Commun. Comput. Phys.
doi: 10.4208/cicp.OA-2018-0031

Vol. 25, No. 1, pp. 41-67
January 2019

Local RBF Algorithms for Elliptic Boundary Value Problems in Annular Domains

C. S. Chen[1,*] and Andreas Karageorghis[2]

[1] *School of Civil Engineering and Architecture, Nanchang University, Nanchang, China, and School of Mathematics and Natural Sciences, University of Southern Mississippi, Hattiesburg, MS 39406, USA.*
[2] *Department of Mathematics and Statistics, University of Cyprus, P.O. Box 20537, 1678 Nicosia, Cyprus.*

Received 2 February 2018; Accepted (in revised version) 30 April 2018

Abstract. A local radial basis function method (LRBF) is applied for the solution of boundary value problems in annular domains governed by the Poisson equation, the inhomogeneous biharmonic equation and the inhomogeneous Cauchy-Navier equations of elasticity. By appropriately choosing the collocation points we obtain linear systems in which the coefficient matrices possess block sparse circulant structures and which can be solved efficiently using matrix decomposition algorithms (MDAs) and fast Fourier transforms (FFTs). The MDAs used are appropriately modified to take into account the sparsity of the arrays involved in the discretization. The leave-one-out cross validation (LOOCV) algorithm is employed to obtain a suitable value for the shape parameter in the radial basis functions (RBFs) used. The selection of the nearest centres for each local influence domain is carried out using a modification of the kdtree algorithm. In several numerical experiments, it is demonstrated that the proposed algorithm is both accurate and capable of solving large scale problems.

AMS subject classifications: 65N35, 65N22

Key words: Radial basis functions, Kansa method, Poisson equation, biharmonic equation, Cauchy-Navier equations of elasticity, matrix decomposition algorithms, fast Fourier transforms.

1 Introduction

The local radial basis function (LRBF) method was first discussed in [30] and then independently introduced in [28,31,32,34] introduced, see also [21]. In contrast to the traditional meshed based methods [1,2], LRBF is a meshless method which may be viewed as

*Corresponding author. *Email addresses:* `cschen.math@gmail.com` (C. S. Chen), `andreask@ucy.ac.cy` (A. Karageorghis) a special case of the Kansa method [15]. Meshless methods are well-suited for the numerical solution of boundary and initial value problems in two and three dimensions. Unlike the global Kansa-radial basis function (RBF) method [6–8] which leads to dense and poorly conditioned linear systems, the LRBF method leads to sparse systems. In recent years, the LRBF method has been successfully applied to a large variety of problems in science and engineering. No special treatment for pre-conditioning is required [22,23,33].

Efficient global Kansa-RBF algorithms for problems in geometries possessing radial symmetry were proposed in [18–20, 25]. These algorithms are *matrix decomposition algorithms (MDAs)* [4, 5] and make use of fast Fourier transforms (FFTs). This allows us to decompose a large system into a series of small systems which can be solved efficiently and thus the issue of ill-conditioning occurring for large dense matrices is resolved. However, when the number of collocation points is sufficiently large, the rank of the smaller decomposed matrices could still be large. In such a case, the memory space required to store these (not so small) matrices as well as the computational cost are still challenging issues. Furthermore, when the rank of the decomposed matrices becomes larger, the computational cost of finding a suitable shape parameter using LOOCV which is adopted in [25] increases rapidly. To alleviate these difficulties for very large-scale problems, a localized RBF method can be considered so that the decomposed matrices are sparse. Our goal in this work is to formulate the MDAs developed in [25] for the global Kansa-RBF method, for the LRBF method. As will be demonstrated, this leads to very efficient algorithms which exploit *both the structure and the sparsity* of the matrices involved, and thus to substantial savings in both computer time and memory. The nearest centres for each local influence domain in the LRBF method are selected using a modification of the *kdtree algorithm* [29]. While the emphasis of this paper is not on the determination of the optimal value of the shape parameter, we use the leave-one-out cross validation (LOOCV) algorithm [27] as a tool for determining appropriate values of the shape parameter which yield to satisfactorily accurate results. In the numerical examples examined in this paper, we shall focus on the use of the normalized multiquadric (MQ) while stressing that the proposed algorithms are applicable to other RBFs.

The paper is organized as follows. In Section 2 we present the three types of boundary value problems to be considered in the paper, namely Poisson, biharmonic and linear elasticity problems. Some important implementational issues related to the proposed technique are addressed in Section 3. A description of the LRBF method and corresponding MDA for Poisson problems is provided in Section 4 and its extension to biharmonic problems in Section 5. The LRBF method and corresponding MDA for linear elasticity problems is presented in Section 6. In Section 7 the proposed method is applied to several numerical examples and the results analyzed. Finally, some conclusions and ideas for future work are given in Section 8.

