\mathtt{i,n}}) - u(\boldsymbol{x}_{\mathtt{i}})\right), \quad \mathtt{i} = \mathtt{M}_{\mathrm{int}} + 1, \ldots, \mathtt{M},$$

$$\mathtt{F_i} = \sqrt{\lambda_2}\left(\sum_{\mathtt{n}=1}^{\mathtt{N}} a_{\mathtt{n}}\, \frac{\partial\Phi}{\partial n}(c_{\mathtt{n}}, r_{\mathtt{i,n}}) - \frac{\partial u}{\partial n}(\boldsymbol{x}_{\mathtt{i}})\right), \quad \mathtt{i} = \mathtt{M}+1, \ldots, \mathtt{M}+\mathtt{M}_{\mathrm{bry}}.$$

For $\mathtt{i} = 1, \ldots, \mathtt{M}_{\mathrm{int}},\ \mathtt{j} = 1, \ldots, \mathtt{N}$,

$$\mathtt{J_{i,j}} = \Delta^2\Phi(c_{\mathtt{j}}, r_{\mathtt{i,j}}), \qquad \mathtt{J_{i,N+j}} = a_{\mathtt{j}}\frac{\partial\Delta^2\Phi(c_{\mathtt{j}}, r_{\mathtt{i,j}})}{\partial c_{\mathtt{j}}},$$

$$\mathtt{J_{i,2N+j}} = a_{\mathtt{j}}\frac{\partial\Delta^2\Phi(c_{\mathtt{j}}, r_{\mathtt{i,j}})}{\partial \mathrm{x}_{\mathtt{j}}}, \qquad \mathtt{J_{i,3N+j}} = a_{\mathtt{j}}\frac{\partial\Delta^2\Phi(c_{\mathtt{j}}, r_{\mathtt{i,j}})}{\partial \mathrm{y}_{\mathtt{j}}}.$$

For $\mathtt{i} = \mathtt{M}_{\mathrm{int}} + 1, \ldots, \mathtt{M},\ \mathtt{j} = 1, \ldots, \mathtt{N}$,

$$\mathtt{J_{i,j}} = \sqrt{\lambda_1}\,\Phi(c_{\mathtt{j}}, r_{\mathtt{i,j}}), \qquad \mathtt{J_{i,N+j}} = \sqrt{\lambda_1}\, a_{\mathtt{j}}\frac{\partial\Phi(c_{\mathtt{j}}, r_{\mathtt{i,j}})}{\partial c_{\mathtt{j}}},$$

$$\mathtt{J_{i,2N+j}} = \sqrt{\lambda_1}\, a_{\mathtt{j}}\frac{\partial\Phi(c_{\mathtt{j}}, r_{\mathtt{i,j}})}{\partial \mathrm{x}_{\mathtt{j}}}, \qquad \mathtt{J_{i,3N+j}} = \sqrt{\lambda_1}\, a_{\mathtt{j}}\frac{\partial\Phi(c_{\mathtt{j}}, r_{\mathtt{i,j}})}{\partial \mathrm{y}_{\mathtt{j}}}.$$

For $\mathtt{i} = \mathtt{M}+1, \ldots, \mathtt{M}+\mathtt{M}_{\mathrm{bry}},\ \mathtt{j} = 1, \ldots, \mathtt{N}$,

$$\mathtt{J_{i,j}} = \sqrt{\lambda_2}\,\frac{\partial\Phi}{\partial n}(c_{\mathtt{j}}, r_{\mathtt{i,j}}) = \sqrt{\lambda_2}\,\left(n_x\frac{\partial\Phi}{\partial x} + n_y\frac{\partial\Phi}{\partial y}\right)(c_{\mathtt{j}}, r_{\mathtt{i,j}}),$$

$$\mathtt{J_{i,N+j}} = \sqrt{\lambda_2}\, a_{\mathtt{j}}\frac{\partial}{\partial c_{\mathtt{j}}}\left(\frac{\partial\Phi}{\partial n}(c_{\mathtt{j}}, r_{\mathtt{i,j}})\right) = \sqrt{\lambda_2}\, a_{\mathtt{j}}\frac{\partial}{\partial c_{\mathtt{j}}}\left(\left(n_x\frac{\partial\Phi}{\partial x} + n_y\frac{\partial\Phi}{\partial y}\right)(c_{\mathtt{j}}, r_{\mathtt{i,j}})\right),$$

$$\mathtt{J_{i,2N+j}} = \sqrt{\lambda_2}\, a_{\mathtt{j}}\frac{\partial}{\partial \mathrm{x}_{\mathtt{j}}}\left(\frac{\partial\Phi}{\partial n}(c_{\mathtt{j}}, r_{\mathtt{i,j}})\right) = \sqrt{\lambda_2}\, a_{\mathtt{j}}\,\frac{\partial}{\partial \mathrm{x}_{\mathtt{j}}}\left(\left(n_x\frac{\partial\Phi}{\partial x} + n_y\frac{\partial\Phi}{\partial y}\right)(c_{\mathtt{j}}, r_{\mathtt{i,j}})\right),$$

$$\mathtt{J_{i,3N+j}} = \sqrt{\lambda_2}\, a_{\mathtt{j}}\frac{\partial}{\partial \mathrm{y}_{\mathtt{j}}}\left(\frac{\partial\Phi}{\partial n}(c_{\mathtt{j}}, r_{\mathtt{i,j}})\right) = \sqrt{\lambda_2}\, a_{\mathtt{j}}\,\frac{\partial}{\partial \mathrm{y}_{\mathtt{j}}}\left(\left(n_x\frac{\partial\Phi}{\partial x} + n_y\frac{\partial\Phi}{\partial y}\right)(c_{\mathtt{j}}, r_{\mathtt{i,j}})\right).$$

4.2.2. *Results.* Some results for various values of `Tol` and $\lambda_1 = \lambda_2$ are depicted in Table 5. In Figure 4 we present the initial locations of the sources and collocation points for $\lambda_1 = \lambda_2$ using $\mathtt{M}_{\mathrm{int}} = 300$, $\mathtt{M}_{\mathrm{bry}} = 240$, $\mathtt{N}_{\mathrm{int}} = 50$, $\mathtt{N}_{\mathrm{bry}} = 40$, $\eta = 2.2$, $\xi = 1.2$, $d_{\min} = 0.1$, $d_{\max} = 2.8$. As in Example 1 the accuracy of the method is not sensitive to the decreasing values of `Tol`.

4.3. **Example 3.** We examine the 3D second order BVP consisting of the PDE

$$\mathcal{L}u = \Delta u = \sin(\pi x)\sin(\pi y)\sin(\pi z) \quad \text{in } \Omega, \tag{4.8}$$

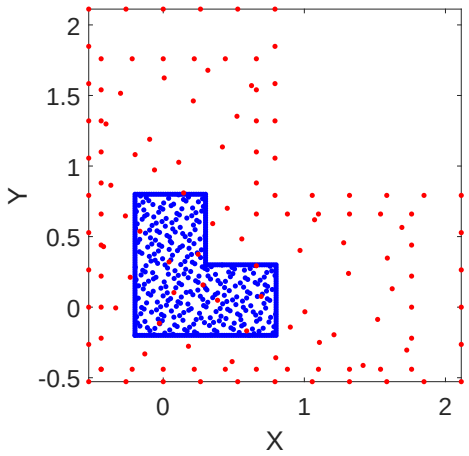


FIGURE 4. Example 2: Initial positions of sources (●) and collocation points (●).

	Tol	E	$\mathcal{E}$	niter	CPU (secs)
$\lambda_1 = \lambda_2 = 10000$					
	1.0(-8)	5.691(-6)	6.624(-8)	38	2.05
	1.0(-10)	1.740(-6)	2.055(-8)	114	7.87
	1.0(-12)	8.954(-7)	1.095(-8)	385	36.05
$\lambda_1 = \lambda_2 = 1000$					
	1.0(-8)	7.776(-4)	1.331(-5)	11	0.52
	1.0(-10)	7.756(-4)	1.325(-5)	13	0.57
	1.0(-12)	7.622(-4)	1.300(-5)	500	20.28
$\lambda_1 = \lambda_2 = 100$					
	1.0(-8)	1.940(-5)	3.602(-7)	18	1.00
	1.0(-10)	7.979(-6)	1.218(-7)	55	2.54
	1.0(-12)	4.121(-6)	4.920(-8)	145	7.02

TABLE 5. Example 2: Results for various values of Tol for the L-shaped domain.

and a Dirichlet BC calculated from the exact solution

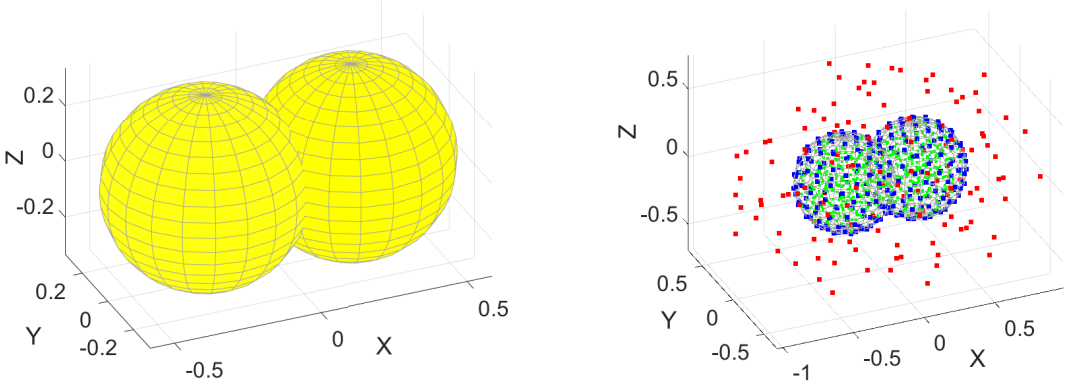
$$u(x, y, z) = -\frac{1}{3\pi^2} \sin(\pi x) \sin(\pi y) \sin(\pi z). \tag{4.9}$$

The computational domain Ω is the double–sphere shown in Figure 5(a). A typical distribution of collocation points and initial centre points (red dots) is shown in Figure 5(b). The parametric equation of the double–sphere is given as follows:

$$\Omega = \{(x, y, z) \in \mathbb{R}^3 : H(x, y, z) \leq 1/3\}$$

where

$$H(x,y,z) = \min\left\{\left(x-\frac{1}{4}\right)^2, \left(x+\frac{1}{4}\right)^2\right\} + y^2 + z^2.$$



(a) (b)

FIGURE 5. Example 3: The profiles of (a) the double-sphere and (b) initial positions of sources (•), boundary collocation points (•), and interior collocation points (•).

The boundary and interior collocation points are distributed approximately uniformly. In this example, centres were chosen differently than in the 2D cases due to the complexity of the 3D domain. More specifically, they were all located in an ellipsoid with a three axes ratio 1:0.7:0.7 centred at the origin and the magnification factor was taken as $\eta = 1.3$. In the numerical implementation, we chose $\mathsf{M}_{\rm int} = 900$, $\mathsf{M}_{\rm bry} = 600$, $\mathsf{N}_{\rm int} + \mathsf{N}_{\rm bry} = 220$, $d_{\min} = 0.5$, $d_{\max} = 2.8$. We also took a set of $L = 201$ interior Halton test points.

4.3.1. *Implementation details.* The vector $\mathbf{F}$ (3.5) and Jacobian $\mathtt{J}$ (3.7) are

$$\mathsf{F}_{\mathsf{i}} = \sum_{\mathsf{n}=1}^{\mathsf{N}} a_{\mathsf{n}}\, \Delta\Phi(c_{\mathsf{n}}, r_{\mathsf{i},\mathsf{n}}) - \Delta u(\boldsymbol{x}_{\mathsf{i}}), \quad \mathsf{i} = 1, \ldots, \mathsf{M}_{\rm int},$$

$$\mathsf{F}_{\mathsf{i}} = \sqrt{\lambda}\,\Big(\sum_{\mathsf{n}=1}^{\mathsf{N}} a_{\mathsf{n}}\, \Phi(c_{\mathsf{n}}, r_{\mathsf{i},\mathsf{n}}) - u(\boldsymbol{x}_{\mathsf{i}})\Big), \quad \mathsf{i} = \mathsf{M}_{\rm int}+1, \ldots, \mathsf{M}.$$

For $\mathsf{i} = 1, \ldots, \mathsf{M}_{\rm int}$, $\mathsf{j} = 1, \ldots, \mathsf{N}$,

$$\mathtt{J}_{\mathsf{i},\mathsf{j}} = \Delta\Phi(c_{\mathsf{j}}, r_{\mathsf{i},\mathsf{j}}), \quad \mathtt{J}_{\mathsf{i},\mathsf{N}+\mathsf{j}} = a_{\mathsf{j}} \frac{\partial \Delta\Phi(c_{\mathsf{j}}, r_{\mathsf{i},\mathsf{j}})}{\partial c_{\mathsf{j}}}, \quad \mathtt{J}_{\mathsf{i},2\mathsf{N}+\mathsf{j}} = a_{\mathsf{j}} \frac{\partial \Delta\Phi(c_{\mathsf{j}}, r_{\mathsf{i},\mathsf{j}})}{\partial \mathrm{x}_{\mathsf{j}}},$$

$$\mathtt{J}_{\mathsf{i},3\mathsf{N}+\mathsf{j}} = a_{\mathsf{j}} \frac{\partial \Delta\Phi(c_{\mathsf{j}}, r_{\mathsf{i},\mathsf{j}})}{\partial \mathrm{y}_{\mathsf{j}}}, \quad \mathtt{J}_{\mathsf{i},4\mathsf{N}+\mathsf{j}} = a_{\mathsf{j}} \frac{\partial \Delta\Phi(c_{\mathsf{j}}, r_{\mathsf{i},\mathsf{j}})}{\partial \mathrm{z}_{\mathsf{j}}}.$$

For $\mathsf{i} = \mathsf{M}_{\rm int}+1,\ldots,\mathsf{M},\ \mathsf{j}=1,\ldots,\mathsf{N}$,

$$\mathtt{J}_{\mathsf{i},\mathsf{j}} = \sqrt{\lambda}\,\Phi(c_{\mathsf{j}}, r_{\mathsf{i},\mathsf{j}}), \quad \mathtt{J}_{\mathsf{i},\mathsf{N}+\mathsf{j}} = \sqrt{\lambda}\,a_{\mathsf{j}}\frac{\partial\Phi(c_{\mathsf{j}}, r_{\mathsf{i},\mathsf{j}})}{\partial c_{\mathsf{j}}}, \quad \mathtt{J}_{\mathsf{i},2\mathsf{N}+\mathsf{j}} = \sqrt{\lambda}\,a_{\mathsf{j}}\frac{\partial\Phi(c_{\mathsf{j}}, r_{\mathsf{i},\mathsf{j}})}{\partial \mathrm{x}_{\mathsf{j}}},$$
$$\mathtt{J}_{\mathsf{i},3\mathsf{N}+\mathsf{j}} = \sqrt{\lambda}\,a_{\mathsf{j}}\frac{\partial\Phi(c_{\mathsf{j}}, r_{\mathsf{i},\mathsf{j}})}{\partial \mathrm{y}_{\mathsf{j}}}, \quad \mathtt{J}_{\mathsf{i},4\mathsf{N}+\mathsf{j}} = \sqrt{\lambda}\,a_{\mathsf{j}}\frac{\partial\Phi(c_{\mathsf{j}}, r_{\mathsf{i},\mathsf{j}})}{\partial \mathrm{z}_{\mathsf{j}}}.$$

4.3.2. *Results.* Some results for various values of `Tol` and λ with $\eta = 1.3$ are depicted in Table 6. The accuracy is unaffected by reducing the value of `Tol` and increases with increasing λ.