2 The problems

In all problems considered the domain Ω is the annulus

$$\Omega=\left\{\boldsymbol{x}\in\mathbb{R}^2:\varrho_1<|\boldsymbol{x}|<\varrho_2\right\}, \tag{2.1}$$

where $\partial\Omega=\partial\Omega_1\cup\partial\Omega_2$, $\partial\Omega_1\cap\partial\Omega_2=\emptyset$ and $\partial\Omega_1=\left\{\boldsymbol{x}\in\mathbb{R}^2:|\boldsymbol{x}|=\varrho_1\right\}$ and $\partial\Omega_2=\left\{\boldsymbol{x}\in\mathbb{R}^2:|\boldsymbol{x}|=\varrho_2\right\}$.

2.1 Poisson problems

We first consider the Poisson equation

$$\Delta u=f \quad \text{in } \Omega, \tag{2.2a}$$

subject to the boundary conditions

$$u=g_D \quad \text{on } \partial\Omega, \tag{2.2b}$$

or

$$\frac{\partial u}{\partial \mathbf{n}}=g_N \quad \text{on } \partial\Omega_1 \qquad u=g_D \quad \text{on } \partial\Omega_2, \tag{2.2c}$$

where f,g_D and g_N are given functions. In (2.2c) and throughout the paper, $\partial/\partial\mathrm{n}$ denotes the derivative along the outward unit normal vector to the boundary denoted by $\mathbf{n}=(\mathrm{n_x},\mathrm{n_y})$.

2.2 Biharmonic problems

We next consider the biharmonic equation

$$\Delta^2 u=f \quad \text{in } \Omega, \tag{2.3a}$$

subject to either the boundary conditions

$$u=g_D \quad \text{and} \quad \frac{\partial u}{\partial \mathrm{n}}=g_N \quad \text{on } \partial\Omega, \tag{2.3b}$$

or the boundary conditions

$$u=g_D \quad \text{and} \quad \Delta u=g_L \quad \text{on } \partial\Omega, \tag{2.3c}$$

where f, g_D, g_N and g_L are given functions. Note that boundary value problem (2.3a)-(2.3b) is known as the *first biharmonic problem* while the boundary value problem consisting of (2.3a) and (2.3c) is known as the *second biharmonic problem*.

2.3 Linear elasticity problems

We finally consider the Cauchy–Navier system for linear elasticity (see, e.g. [13])

$$\begin{cases} \mathcal{L}_1(u_1,u_2)=\mu\Delta u_1+\dfrac{\mu}{1-2\nu}\left(\dfrac{\partial^2 u_1}{\partial x^2}+\dfrac{\partial^2 u_2}{\partial x\partial y}\right)=f_1, \\ \mathcal{L}_2(u_1,u_2)=\dfrac{\mu}{1-2\nu}\left(\dfrac{\partial^2 u_1}{\partial x\partial y}+\dfrac{\partial^2 u_2}{\partial y^2}\right)+\mu\Delta u_2=f_2, \end{cases} \quad \text{in } \Omega, \tag{2.4a}$$

subject to either the Dirichlet boundary conditions

$$u_1=g_{D_1} \quad \text{and} \quad u_2=g_{D_2} \quad \text{on } \partial\Omega, \tag{2.4b}$$

or the mixed boundary conditions

$$t_1=g_{N_1}, \ \ t_2=g_{N_2} \quad \text{on } \partial\Omega_1, \quad \text{and} \quad u_1=g_{D_1}, \ \ u_2=g_{D_2} \quad \text{on } \partial\Omega_2, \tag{2.4c}$$

where the domain Ω is defined in (2.1), the boundary segments $\partial\Omega_i$, $i=1,2$, are defined as in Section 2.1, and f_i, g_{D_i}, g_{N_i}, $i=1,2$, are given functions. In (2.4), (u_1,u_2) denote the displacements and (t_1,t_2) denote the tractions which are defined by [13]

$$\begin{aligned} t_1&=2\mu\left[\left(\frac{1-\nu}{1-2\nu}\right)\frac{\partial u_1}{\partial x}+\left(\frac{\nu}{1-2\nu}\right)\frac{\partial u_2}{\partial y}\right]\mathrm{n_x}+\mu\left[\frac{\partial u_1}{\partial y}+\frac{\partial u_2}{\partial x}\right]\mathrm{n_y}, \\ t_2&=\mu\left[\frac{\partial u_1}{\partial y}+\frac{\partial u_2}{\partial x}\right]\mathrm{n_x}+2\mu\left[\left(\frac{\nu}{1-2\nu}\right)\frac{\partial u_1}{\partial x}+\left(\frac{1-\nu}{1-2\nu}\right)\frac{\partial u_2}{\partial y}\right]\mathrm{n_y}. \end{aligned}$$

In (2.4a) the constant $\nu\in[0,1/2)$ is Poisson's ratio and the constant $\mu>0$ is the shear modulus.