	`Tol`	E	$\mathcal{E}$	`niter`	CPU (secs)
$\lambda = 10000$					
	1.0(-6)	7.482(-5)	2.178(-7)	12	2.14
	1.0(-8)	7.482(-5)	2.178(-7)	12	2.14
$\lambda = 1000$					
	1.0(-6)	1.028(-3)	4.377(-6)	12	2.12
	1.0(-8)	1.028(-3)	4.377(-6)	12	2.14
$\lambda = 100$					
	1.0(-6)	2.687(-4)	7.498(-7)	12	2.09
	1.0(-8)	2.687(-4)	7.498(-7)	12	2.11

TABLE 6. Example 3: Results for various values of `Tol` and λ for the double-sphere domain.

4.4. **Example 4.** We next consider the 3D fourth order PDE

$$\mathcal{L}u = \Delta^2 u = -3\pi^2 \sin(\pi x)\sin(\pi y)\sin(\pi z) \quad \text{in} \quad \Omega, \tag{4.10}$$

subject to BCs for u and its normal derivative $\partial u/\partial n$, calculated from the exact solution (4.9). The domain Ω is the Stanford Bunny shown in Figure 6(a). The boundary collocation points of the Bunny are obtained from [33]. A typical distribution of the boundary points is shown in Figure 6(b). The interior collocation points are distributed uniformly inside the domain We took all the centres in a unit sphere centred at the geometric centre of the Bunny which is (-0.0836, 0.5456, 0) and the magnification factor η to be 1.4.

4.4.1. *Implementation details.* The vector $\mathbf{F}$ (3.12) and Jacobian $\mathtt{J}$ are

$$\mathtt{F}_{\mathsf{i}} = \sum_{\mathsf{n}=1}^{\mathsf{N}} a_{\mathsf{n}}\,\Delta^2\Phi(c_{\mathsf{n}}, r_{\mathsf{i},\mathsf{n}}) - \Delta^2 u(\boldsymbol{x}_{\mathsf{i}}), \quad \mathsf{i} = 1,\ldots,\mathsf{M}_{\rm int},$$

$$\mathtt{F}_{\mathsf{i}} = \sqrt{\lambda_1}\,\Big(\sum_{\mathsf{n}=1}^{\mathsf{N}} a_{\mathsf{n}}\,\Phi(c_{\mathsf{n}}, r_{\mathsf{i},\mathsf{n}}) - u(\boldsymbol{x}_{\mathsf{i}})\Big), \quad \mathsf{i} = \mathsf{M}_{\rm int}+1,\ldots,\mathsf{M},$$

$$\mathtt{F}_{\mathsf{i}} = \sqrt{\lambda_2}\,\Big(\sum_{\mathsf{n}=1}^{\mathsf{N}} a_{\mathsf{n}}\,\frac{\partial\Phi}{\partial n}(c_{\mathsf{n}}, r_{\mathsf{i},\mathsf{n}}) - \frac{\partial u}{\partial n}(\boldsymbol{x}_{\mathsf{i}})\Big), \quad \mathsf{i} = \mathsf{M}+1,\ldots,\mathsf{M}+\mathsf{M}_{\rm bry}.$$

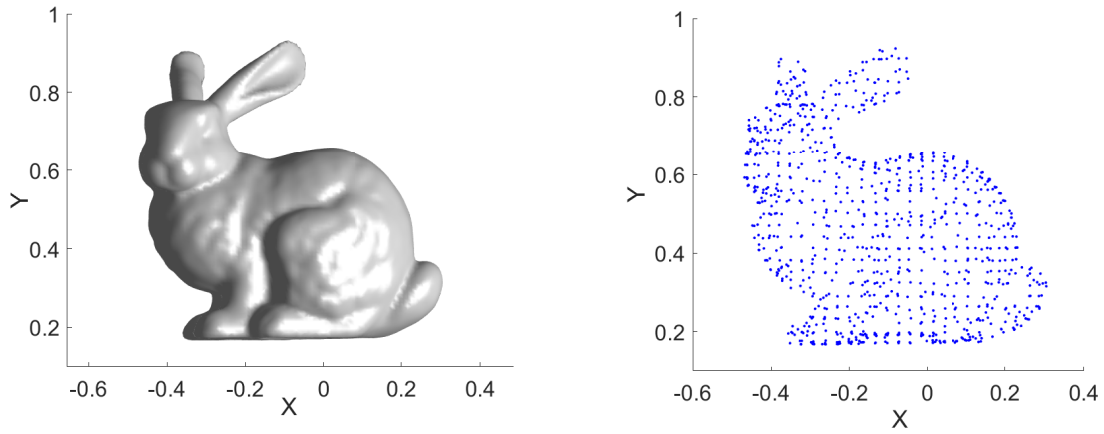


FIGURE 6. Example 4: The profile of (a) Bunny domain (b) typical boundary points.

For $\mathsf{i} = 1, \ldots, \mathsf{M}_{\rm int},\ \mathsf{j} = 1, \ldots, \mathsf{N}$,

$$\mathsf{J}_{\mathsf{i,j}} = \Delta^2\Phi(c_{\mathsf{j}}, r_{\mathsf{i,j}}), \quad \mathsf{J}_{\mathsf{i,N+j}} = a_{\mathsf{j}}\frac{\partial\Delta^2\Phi(c_{\mathsf{j}}, r_{\mathsf{i,j}})}{\partial c_{\mathsf{j}}}, \quad \mathsf{J}_{\mathsf{i,2N+j}} = a_{\mathsf{j}}\frac{\partial\Delta^2\Phi(c_{\mathsf{j}}, r_{\mathsf{i,j}})}{\partial \mathrm{x}_{\mathsf{j}}},$$
$$\mathsf{J}_{\mathsf{i,3N+j}} = a_{\mathsf{j}}\frac{\partial\Delta^2\Phi(c_{\mathsf{j}}, r_{\mathsf{i,j}})}{\partial \mathrm{y}_{\mathsf{j}}}, \quad \mathsf{J}_{\mathsf{i,4N+j}} = a_{\mathsf{j}}\frac{\partial\Delta^2\Phi(c_{\mathsf{j}}, r_{\mathsf{i,j}})}{\partial \mathrm{z}_{\mathsf{j}}}.$$

For $\mathsf{i} = \mathsf{M}_{\rm int}+1, \ldots, \mathsf{M},\ \mathsf{j} = 1, \ldots, \mathsf{N}$,

$$\mathsf{J}_{\mathsf{i,j}} = \sqrt{\lambda_1}\,\Phi(c_{\mathsf{j}}, r_{\mathsf{i,j}}), \quad \mathsf{J}_{\mathsf{i,N+j}} = \sqrt{\lambda_1}\,a_{\mathsf{j}}\frac{\partial\Phi(c_{\mathsf{j}}, r_{\mathsf{i,j}})}{\partial c_{\mathsf{j}}}, \quad \mathsf{J}_{\mathsf{i,2N+j}} = \sqrt{\lambda_1}\,a_{\mathsf{j}}\frac{\partial\Phi(c_{\mathsf{j}}, r_{\mathsf{i,j}})}{\partial \mathrm{x}_{\mathsf{j}}},$$
$$\mathsf{J}_{\mathsf{i,3N+j}} = \sqrt{\lambda_1}\,a_{\mathsf{j}}\frac{\partial\Phi(c_{\mathsf{j}}, r_{\mathsf{i,j}})}{\partial \mathrm{y}_{\mathsf{j}}}, \quad \mathsf{J}_{\mathsf{i,4N+j}} = \sqrt{\lambda_1}\,a_{\mathsf{j}}\frac{\partial\Phi(c_{\mathsf{j}}, r_{\mathsf{i,j}})}{\partial \mathrm{z}_{\mathsf{j}}}.$$