For convenience, we shall write the Cauchy-Navier equations (2.4a) as

$$\begin{cases} \mathcal{L}_1^1(u_1)+\mathcal{L}_1^2(u_2)=f_1, \\ \mathcal{L}_2^1(u_1)+\mathcal{L}_2^2(u_2)=f_2, \end{cases} \quad \text{in } \Omega, \tag{2.5}$$

where the linear operators $\mathcal{L}_i^j$, $i,j=1,2$, are defined as

$$\mathcal{L}_1^1\equiv\mu\Delta+\frac{\mu}{1-2\nu}\frac{\partial^2}{\partial x^2}, \quad \mathcal{L}_1^2\equiv\frac{\mu}{1-2\nu}\frac{\partial^2}{\partial x\partial y}, \quad \mathcal{L}_2^1\equiv\mathcal{L}_1^2, \quad \mathcal{L}_2^2\equiv\mu\Delta+\frac{\mu}{1-2\nu}\frac{\partial^2}{\partial y^2}.$$

3 Implementational considerations

3.1 Preliminaries

In the application of the LRBF the solution of the problem is expressed in terms of RBFs. An RBF $\phi(x,y)$ is a function which may be written in the form

$$\phi(x,y)=\Phi(r), \quad \text{where} \quad r^2=(x-\mathrm{x})^2+(y-\mathrm{y})^2. \tag{3.1}$$

Thus each RBF ϕ is associated with a point (x,y) which is usually referred to as a *center*. In addition to the centres we consider the *collocation points* where the differential equations and boundary conditions are collocated. Note that, in general, the numbers and locations of the collocation points and the centres differ and the number of centres is normally taken to be less than the number of collocation points. In the current work the centres and the collocation points are the same. Often, RBFs contain a parameter called the *shape parameter*, the optimal determination of which remains a major challenge.

3.2 Distribution of collocation points

First, we distribute a set of $\mathcal{K}$ collocation points (centres) $\mathcal{X}=(\boldsymbol{x}_i)_{i=1}^{\mathcal{K}}$ in $\overline{\Omega}$ in the following way. We define the M angles

$$\vartheta_m=\frac{2\pi(m-1)}{M}, \quad m=1,\cdots,M, \tag{3.2}$$

and the N radii

$$\mathrm{r}_n=\varrho_1+(\varrho_2-\varrho_1)\frac{n-1}{N-1}, \quad n=1,\cdots,N. \tag{3.3}$$

The collocation points $\{(x_{mn},y_{mn})\}_{m=1,n=1}^{M,N}$ are defined as follows:

$$x_{mn}=\mathrm{r}_n\cos\left(\vartheta_m+\frac{2\pi\alpha_n}{M}\right), \quad y_{mn}=\mathrm{r}_n\sin\left(\vartheta_m+\frac{2\pi\alpha_n}{M}\right), \quad m=1,\cdots,M, \quad n=1,\cdots,N. \tag{3.4}$$

In (3.4) the parameters $\{\alpha_n\}_{n=1}^{N}\in[-1/2,1/2]$ correspond to rotations of the collocation points which lie on concentric circles and may be used to produce more uniform distributions.

The $\mathcal{K}_{\text{int}}$ *interior points* $(\boldsymbol{x}_i)_{i=1}^{\mathcal{K}_{\text{int}}}$ are taken as

$$\boldsymbol{x}_{(n-2)M+m}=(x_{mn},y_{mn}), \quad m=1,\cdots,M, \quad n=2,\cdots,N-1, \tag{3.5}$$

and the $\mathcal{K}_{\text{bry}}$ *boundary points* $(\boldsymbol{x}_i)_{i=\mathcal{K}_{\text{int}}+1}^{\mathcal{K}_{\text{int}}+\mathcal{K}_{\text{bry}}}$ as

$$\boldsymbol{x}_{(N-2)M+m}=(x_{m1},y_{m1}), \quad \text{and} \quad \boldsymbol{x}_{(N-1)M+m}=(x_{mN},y_{mN}), \quad m=1,\cdots,M, \tag{3.6}$$

where, clearly, $\mathcal{K}_{\text{int}}=(N-2)M$, $\mathcal{K}_{\text{bry}}=2M$ and $\mathcal{K}=MN$. The points $(\boldsymbol{x}_i)_{i=\mathcal{K}_{\text{int}}+1}^{\mathcal{K}_{\text{int}}+\mathcal{K}_{\text{bry}_1}}$ are the boundary points on $\partial\Omega_1$ and the points $(\boldsymbol{x}_i)_{i=\mathcal{K}_{\text{int}}+\mathcal{K}_{\text{bry}_1}+1}^{\mathcal{K}}$ are the boundary points on $\partial\Omega_2$. Clearly, $\mathcal{K}_{\text{bry}}=\mathcal{K}_{\text{bry}_1}+\mathcal{K}_{\text{bry}_2}$ and $\mathcal{K}=\mathcal{K}_{\text{int}}+\mathcal{K}_{\text{bry}_1}+\mathcal{K}_{\text{bry}_2}$.