For $\mathsf{i} = \mathsf{M}+1, \ldots, \mathsf{M}+\mathsf{M}_{\rm bry},\ \mathsf{j} = 1, \ldots, \mathsf{N}$,

$$\mathsf{J}_{\mathsf{i,j}} = \sqrt{\lambda_2}\,\frac{\partial\Phi}{\partial n}(c_{\mathsf{j}}, r_{\mathsf{i,j}}) = \sqrt{\lambda_1}\,\left(n_x\frac{\partial\Phi}{\partial x} + n_y\frac{\partial\Phi}{\partial y} + n_z\frac{\partial\Phi}{\partial z}\right)(c_{\mathsf{j}}, r_{\mathsf{i,j}}),$$
$$\mathsf{J}_{\mathsf{i,N+j}} = \sqrt{\lambda_2}\,a_{\mathsf{j}}\frac{\partial}{\partial c_{\mathsf{j}}}\left(\frac{\partial\Phi}{\partial n}(c_{\mathsf{j}}, r_{\mathsf{i,j}})\right) = \sqrt{\lambda_2}\,a_{\mathsf{j}}\frac{\partial}{\partial c_{\mathsf{j}}}\left(\left(n_x\frac{\partial\Phi}{\partial x} + n_y\frac{\partial\Phi}{\partial y} + n_z\frac{\partial\Phi}{\partial z}\right)(c_{\mathsf{j}}, r_{\mathsf{i,j}})\right),$$
$$\mathsf{J}_{\mathsf{i,2N+j}} = \sqrt{\lambda_2}\,a_{\mathsf{j}}\frac{\partial}{\partial \mathrm{x}_{\mathsf{j}}}\left(\frac{\partial\Phi}{\partial n}(c_{\mathsf{j}}, r_{\mathsf{i,j}})\right) = \sqrt{\lambda_2}\,a_{\mathsf{j}}\frac{\partial}{\partial \mathrm{x}_{\mathsf{j}}}\left(\left(n_x\frac{\partial\Phi}{\partial x} + n_y\frac{\partial\Phi}{\partial y} + n_z\frac{\partial\Phi}{\partial z}\right)(c_{\mathsf{j}}, r_{\mathsf{i,j}})\right),$$
$$\mathsf{J}_{\mathsf{i,3N+j}} = \sqrt{\lambda_2}\,a_{\mathsf{j}}\frac{\partial}{\partial \mathrm{y}_{\mathsf{j}}}\left(\frac{\partial\Phi}{\partial n}(c_{\mathsf{j}}, r_{\mathsf{i,j}})\right) = \sqrt{\lambda_2}\,a_{\mathsf{j}}\frac{\partial}{\partial \mathrm{y}_{\mathsf{j}}}\left(\left(n_x\frac{\partial\Phi}{\partial x} + n_y\frac{\partial\Phi}{\partial y} + n_z\frac{\partial\Phi}{\partial z}\right)(c_{\mathsf{j}}, r_{\mathsf{i,j}})\right),$$
$$\mathsf{J}_{\mathsf{i,4N+j}} = \sqrt{\lambda_2}\,a_{\mathsf{j}}\frac{\partial}{\partial \mathrm{z}_{\mathsf{j}}}\left(\frac{\partial\Phi}{\partial n}(c_{\mathsf{j}}, r_{\mathsf{i,j}})\right) = \sqrt{\lambda_2}\,a_{\mathsf{j}}\frac{\partial}{\partial \mathrm{z}_{\mathsf{j}}}\left(\left(n_x\frac{\partial\Phi}{\partial x} + n_y\frac{\partial\Phi}{\partial y} + n_z\frac{\partial\Phi}{\partial z}\right)(c_{\mathsf{j}}, r_{\mathsf{i,j}})\right).$$

4.4.2. *Results.* In the numerical implementation, we chose $\mathsf{M}_{\text{int}} = 900$, $\mathsf{M}_{\text{bry}} = 700$, $\mathsf{N}_{\text{int}} + \mathsf{N}_{\text{bry}} = 240$, $d_{\min} = 0.1$, $d_{\max} = 2.5$, and also took $L = 400$ interior test points. Some results for various values of `Tol`, λ_1, λ_2, are depicted in Table 7.

	`Tol`	E	$\mathcal{E}$	`niter`	CPU (secs)
$\lambda_1 = \lambda_2 = 10000$					
	1.0(-6)	1.546(-4)	9.245(-7)	2	1.09
	1.0(-8)	1.546(-4)	9.245(-7)	2	1.40
	1.0(-10)	1.546(-4)	9.245(-7)	2	1.40
$\lambda_1 = \lambda_2 = 1000$					
	1.0(-6)	3.626(-4)	2.260(-6)	8	2.67
	1.0(-8)	3.626(-4)	2.260(-6)	8	2.70
	1.0(-10)	3.626(-4)	2.260(-6)	8	2.72
$\lambda_1 = \lambda_2 = 100$					
	1.0(-6)	9.205(-4)	4.909(-6)	8	2.71
	1.0(-8)	6.749(-4)	3.754(-6)	50	18.52
	1.0(-10)	6.749(-4)	3.754(-6)	50	18.76

TABLE 7. Example 4: Results for various values of `Tol` and λ for the Bunny domain.

4.5. **Example 5.** In this final example, we consider the following BVP for the Poisson equation in the unit square $\Omega = (0,1)^2$

$$\Delta u = \mathrm{e}^{x+y} \quad \text{in } \Omega, \tag{4.11a}$$

$$u = 0 \quad \text{on } \partial\Omega. \tag{4.11b}$$

Note that in this case, no exact solution is available and we compare the obtained solution with those computed with the traditional Kansa method and the finite element method (FEM). For the Kansa method, we used the normalized multiquadric (MQ) as the RBF with $L = 300$ interior points and 160 boundary points whilst, for the selection of the shape parameter, we employed the modified Franke formula [24]. The FEM calculations were carried out using the MATLAB© PDE Toolbox with 2705 linear elements. For our proposed method, we chose $\lambda = 1000$, and the parameters initial $[d_{\min}, d_{\max}] = [1.1, 3]$, $\eta = 3$. The profile of the approximate solution of BVP (4.11) obtained with the FEM is shown in Figure 7, while in Table 8 we present the errors $\mathcal{E}$ of our approximations for various numbers of collocation points and centres, with respect to those obtained using the traditional Kansa method and the FEM. From this table we observe that our results are slightly closer to the results obtained with the Kansa method than those obtained with the FEM.

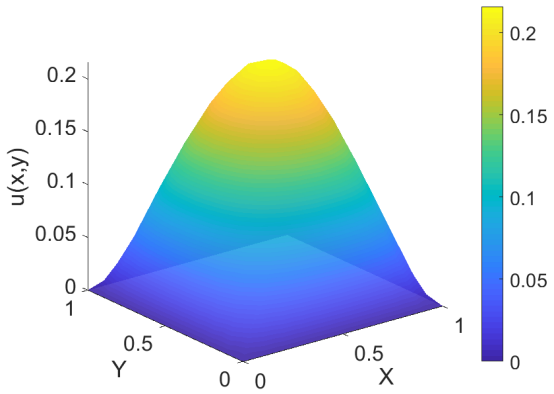


FIGURE 7. Example 4: The profile of the approximate solution using FEM with 2705 linear elements.

M_{int}	M_{bry}	N_{int}	N_{bry}	Kansa method	FEM
250	80	45	40	2.464(-4)	5.832(-4)
300	160	50	60	2.413(-4)	6.768(-4)
400	180	80	60	3.214(-4)	1.066(-3)

TABLE 8. Example 5: Errors $\mathcal{E}$ with respect to the Kansa method and FEM approximations using various numbers of collocation points and centres.

5. Conclusions

In this paper we approximate the solution of Poisson and biharmonic BVPs using an RBF neural network approach. We train the RBF neural network by minimizing a nonlinear cost functional, and this is carried out using the MATLAB© optimization toolbox function `lsqnonlin`. In extensive preliminary tests, we found that the Gaussian RBF is far superior to the multiquadric RBF (MQ) in terms of both accuracy and efficiency. As a result, we only present the results obtained using the Gaussian RBF. In Example 1 we showed that providing the exact Jacobian in `lsqnonlin` yields substantial savings in computational time and this enabled us to solve 3D BVPs in complex domains efficiently. With the help of symbolic computational tools, the (tedious) derivation of the exact Jacobian can be achieved relatively easily. To improve the accuracy of our approach, we extend the domain, in which the initial positions of the centres are placed, to a domain outside the physical boundary of the problem [5, 25]. Moreover, we found that relatively few iterations are required to reach high accuracy. One of the advantages of the proposed approach is that we can obtain the RBF parameters and weights simultaneously through the sophisticated nonlinear least squares solver `lsqnonlin` which greatly simplifies of the solution process. As a result, the challenge of pre–selecting the RBF shape parameter values, which is often problem dependent, is alleviated. Overall, if we choose the initial values of the shape parameters and the source locations properly, we can obtain fairly accurate results efficiently. In view of the fact that we are using

nonlinear minimization anyway, the proposed approach would lend itself naturally to the solution of nonlinear BVPs [19] and inverse geometric BVPs [20]. This will be the subject of a future study.