3.3 Selection of shape parameter

In the applications of numerical methods which use RBFs, it is important to choose a suitable RBF. Moreover, in the cases where these RBFs involve a shape parameter, it is crucial and challenging to determine an appropriate value of the shape parameter which yields accurate results [9,16,27]. As stated in Section 1, the emphasis of the current study is not on the determination of the optimal value of the shape parameter. In the current study, we will use the so called LOOCV algorithm proposed by Rippa [27] to find a sub-optimal value of the shape parameter. Certain MATLAB© codes for LOOCV may be found in [12]. As described in [25], we shall use the local minimizer MATLAB© function `fminbnd` to find the minimum of a function of one variable within a fixed interval. Initial guesses for the lower and upper bounds denoted by `min` and `max` need to be provided so that the search takes place in the interval [`min`, `max`]. An attractive feature of LOOCV is that the exact solution of the given problem does not need to be known in its application for the selection of a (sub-optimal) shape parameter.

In this work, we shall determine an appropriate value for the shape parameter by considering the solution of the *local* systems in the LRBF. We shall describe this technique by considering the Dirichlet Poisson problem (2.2a) with boundary conditions (2.2b) or (2.2c). For the local system (4.10), to find a suitable shape parameter, we modify the MATLAB© code in [12] as follows:

```
function ceps =costEps(c,RBF,D,LapRBF,DM)
A=RBF(DM,c);
rhs=LapRBF(D,c);
invA=pinv(A);
errorvector=(invA*rhs)./diag(invA);
ceps=norm(errorvector);
```

where `DM` is the distancematrix B_i in (4.10) and `D` is the distance vector in (4.8). For the normalized MQ, `RBF` and `LapRBF` are given as follows:

```
RBF = @(r,c) sqrt(1+(c*r).^2);
LapRBF = @ (r,c) c*c*((r*c).^2+2)./((r*c).^2+1).^(3/2);
```

The cost function `costEps` is given by

```
[c,fval] = fminbnd(@(c) costEps(c,RBF,D,LapRBF,DM),minc,maxc);
```

where `fminbnd` is the MATLAB© function finding the minimum of the cost function for the shape parameter `c` and `minc` and `maxc` define the initial search interval of the shape parameter.

In some instances it was sufficient to determine the shape parameter by considering LOOCV in a single randomly chosen local influence domain but in other instances this lead to poor results. We therefore adopted a different strategy in which the shape parameter was taken to be equal to the average of the LOOCV parameters obtained in each of the local influence domains involved. Because of the circulant structure of the global matrix (see (4.16)), we only need to find the average of N local influence domains (where N is defined in (3.3)) and, since each local matrix in (4.10) is relatively small, the additional computational cost is negligible.

3.4 Efficient selection of neighbouring points

As described in Section 3.2, an important feature of the LRBF method is the selection of the κ nearest centres for each local influence domain. One way of efficiently selecting these points, as proposed in [36], is to use the *kdtree algorithm* [29, page 389], see also [11, Appendix A.2]. In particular, we use the MATLAB© executable (MEX) functions `kdtree_build` and `kdtree_k_nearest-neighbors`. Despite the efficiency of such a search algorithm, it should be pointed out that this procedure can be costly when the number of collocation points is sufficiently large and that the most time-consuming part of the algorithm is the building of the tree (`kdtree_build`) before the search. For example, it requires approximately thirty minutes to build the kdtree for the case of one million collocation points. An advantage of the proposed algorithm is that for every circulant submatrix this selection needs to be carried out only once. Moreover, the search algorithm is made more efficient by only to conducting the search in a small section of the circular domain as shown in Fig. 1. More specifically, since each point has a specific index number between 1 to MN, we store only the indices of the collocation points in the red region in Fig. 1. The tree is built for the relatively small number of these points and then the search process is carried out. In this way, the computational cost in the case of a very large number of collocation points is significantly reduced.

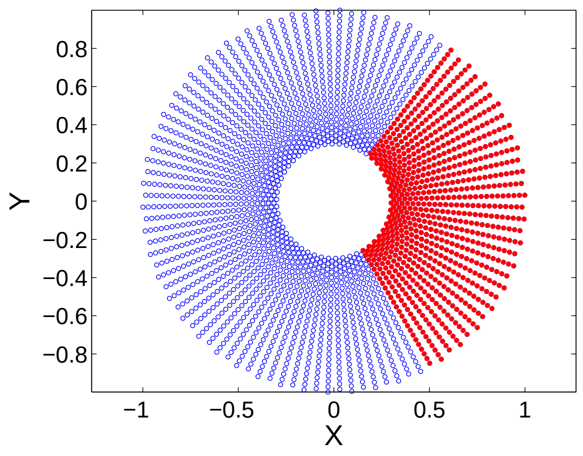


Figure 1: Reduced search points (red).

4 Poisson problems

4.1 The LRBF method

In this section we describe the LRBF method for Poisson problems. It should be noted that the description that follows could be applied to any equation of the form $\mathcal{L}u=f$ where $\mathcal{L}$ is a second order linear elliptic operator by simply replacing Δ by $\mathcal{L}$. However, the LRBF–MDA proposed in this work is only applicable to the Laplace and Helmholtz operators.

Let $\mathcal{X}=(\boldsymbol{x}_i)_{i=1}^{N}$ be a set of collocation points in $\overline{\Omega}$. For point $\boldsymbol{x}_i\in\Omega$, we choose the set of κ nearest neighbouring points denoted by $\mathcal{X}_i=(\boldsymbol{x}_\ell^i)_{\ell=1}^{\kappa}$ which lie in the *local influence domain* of $\boldsymbol{x}_i$. The points of $\mathcal{X}_i$ are indexed locally as $\boldsymbol{x}_\ell^i=\boldsymbol{x}_{\ell(i)}$ and these overlapping sets of points $(\mathcal{X}_i)_{i=1}^{\mathcal{K}}$ are such that $(\boldsymbol{x}_i)_{i=1}^{\mathcal{K}}=\bigcup_{i=1}^{\mathcal{K}}\mathcal{X}_i=\mathcal{X}$. We next consider collocating in the local influence domain $\mathcal{X}_i$. For $\boldsymbol{x}\in\mathcal{X}_i$, the approximate solution of the given partial differential equation can be written as

$$u_{\mathcal{K}}(\boldsymbol{x})=\sum_{\ell=1}^{\kappa}\alpha_\ell^i\Phi(||\boldsymbol{x}-\boldsymbol{x}_\ell^i||),\quad \boldsymbol{x}\in\mathcal{X}_i, \tag{4.1}$$

where the function Φ is an appropriately chosen RBF and the coefficients $\alpha_\ell^i=\alpha_{\ell(i)}$ are indexed accordingly.

Next, we collocate the *local* approximation at all the points of $\mathcal{X}_i$,

$$u_{\mathcal{K}}(\boldsymbol{x}_j^i)=\sum_{\ell=1}^{\kappa}\alpha_\ell^i\Phi(||\boldsymbol{x}_j^i-\boldsymbol{x}_\ell^i||),\quad \text{for}\quad j=1,\cdots,\kappa, \tag{4.2}$$

which generates the *local* system

$$\boldsymbol{u}_{\mathcal{K}}^i=\begin{bmatrix} u_{\mathcal{K}}(\boldsymbol{x}_1^i)\\ u_{\mathcal{K}}(\boldsymbol{x}_2^i)\\ \vdots\\ u_{\mathcal{K}}(\boldsymbol{x}_\kappa^i)\end{bmatrix}=\begin{bmatrix}\Phi(||\boldsymbol{x}_1^i-\boldsymbol{x}_1^i||) & \Phi(||\boldsymbol{x}_1^i-\boldsymbol{x}_2^i||) & \cdots & \Phi(||\boldsymbol{x}_1^i-\boldsymbol{x}_\kappa^i||)\\ \Phi(||\boldsymbol{x}_2^i-\boldsymbol{x}_1^i||) & \Phi(||\boldsymbol{x}_2^i-\boldsymbol{x}_2^i||) & \cdots & \Phi(||\boldsymbol{x}_2^i-\boldsymbol{x}_\kappa^i||)\\ \vdots & \vdots & \vdots & \vdots\\ \Phi(||\boldsymbol{x}_\kappa^i-\boldsymbol{x}_1^i||) & \Phi(||\boldsymbol{x}_\kappa^i-\boldsymbol{x}_2^i||) & \cdots & \Phi(||\boldsymbol{x}_\kappa^i-\boldsymbol{x}_\kappa^i||)\end{bmatrix}\begin{bmatrix}\alpha_1^i\\ \alpha_2^i\\ \vdots\\ \alpha_\kappa^i\end{bmatrix}=B_i\boldsymbol{\alpha}^i, \tag{4.3}$$