Acknowledgements

The second author gratefully acknowledges the financial support of the Ministry of Science and Technology (MOST), Taiwan, under the recruitment of visiting science and technology personnel with subsidies (109-2811-E-002-516 and 110-2811-E-002-518).

Appendix

For the 2D Gaussian RBF

$$\Phi(c_{\mathsf{n}}, r_{\mathsf{n}}) = \mathrm{e}^{-c_{\mathsf{n}} r_{\mathsf{n}}^2}$$

we need the derivatives

$$\frac{\partial \Phi}{\partial x}(c_{\mathsf{n}}, r_{\mathsf{n}}, x - \mathrm{x}_{\mathsf{n}}) \;=\; -2c_{\mathsf{n}}(x - \mathrm{x}_{\mathsf{n}})\, \mathrm{e}^{-c_{\mathsf{n}} r_{\mathsf{n}}^2}\;, \quad \frac{\partial \Phi}{\partial y}(c_{\mathsf{n}}, r_{\mathsf{n}}, y - \mathrm{y}_{\mathsf{n}}) \;=\; -2c_{\mathsf{n}}(y - \mathrm{y}_{\mathsf{n}})\, \mathrm{e}^{-c_{\mathsf{n}} r_{\mathsf{n}}^2}\;,$$

$$\frac{\partial^2 \Phi}{\partial x^2}(c_{\mathsf{n}}, r_{\mathsf{n}}, y - \mathrm{y}_{\mathsf{n}}) \;=\; 2c_{\mathsf{n}}\left(2c_{\mathsf{n}}(x - \mathrm{x}_{\mathsf{n}})^2 - 1\right)\mathrm{e}^{-c_{\mathsf{n}} r_{\mathsf{n}}^2}\;, \quad \frac{\partial^2 \Phi}{\partial y^2}(c_{\mathsf{n}}, r_{\mathsf{n}}, x - \mathrm{x}_{\mathsf{n}}) \;=\; 2c_{\mathsf{n}}\left(2c_{\mathsf{n}}(y - \mathrm{y}_{\mathsf{n}})^2 - 1\right)\mathrm{e}^{-c_{\mathsf{n}} r_{\mathsf{n}}^2}\;,$$

$$\Delta\Phi(c_{\mathsf{n}}, r_{\mathsf{n}}) \;=\; 4c_{\mathsf{n}}\left(c_{\mathsf{n}}\, r_{\mathsf{n}}^2 - 1\right)\, \mathrm{e}^{-c_{\mathsf{n}} r_{\mathsf{n}}^2}\;,$$

$$\frac{\partial \Delta\Phi}{\partial x}(c_{\mathsf{n}}, r_{\mathsf{n}}, x - \mathrm{x}_{\mathsf{n}}) \;=\; 8c_{\mathsf{n}}^2\,(x - \mathrm{x}_{\mathsf{n}})\, \left(2 - c_{\mathsf{n}}\, r_{\mathsf{n}}^2\right)\, \mathrm{e}^{-c_{\mathsf{n}} r_{\mathsf{n}}^2}\;,$$

$$\frac{\partial \Delta\Phi}{\partial y}(c_{\mathsf{n}}, r_{\mathsf{n}}, y - \mathrm{y}_{\mathsf{n}}) \;=\; 8c_{\mathsf{n}}^2\,(y - \mathrm{y}_{\mathsf{n}})\, \left(2 - c_{\mathsf{n}}\, r_{\mathsf{n}}^2\right)\, \mathrm{e}^{-c_{\mathsf{n}} r_{\mathsf{n}}^2}\;,$$

$$\Delta^2\Phi(c_{\mathsf{n}}, r_{\mathsf{n}}) \;=\; 16c_{\mathsf{n}}^2\, \left(2 - 4c_{\mathsf{n}}\, r_{\mathsf{n}}^2 + c_{\mathsf{n}}^2\, r_{\mathsf{n}}^4\right)\, \mathrm{e}^{-c_{\mathsf{n}} r_{\mathsf{n}}^2}\;.$$

In addition, we require the following derivatives with respect to the shape parameters c_{n}

$$\frac{\partial \Phi(c_{\mathsf{n}}, r_{\mathsf{n}})}{\partial c_{\mathsf{n}}} = -r_{\mathsf{n}}^2 \mathrm{e}^{-c_{\mathsf{n}} r_{\mathsf{n}}^2},$$

$$\frac{\partial}{\partial c_{\mathsf{n}}}\left(\frac{\partial \Phi(c_{\mathsf{n}}, r_{\mathsf{n}})}{\partial x}\right) = 2(x - \mathrm{x}_{\mathsf{n}})\left(c_{\mathsf{n}}\, r_{\mathsf{n}}^2 - 1\right)\mathrm{e}^{-c_{\mathsf{n}} r_{\mathsf{n}}^2},$$

$$\frac{\partial}{\partial c_{\mathsf{n}}}\left(\frac{\partial \Phi(c_{\mathsf{n}}, r_{\mathsf{n}})}{\partial y}\right) = 2(y - \mathrm{y}_{\mathsf{n}})\left(c_{\mathsf{n}}\, r_{\mathsf{n}}^2 - 1\right)\mathrm{e}^{-c_{\mathsf{n}} r_{\mathsf{n}}^2},$$

$$\frac{\partial \Delta\Phi(c_{\mathsf{n}}, r_{\mathsf{n}})}{\partial c_{\mathsf{n}}} = -4\left(c_{\mathsf{n}}^2 r_{\mathsf{n}}^4 - 4c_{\mathsf{n}} r_{\mathsf{n}}^2 + 1\right)\mathrm{e}^{-c_{\mathsf{n}} r_{\mathsf{n}}^2},$$

$$\frac{\partial \Delta^2\Phi(c_{\mathsf{n}}, r_{\mathsf{n}})}{\partial c_{\mathsf{n}}} = -16c_{\mathsf{n}}\left(c_{\mathsf{n}}^3 r_{\mathsf{n}}^6 - 8c_{\mathsf{n}}^2 r_{\mathsf{n}}^4 + 14c_{\mathsf{n}} r_{\mathsf{n}}^2 - 4\right)\mathrm{e}^{-c_{\mathsf{n}} r_{\mathsf{n}}^2}.$$

We shall also need the following derivatives with respect to the centre coordinates x_{n} and y_{n}

$$\frac{\partial \Phi}{\partial \mathrm{x}_{\mathsf{n}}}(c_{\mathsf{n}}, r_{\mathsf{n}}, x - \mathrm{x}_{\mathsf{n}}) \;=\; 2c_{\mathsf{n}}(x - \mathrm{x}_{\mathsf{n}})\, \mathrm{e}^{-c_{\mathsf{n}} r_{\mathsf{n}}^2}\;, \quad \frac{\partial \Phi}{\partial \mathrm{y}_{\mathsf{n}}}(c_{\mathsf{n}}, r_{\mathsf{n}}, y - \mathrm{y}_{\mathsf{n}}) \;=\; 2c_{\mathsf{n}}(y - \mathrm{y}_{\mathsf{n}})\, \mathrm{e}^{-c_{\mathsf{n}} r_{\mathsf{n}}^2}\;,$$

$$\frac{\partial}{\partial \mathrm{x_n}}\left(\frac{\partial \Phi(c_\mathsf{n}, r_\mathsf{n})}{\partial x}\right) = 2c_\mathsf{n}\left(1-2c_\mathsf{n}(x-\mathrm{x_n})^2\right)\ \mathrm{e}^{-c_\mathsf{n} r_\mathsf{n}^2},$$