where the matrix $B_i\in\mathbb{R}^{\kappa\times\kappa}$ is clearly symmetric and the vector $\boldsymbol{\alpha}^i\in\mathbb{R}^{\kappa\times 1}$ is the vector of coefficients. From approximation (4.1) for the centre and (4.3), we can write

$$u_{\mathcal{K}}(\boldsymbol{x}_i)=\sum_{\ell=1}^{\kappa}\alpha_\ell^i\Phi(||\boldsymbol{x}_i-\boldsymbol{x}_\ell^i||)=(\boldsymbol{h}_\Phi^i)^T\boldsymbol{\alpha}^i=(\boldsymbol{h}_\Phi^i)^TB_i^{-1}\boldsymbol{u}_{\mathcal{K}}^i=\left(\boldsymbol{w}^i\right)^T\boldsymbol{u}_{\mathcal{K}}^i, \tag{4.4}$$

where

$$\boldsymbol{h}_\Phi^i=\Big[\Phi(||\boldsymbol{x}_i-\boldsymbol{x}_1^i||),\Phi(||\boldsymbol{x}_i-\boldsymbol{x}_2^i||),\cdots,\Phi(||\boldsymbol{x}_i-\boldsymbol{x}_\kappa^i||)\Big]^T,\quad \boldsymbol{w}^i=B_i^{-1}\boldsymbol{h}_\Phi^i, \tag{4.5}$$

and

$$\boldsymbol{u}_{\mathcal{K}}^i=\left[u_{\mathcal{K}}(\boldsymbol{x}_{1(i)}),u_{\mathcal{K}}(\boldsymbol{x}_{2(i)}),\cdots,u_{\mathcal{K}}(\boldsymbol{x}_{\kappa(i)})\right]^T.$$

In (4.4)-(4.5) we have used the fact that since the matrix B_i is symmetric so is its inverse. Note that from (4.4), it follows that the value of $u_{\mathcal{K}}(\boldsymbol{x}_i)$ is expressed as a weighted average of the values of the $u_{\mathcal{K}}(\boldsymbol{x}_\ell^i),\ell=1,\cdots,\kappa$, that is, $u(\boldsymbol{x}_i)$ is approximated by a weighted average of the values of the neighbouring $u(\boldsymbol{x}_\ell^i),\ell=1,\cdots,\kappa$. The invertibility of the matrix B_i is discussed in, e.g., [35].

In the application of the LRBF method for the solution of boundary value problem (2.2) we satisfy the Poisson equation at each interior point of Ω in $\mathcal{X}$, that is

$$\Delta u_{\mathcal{K}}(\boldsymbol{x})=f(\boldsymbol{x}),\quad \boldsymbol{x}\in\mathcal{X}_i, \tag{4.6}$$

or, locally, for $\boldsymbol{x}=\boldsymbol{x}_i$,

$$\Delta u_{\mathcal{K}}(\boldsymbol{x}_i)=\sum_{\ell=1}^{\kappa}\alpha_\ell^i\Delta\Phi(||\boldsymbol{x}_i-\boldsymbol{x}_\ell^i||)=(\boldsymbol{h}_{\Delta\Phi}^i)^TB_i^{-1}\boldsymbol{u}_{\mathcal{K}}^i=\left(\boldsymbol{w}_\Delta^i\right)^T\boldsymbol{u}_{\mathcal{K}}^i=f(\boldsymbol{x}_i),\quad i=1,\cdots,\mathcal{K}_{\text{int}}, \tag{4.7}$$

where

$$\boldsymbol{h}_{\Delta\Phi}^i=\left[\Delta\Phi(||\boldsymbol{x}_i-\boldsymbol{x}_1^i||),\Delta\Phi(||\boldsymbol{x}_i-\boldsymbol{x}_2^i||),\cdots,\Delta\Phi(||\boldsymbol{x}_i-\boldsymbol{x}_\kappa^i||)\right]^T \tag{4.8}$$

and

$$\boldsymbol{w}_\Delta^i=B_i^{-1}\boldsymbol{h}_{\Delta\Phi}^i. \tag{4.9}$$

The local elements $\boldsymbol{w}_\Delta^i$ can be appropriately placed in the global matrix [36]. Note that (4.7) is essentially a finite difference approximation of the Laplacian of u in terms of the values of u at its neighbouring points, which justifies the fact that the LRBF method is also known as the RBF-finite difference method [32]. In each set $\mathcal{X}_i$, we determine the vector $\boldsymbol{w}_\Delta^i$ by solving the $\kappa\times\kappa$ system

$$B_i\boldsymbol{w}_\Delta^i=\boldsymbol{h}_{\Delta\Phi}^i. \tag{4.10}$$

The vectors $\boldsymbol{w}_\Delta^i$, $i=1,\cdots,\mathcal{K}$, contain all the necessary coefficients of the approximations $(u_{\mathcal{K}}(\boldsymbol{x}_i))_{i=1}^{\mathcal{K}}$ in the global system. However, these need to be distributed accordingly in the global matrix and each will be incorporated into a line of the global matrix which will contain zeros except for the elements of $\boldsymbol{w}_\Delta^i$ at the appropriate positions.

In the case when Neumann boundary conditions are prescribed on $\partial\Omega_1$, we need to evaluate $\partial u_{\mathcal{K}}/\partial x$ and $\partial u_{\mathcal{K}}/\partial y$ at the boundary points of $\partial\Omega_1$. We thus locally calculate

$$\frac{\partial u_{\mathcal{K}}}{\partial x}(\boldsymbol{x}_i)=\sum_{\ell=1}^{\kappa}\alpha_\ell^i\frac{\partial\Phi}{\partial x}(||\boldsymbol{x}_i-\boldsymbol{x}_\ell^i||)=(\boldsymbol{h}_{\Phi_x}^i)^TB_i^{-1}\boldsymbol{u}_{\mathcal{K}}^i=\left(\boldsymbol{w}_x^i\right)^T\boldsymbol{u}_{\mathcal{K}}^i,\quad i=\mathcal{K}_{\text{int}}+1,\cdots,\mathcal{K}_{\text{int}}+\mathcal{K}_{\text{bry}_1}, \tag{4.11}$$

where

$$\boldsymbol{h}_{\Phi_x}^i=\left[\frac{\partial\Phi}{\partial x}(||\boldsymbol{x}_i-\boldsymbol{x}_1^i||),\frac{\partial\Phi}{\partial x}(||\boldsymbol{x}_i-\boldsymbol{x}_2^i||),\cdots,\frac{\partial\Phi}{\partial x}(||\boldsymbol{x}_i-\boldsymbol{x}_\kappa^i||)\right]^T, \tag{4.12}$$

and

$$\boldsymbol{w}_x^i = B_i^{-1}\boldsymbol{h}_{\Phi_x}^i.$$

Clearly, $\partial u_{\mathcal{K}}/\partial y$ can be obtained in a similar way.

At the boundary point $\boldsymbol{x}_i$ on $\partial\Omega_1$ we have that

$$\begin{aligned}\frac{\partial u_{\mathcal{K}}}{\partial \mathrm{n}}(\boldsymbol{x}_i) &= \sum_{\ell=1}^{\kappa}\alpha_\ell^i \frac{\partial\Phi}{\partial \mathrm{n}}(||\boldsymbol{x}_i-\boldsymbol{x}_\ell^i||) = \sum_{\ell=1}^{\kappa}\alpha_\ell^i\left(\frac{\partial\Phi}{\partial x}(||\boldsymbol{x}_i-\boldsymbol{x}_\ell^i||)\mathrm{n}_x(\boldsymbol{x}_i)+\frac{\partial\Phi}{\partial y}(||\boldsymbol{x}_i-\boldsymbol{x}_\ell^i||)\mathrm{n}_y(\boldsymbol{x}_i)\right)\\ &= (\boldsymbol{h}_{\Phi_\mathrm{n}}^i)^T B_i^{-1}\boldsymbol{u}_{\mathcal{K}}^i = \left(\boldsymbol{w}_\mathrm{n}^i\right)^T \boldsymbol{u}_{\mathcal{K}}^i = g_N(\boldsymbol{x}_i), \quad i=\mathcal{K}_{\text{int}}+1,\cdots,\mathcal{K}_{\text{int}}+\mathcal{K}_{\text{bry}_1}, \end{aligned} \tag{4.13}$$

where $\boldsymbol{h}_{\Phi_\mathrm{n}}^i = \mathrm{n}_x(\boldsymbol{x}_i)\boldsymbol{h}_{\Phi_x}^i + \mathrm{n}_y(\boldsymbol{x}_i)\boldsymbol{h}_{\Phi_y}^i$ and $\boldsymbol{w}_\mathrm{n}^i = B_i^{-1}\boldsymbol{h}_{\Phi_\mathrm{n}}^i$. The vectors $\boldsymbol{w}_\mathrm{n}^i$ are determined by solving the $\kappa\times\kappa$ systems $B_i\boldsymbol{w}_\mathrm{n}^i = \boldsymbol{h}_{\Phi_\mathrm{n}}^i$ and their elements are appropriately placed in the global matrix [36].

In the case of Dirichlet boundary conditions, things are considerably easier as we simply need to impose the boundary conditions

$$u_{\mathcal{K}}(\boldsymbol{x}_i) = g_D(\boldsymbol{x}_i), \quad i=\mathcal{K}_{\text{int}}+1,\cdots,\mathcal{K}_{\text{int}}+\mathcal{K}_{\text{bry}}, \tag{4.14}$$

which means that the lines in the global matrix which correspond to these equations will only contain one non-zero element (equal to 1) in the global position of the corresponding boundary point.