$$\frac{\partial}{\partial \mathrm{y_n}}\left(\frac{\partial \Phi(c_\mathsf{n}, r_\mathsf{n})}{\partial y}\right) = 2c_\mathsf{n}\left(1-2c_\mathsf{n}(y-\mathrm{y_n})^2\right)\ \mathrm{e}^{-c_\mathsf{n} r_\mathsf{n}^2},$$

$$\frac{\partial}{\partial \mathrm{x_n}}\left(\frac{\partial \Phi(c_\mathsf{n}, r_\mathsf{n})}{\partial y}\right) = \frac{\partial}{\partial \mathrm{y_n}}\left(\frac{\partial \Phi(c_\mathsf{n}, r_\mathsf{n})}{\partial x}\right) = -4c_\mathsf{n}^2(x-\mathrm{x_n})(y-\mathrm{y_n})\ \mathrm{e}^{-c_\mathsf{n} r_\mathsf{n}^2},$$

$$\frac{\partial \Delta \Phi}{\partial \mathrm{x_n}}(c_\mathsf{n}, r_\mathsf{n}, x-\mathrm{x_n}) \ = \ 8c_\mathsf{n}^2(x-\mathrm{x_n})\left(c_\mathsf{n} r_\mathsf{n}^2-2\right)\ \mathrm{e}^{-c_\mathsf{n} r_\mathsf{n}^2}\ ,$$

$$\frac{\partial \Delta \Phi}{\partial \mathrm{y_n}}(c_\mathsf{n}, r_\mathsf{n}, y-\mathrm{y_n}) \ = \ 8c_\mathsf{n}^2(y-\mathrm{y_n})\left(c_\mathsf{n} r_\mathsf{n}^2-2\right)\ \mathrm{e}^{-c_\mathsf{n} r_\mathsf{n}^2}\ ,$$

$$\frac{\partial \Delta^2 \Phi}{\partial \mathrm{x_n}}(c_\mathsf{n}, r_\mathsf{n}, x-\mathrm{x_n}) \ = \ 32c_\mathsf{n}^3(x-\mathrm{x_n})\left(c_\mathsf{n}^2 r_\mathsf{n}^4-6c_\mathsf{n} r_\mathsf{n}^2+6\right)\ \mathrm{e}^{-c_\mathsf{n} r_\mathsf{n}^2}\ ,$$

$$\frac{\partial \Delta^2 \Phi}{\partial \mathrm{y_n}}(c_\mathsf{n}, r_\mathsf{n}, y-\mathrm{y_n}) \ = 32c_\mathsf{n}^3(y-\mathrm{y_n})\left(c_\mathsf{n}^2 r_\mathsf{n}^4-6c_\mathsf{n} r_\mathsf{n}^2+6\right)\ \mathrm{e}^{-c_\mathsf{n} r_\mathsf{n}^2}\ .$$

For the 3D Gaussian RBF

$$\Phi(c_\mathsf{n}, r_\mathsf{n}) = \mathrm{e}^{-c_\mathsf{n} r_\mathsf{n}^2}$$

we need the following derivatives

$$\frac{\partial \Phi}{\partial x}(c_\mathsf{n}, r_\mathsf{n}, x-\mathrm{x_n}) = -2c_\mathsf{n}(x-\mathrm{x_n})\ \mathrm{e}^{-c_\mathsf{n} r_\mathsf{n}^2}, \quad \frac{\partial \Phi}{\partial y}(c_\mathsf{n}, r_\mathsf{n}, y-\mathrm{y_n}) = -2c_\mathsf{n}(y-\mathrm{y_n})\ \mathrm{e}^{-c_\mathsf{n} r_\mathsf{n}^2},$$

$$\frac{\partial \Phi}{\partial z}(c_\mathsf{n}, r_\mathsf{n}, z-\mathrm{z_n}) = -2c_\mathsf{n}(z-\mathrm{z_n})\ \mathrm{e}^{-c_\mathsf{n} r_\mathsf{n}^2},$$

$$\frac{\partial^2 \Phi}{\partial x^2}(c_\mathsf{n}, r_\mathsf{n}, y-\mathrm{y_n}, z-\mathrm{z_n}) \ = \ 2c_\mathsf{n}\left(2c_\mathsf{n}(x-\mathrm{x_n})^2-1\right)\mathrm{e}^{-c_\mathsf{n} r_\mathsf{n}^2}\ ,$$

$$\frac{\partial^2 \Phi}{\partial y^2}(c_\mathsf{n}, r_\mathsf{n}, x-\mathrm{x_n}, z-\mathrm{z_n}) \ = \ 2c_\mathsf{n}\left(2c_\mathsf{n}(y-\mathrm{y_n})^2-1\right)\mathrm{e}^{-c_\mathsf{n} r_\mathsf{n}^2}\ ,$$

$$\frac{\partial^2 \Phi}{\partial z^2}(c_\mathsf{n}, r_\mathsf{n}, x-\mathrm{x_n}, y-\mathrm{y_n}) \ = \ 2c_\mathsf{n}\left(2c_\mathsf{n}(z-\mathrm{z_n})^2-1\right)\mathrm{e}^{-c_\mathsf{n} r_\mathsf{n}^2}\ ,$$

$$\Delta \Phi(c_\mathsf{n}, r_\mathsf{n}) \ = \ 2c_\mathsf{n}\ \left(2c_\mathsf{n}\, r_\mathsf{n}^2-3\right)\ \mathrm{e}^{-c_\mathsf{n} r_\mathsf{n}^2}\ ,$$

$$\Delta^2 \Phi(c_\mathsf{n}, r_\mathsf{n}) \ = \ 4c_\mathsf{n}^2\ \left(4c_\mathsf{n}^2\, r_\mathsf{n}^4-20c_\mathsf{n}\, r_\mathsf{n}^2+15\right)\ \mathrm{e}^{-c_\mathsf{n} r_\mathsf{n}^2}\ ,$$

and also the derivatives with respect to the shape parameters c_n

$$\frac{\partial \Phi(c_\mathsf{n}, r_\mathsf{n})}{\partial c_\mathsf{n}} = -r_\mathsf{n}^2 \mathrm{e}^{-c_\mathsf{n} r_\mathsf{n}^2},$$

$$\frac{\partial}{\partial c_\mathsf{n}}\left(\frac{\partial \Phi(c_\mathsf{n}, r_\mathsf{n})}{\partial x}\right) = 2(x-\mathrm{x_n})\left(c_\mathsf{n}\, r_\mathsf{n}^2-1\right)\mathrm{e}^{-c_\mathsf{n} r_\mathsf{n}^2},$$

$$\frac{\partial}{\partial c_\mathsf{n}}\left(\frac{\partial \Phi(c_\mathsf{n}, r_\mathsf{n})}{\partial y}\right) = 2(y-\mathrm{y_n})\left(c_\mathsf{n}\, r_\mathsf{n}^2-1\right)\mathrm{e}^{-c_\mathsf{n} r_\mathsf{n}^2},$$

$$\frac{\partial}{\partial c_{\mathsf{n}}}\left(\frac{\partial \Phi(c_{\mathsf{n}}, r_{\mathsf{n}})}{\partial z}\right) = 2(z - \mathrm{z}_{\mathsf{n}})\left(c_{\mathsf{n}}\, r_{\mathsf{n}}^2 - 1\right) \mathrm{e}^{-c_{\mathsf{n}} r_{\mathsf{n}}^2},$$

$$\frac{\partial \Delta\Phi(c_{\mathsf{n}}, r_{\mathsf{n}})}{\partial c_{\mathsf{n}}} = -2\left(2c_{\mathsf{n}}^2 r_{\mathsf{n}}^4 - 7c_{\mathsf{n}} r_{\mathsf{n}}^2 + 3\right) \mathrm{e}^{-c_{\mathsf{n}} r_{\mathsf{n}}^2},$$

$$\frac{\partial \Delta^2\Phi(c_{\mathsf{n}}, r_{\mathsf{n}})}{\partial c_{\mathsf{n}}} = -4c_{\mathsf{n}}\left(4c_{\mathsf{n}}^3 r_{\mathsf{n}}^6 - 36c_{\mathsf{n}}^2 r_{\mathsf{n}}^4 + 75c_{\mathsf{n}} r_{\mathsf{n}}^2 - 30\right) \mathrm{e}^{-c_{\mathsf{n}} r_{\mathsf{n}}^2}.$$

We shall also need the following derivatives with respect to the centre coordinates x_{n}, y_{n} and z_{n}

$$\frac{\partial \Phi}{\partial \mathrm{x}_{\mathsf{n}}}(c_{\mathsf{n}}, r_{\mathsf{n}}, x - \mathrm{x}_{\mathsf{n}}) = 2c_{\mathsf{n}}(x - \mathrm{x}_{\mathsf{n}}) \mathrm{e}^{-c_{\mathsf{n}} r_{\mathsf{n}}^2},$$

$$\frac{\partial \Phi}{\partial \mathrm{y}_{\mathsf{n}}}(c_{\mathsf{n}}, r_{\mathsf{n}}, y - \mathrm{y}_{\mathsf{n}}) = 2c_{\mathsf{n}}(y - \mathrm{y}_{\mathsf{n}}) \mathrm{e}^{-c_{\mathsf{n}} r_{\mathsf{n}}^2},$$

$$\frac{\partial \Phi}{\partial \mathrm{z}_{\mathsf{n}}}(c_{\mathsf{n}}, r_{\mathsf{n}}, z - \mathrm{z}_{\mathsf{n}}) = 2c_{\mathsf{n}}(z - \mathrm{z}_{\mathsf{n}}) \mathrm{e}^{-c_{\mathsf{n}} r_{\mathsf{n}}^2},$$

$$\frac{\partial}{\partial \mathrm{x}_{\mathsf{n}}}\left(\frac{\partial \Phi(c_{\mathsf{n}}, r_{\mathsf{n}})}{\partial x}\right) = 2c_{\mathsf{n}}\left(1 - 2c_{\mathsf{n}}(x - \mathrm{x}_{\mathsf{n}})^2\right) \mathrm{e}^{-c_{\mathsf{n}} r_{\mathsf{n}}^2},$$

$$\frac{\partial}{\partial \mathrm{y}_{\mathsf{n}}}\left(\frac{\partial \Phi(c_{\mathsf{n}}, r_{\mathsf{n}})}{\partial y}\right) = 2c_{\mathsf{n}}\left(1 - 2c_{\mathsf{n}}(y - \mathrm{y}_{\mathsf{n}})^2\right) \mathrm{e}^{-c_{\mathsf{n}} r_{\mathsf{n}}^2},$$

$$\frac{\partial}{\partial \mathrm{z}_{\mathsf{n}}}\left(\frac{\partial \Phi(c_{\mathsf{n}}, r_{\mathsf{n}})}{\partial z}\right) = 2c_{\mathsf{n}}\left(1 - 2c_{\mathsf{n}}(z - \mathrm{z}_{\mathsf{n}})^2\right) \mathrm{e}^{-c_{\mathsf{n}} r_{\mathsf{n}}^2},$$

$$\frac{\partial}{\partial \mathrm{x}_{\mathsf{n}}}\left(\frac{\partial \Phi(c_{\mathsf{n}}, r_{\mathsf{n}})}{\partial y}\right) = \frac{\partial}{\partial \mathrm{y}_{\mathsf{n}}}\left(\frac{\partial \Phi(c_{\mathsf{n}}, r_{\mathsf{n}})}{\partial x}\right) = -4c_{\mathsf{n}}^2(x - \mathrm{x}_{\mathsf{n}})(y - \mathrm{y}_{\mathsf{n}}) \mathrm{e}^{-c_{\mathsf{n}} r_{\mathsf{n}}^2},$$

$$\frac{\partial}{\partial \mathrm{x}_{\mathsf{n}}}\left(\frac{\partial \Phi(c_{\mathsf{n}}, r_{\mathsf{n}})}{\partial z}\right) = \frac{\partial}{\partial \mathrm{z}_{\mathsf{n}}}\left(\frac{\partial \Phi(c_{\mathsf{n}}, r_{\mathsf{n}})}{\partial x}\right) = -4c_{\mathsf{n}}^2(x - \mathrm{x}_{\mathsf{n}})(z - \mathrm{z}_{\mathsf{n}}) \mathrm{e}^{-c_{\mathsf{n}} r_{\mathsf{n}}^2},$$

$$\frac{\partial}{\partial \mathrm{y}_{\mathsf{n}}}\left(\frac{\partial \Phi(c_{\mathsf{n}}, r_{\mathsf{n}})}{\partial z}\right) = \frac{\partial}{\partial \mathrm{z}_{\mathsf{n}}}\left(\frac{\partial \Phi(c_{\mathsf{n}}, r_{\mathsf{n}})}{\partial y}\right) = -4c_{\mathsf{n}}^2(y - \mathrm{y}_{\mathsf{n}})(z - \mathrm{z}_{\mathsf{n}}) \mathrm{e}^{-c_{\mathsf{n}} r_{\mathsf{n}}^2},$$

$$\frac{\partial \Delta\Phi}{\partial \mathrm{x}_{\mathsf{n}}}(c_{\mathsf{n}}, r_{\mathsf{n}}, x - \mathrm{x}_{\mathsf{n}}) = 4c_{\mathsf{n}}^2(x - \mathrm{x}_{\mathsf{n}})\left(2c_{\mathsf{n}} r_{\mathsf{n}}^2 - 5\right) \mathrm{e}^{-c_{\mathsf{n}} r_{\mathsf{n}}^2},$$

$$\frac{\partial \Delta\Phi}{\partial \mathrm{y}_{\mathsf{n}}}(c_{\mathsf{n}}, r_{\mathsf{n}}, y - \mathrm{y}_{\mathsf{n}}) = 4c_{\mathsf{n}}^2(y - \mathrm{y}_{\mathsf{n}})\left(2c_{\mathsf{n}} r_{\mathsf{n}}^2 - 5\right) \mathrm{e}^{-c_{\mathsf{n}} r_{\mathsf{n}}^2},$$

$$\frac{\partial \Delta\Phi}{\partial \mathrm{z}_{\mathsf{n}}}(c_{\mathsf{n}}, r_{\mathsf{n}}, y - \mathrm{y}_{\mathsf{n}}) = 4c_{\mathsf{n}}^2(z - \mathrm{z}_{\mathsf{n}})\left(2c_{\mathsf{n}} r_{\mathsf{n}}^2 - 5\right) \mathrm{e}^{-c_{\mathsf{n}} r_{\mathsf{n}}^2},$$

$$\frac{\partial \Delta^2\Phi}{\partial \mathrm{x}_{\mathsf{n}}}(c_{\mathsf{n}}, r_{\mathsf{n}}, x - \mathrm{x}_{\mathsf{n}}) = 8c_{\mathsf{n}}^3(x - \mathrm{x}_{\mathsf{n}})\left(4c_{\mathsf{n}}^2 r_{\mathsf{n}}^4 - 28c_{\mathsf{n}} r_{\mathsf{n}}^2 + 35\right) \mathrm{e}^{-c_{\mathsf{n}} r_{\mathsf{n}}^2},$$

$$\frac{\partial \Delta^2\Phi}{\partial \mathrm{y}_{\mathsf{n}}}(c_{\mathsf{n}}, r_{\mathsf{n}}, y - \mathrm{y}_{\mathsf{n}}) = 8c_{\mathsf{n}}^3(y - \mathrm{y}_{\mathsf{n}})\left(4c_{\mathsf{n}}^2 r_{\mathsf{n}}^4 - 28c_{\mathsf{n}} r_{\mathsf{n}}^2 + 35\right) \mathrm{e}^{-c_{\mathsf{n}} r_{\mathsf{n}}^2},$$

$$\frac{\partial \Delta^2\Phi}{\partial \mathrm{z}_{\mathsf{n}}}(c_{\mathsf{n}}, r_{\mathsf{n}}, z - \mathrm{z}_{\mathsf{n}}) = 8c_{\mathsf{n}}^3(z - \mathrm{z}_{\mathsf{n}})\left(4c_{\mathsf{n}}^2 r_{\mathsf{n}}^4 - 28c_{\mathsf{n}} r_{\mathsf{n}}^2 + 35\right) \mathrm{e}^{-c_{\mathsf{n}} r_{\mathsf{n}}^2}.$$

References

[1] M. M. Alqezweeni and V. I. Gorbachenko, *Solution of partial differential equations on radial basis functions networks*, EasyChair Preprint No 1964, 2019.

[2] M. M. Alqezweeni, V. I. Gorbachenko, M. V. Zhukov, M. S. Jaafar, *Efficient solving of boundary value problems using radial basis function networks learned by trust region method*, Int. J. Math. Math. Sci. **2018**, Art. ID 9457578.

[3] V. Antonov, D. Tarkhov and A. Vasilyev, *Unified approach to constructing the neural network models of real objects. Part 1.*, Math. Meth. Appl. Sci. **41** (2018), 9244–9251.

[4] C. S. Chen, C. M. Fan and P. H. Wen, *The method of particular solutions for solving certain partial differential equations*, Numer. Methods Partial Differential Equations, **28** (2012), 506–522.

[5] C. S. Chen, A. Karageorghis and F. Dou, *A novel RBF collocation method using fictitious centres*, Appl. Math. Lett. **101** (2020), 106069.

[6] Ch. S. K. Dash, A. K. Behera, S. Dehuri and S.-B. Cho, *Radial basis function neural networks: a topical stat-of-the-art survey*, Open Comput. Sci. **6** (2016), 33–63.

[7] L. N. Elisov, V. I. Gorbachenko and M. V. Zhukov, *Learning radial basis function networks with the trust region method for boundary problems*, Autom. Remote Control **79** (2018), 1621–1629.

[8] G. E. Fasshauer, Meshfree *Approximation Methods with MATLAB, Interdisciplinary Mathematical Sciences*, vol. 6, World Scientific Publishing Co. Pte. Ltd., Hackensack, NJ, 2007.

[9] V. Filippov, L. Elisov and V. Gorbachenko, *Radial basis function networks learning to solve approximation problems*, Int. J. Civ. Eng. Technol. **10** (2019), 872–881.

[10] V. Gorbachenko and M. M. Alqezweeni, *Improving algorithms for learning of radial basis functions networks for approximation problems and solving partial differential equations*, 2019 International Conference on Applied and Engineering Mathematics (ICAEM), 2019, pp. 264–268.

[11] V. I. Gorbachenko and M. M. Alqezweeni, *Learning radial basis functions networks in solving boundary value problems*, 2019 International Russian Automation Conference (RusAutoCon), Sochi, Russia, 2019, pp. 1–6.

[12] V. I. Gorbachenko and M. V. Zhukov, *Solving boundary value problems of mathematical physics using radial basis function networks*, Comput. Math. Math. Phys. **57** (2017), 145–155.

[13] V. I. Gorbachenko, T. V. Lazovskaya, D. A. Tarkhov, A. N. Vasilyev and M. V. Zhukov, *Neural network technique in some inverse problems of mathematical physics*, in Advances in Neural Networks – ISNN 2016, L. Cheng, Q. Liu and A. Ronzhin, Editors, Lecture Notes in Computer Science 9719, Springer International Publishing Switzerland, 2016, pp. 310–316.

[14] S. Haykin, *Neural Networks: A Comprehensive Foundation*, Second Edition, Prentice Hall, 1999.

[15] M. A. Jankowska and A. Karageorghis, *Variable shape parameter Kansa RBF method for the solution of nonlinear boundary value problems*, Eng. Anal. Bound. Elem. **103** (2019), 32–40.

[16] L. Jianyu, L. Siwei, Q. Yingjian and H. Yaping, *Numerical solution of elliptic differential equations using radial basis function neural networks*, Neural Networks **16** (2003), 729–734.

[17] S. Kaennakham, P. Paewpolsong, N. Sriapai and S. Tavaen, *Generalized multiquadric radial basis function neural networks (RBFNs) with variable shape parameters for function recovery*, in Fuzzy Systems and Data Mining VII, A. J. Tallón–Ballesteros, Editor, IOS Press, Amsterdam, 2021, pp. 77–85.

[18] E. J. Kansa, *Multiquadrics—a scattered data approximation scheme with applications to computational fluid-dynamics. II. Solutions to parabolic, hyperbolic and elliptic partial differential equations*, Comput. Math. Appl. **19** (1990), 147–161.

[19] A. Karageorghis, *A time–efficient variable shape parameter Kansa–radial basis function method for the solution of nonlinear boundary value problems*, Appl. Math. Comput. **413** (2022), 126613 (22 pages).

[20] A. Karageorghis, D. Lesnic and L. Marin, The MFS for inverse geometric problems, in L. Marin, L. Munteanu and V. Chiroiu, Editors, *Inverse Problems and Computational Mechanics, Vol. 1*, Editura Academiei, Bucharest, Romania, 2011, pp 191-216.

[21] A. Karageorghis, D. Tappoura and C. S. Chen, *Kansa RBF method with auxiliary boundary centres for fourth order boundary value problems*, Math. Comput. Simul. **181** (2021), 581–597.
[22] A. Katsiamis and A. Karageorghis, *Kansa radial basis function method with fictitious centres for solving nonlinear boundary value problems*, Eng. Anal. Bound. Elem. **119** (2020), 293–301.
[23] M. Kumar and N. Yadav, *Multilayer perceptrons and radial basis function neural network methods for the solution of differential equations: A survey*, Comput. Math. Appl. **62** (2011), 3796–3811.
[24] L.H. Kuo, *On the Selection of a Good Shape Parameter for RBF Approximation and its Applications for Solving PDEs*, Ph.D. dissertation, University of Southern Mississippi, 2015.
[25] L. Ling, R. Opfer and R. Schaback, *Results on meshless collocation techniques*, Eng. Anal. Bound. Elem. **30** (2006), 247–253.
[26] N. Mai-Duy and T. Tran-Cong, *Solving biharmonic problems with scattered-point discretization using indirect radial-basis-function networks*, Eng. Anal. Bound. Elem. **30** (2006), 77–87.
[27] A. P. Markopoulos, S. Georgiopoulos and D. E. Manolakos, *On the use of back propagation and radial basis function neural networks in surface roughness prediction*, J. Ind. Eng. Int. **12** (2016), 389–400.
[28] The MathWorks, Inc., 3 Apple Hill Dr., Natick, MA, *Matlab*.
[29] S. A. Sarra and D. Sturgill, *A random variable shape parameter strategy for radial basis function approximation methods*, Eng. Anal. Bound. Elem. **33** (2009), 1239–1245.
[30] D. A. Stenkin and V. I. Gorbachenko, *Solving equations describing processes in a piecewise homogeneous medium on radial basis functions networks*, in Advances in Neural Computation, Machine Learning, and Cognitive Research IV, B. Kryshanovsky, W. Dunin-Barkowski, V. Redko and Y. Tiumentsev, Editors, Studies in Computational Inteligence 925, Springer Nature Switzerland AG, 2021, pp. 412–418.
[31] N. Yadav, A. Yadav and M. Kumar, *An Introduction to Neural Network Methods for Differential Equations*, Springer Briefs in Applied Sciences and Technology, Computational Intelligence, Springer, 2015.
[32] H Zheng, C. Zhang and Z. Yang, *A local radial basis function collocation method for band structure computation of 3D phononic crystals*, Appl. Math. Model. **77** (2020), 1954–1964.
[33] `http://graphics.stanford.edu/data/3Dscanrep/`

Department of Mathematics and Statistics, University of Cyprus/ Πανεπιστημιο Κυπρου,
P.O.Box 20537, 1678 Nicosia/Λευκωσια, Cyprus/Κυπρος
E-mail address: `andreask@ucy.ac.cy`

Department of Civil Engineering, National Taiwan University, Taiwan & School of Mathematics and Natural Sciences, University of Southern Mississippi, Hattiesburg, MS 39406, USA
E-mail address: `cs.chen@usm.edu`