By extending the local elements (4.7), (4.13), and (4.14) to global form and placing them in the global matrix, we obtain the following global sparse system [36]:

$$A\boldsymbol{u}_{\mathcal{K}} = \boldsymbol{b}, \tag{4.15}$$

where the matrix $A\in\mathbb{R}^{\mathcal{K}\times\mathcal{K}}$ is sparse. The right hand side vector $\boldsymbol{b}=[b_1,\cdots,b_{\mathcal{K}}]^T$ is defined as follows:

$$\begin{aligned} &b_i = f(\boldsymbol{x}_i), \quad i=1,\cdots,\mathcal{K}_{\text{int}},\\ &b_i = g_D(\boldsymbol{x}_i) \quad \text{or} \quad g_N(\boldsymbol{x}_i), \quad i=\mathcal{K}_{\text{int}}+1,\cdots,\mathcal{K}_{\text{int}}+\mathcal{K}_{\text{bry}_1},\\ &b_i = g_D(\boldsymbol{x}_i), \quad i=\mathcal{K}_{\text{int}}+\mathcal{K}_{\text{bry}_1}+1,\cdots,\mathcal{K}_{\text{int}}+\mathcal{K}_{\text{bry}_1}+\mathcal{K}_{\text{bry}_2}. \end{aligned}$$

The solution of sparse system (4.15) for the vector $\boldsymbol{u}_{\mathcal{K}}\in\mathbb{R}^{\mathcal{K}\times 1}$ yields the approximations to the solution u at the set of centres $\mathcal{X}$. In particular, if we define the vector $\boldsymbol{u}_{\mathcal{K}} = [u_{\mathcal{K}_1},\cdots,u_{\mathcal{K}_{\mathcal{K}}}]^T$, then $u_{\mathcal{K}_i}$ represents the approximation of the solution at the point $\boldsymbol{x}_i$, $i=1,\cdots,\mathcal{K}$.

In the present case, with the distribution of collocation points described in Section 3.2, system (4.15) has the special structure

$$A\boldsymbol{u} = \begin{pmatrix} A_{1,1} & A_{1,2} & \cdots & A_{1,N}\\ A_{2,1} & A_{2,2} & \cdots & A_{2,N}\\ \vdots & \vdots & \ddots & \vdots\\ A_{N,1} & A_{N,2} & \cdots & A_{N,N}\end{pmatrix}\begin{pmatrix}\boldsymbol{u}_1\\ \boldsymbol{u}_2\\ \vdots\\ \boldsymbol{u}_N\end{pmatrix} = \begin{pmatrix}\boldsymbol{b}_1\\ \boldsymbol{b}_2\\ \vdots\\ \boldsymbol{b}_N\end{pmatrix} = \boldsymbol{b}, \tag{4.16}$$

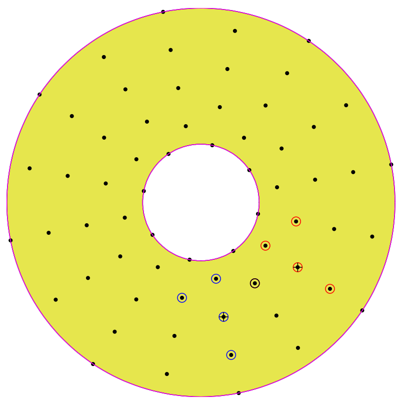


Figure 2: Typical distribution of collocation points and influence domains. The nearest four points to two centres (+) are highlighted in circles.

where the $M\times M$ submatrices A_{n_1,n_2}, $n_1,n_2=1,\cdots,N$, are *circulant* [10] and *sparse*. Hence matrix A in system (4.16) is *block circulant* consisting of sparse blocks. A brief explanation as to why each $M\times M$ submatrix A_{n_1,n_2} in (4.16) is circulant is as follows. On each concentric circle ℓ, say (where $\ell=1,\cdots,N$), we have M centres. For each of these centres, the local system (4.3) or more specifically (4.7) is the same. This means that for each centre $\boldsymbol{x}_i$, $i=1,\cdots,M$, on a concentric circle ℓ, the sets of solutions (coefficients) $\boldsymbol{w}_\Delta^i$ are the same. Thus the coefficients relating each centre $\boldsymbol{x}_i$ with its κ neighbouring points (some of which belong to circles other than ℓ) are the same for each centre on the circle ℓ. When this relation is "transferred" to the global matrix, the matrix resulting from each of the M points on the circle ℓ to the M points on any circle $\ell=1,\cdots,N$, will be circulant. This is because the relation between a point $\boldsymbol{x}_i$ and the non-zero-coefficient elements on circle ℓ', say, will be circulant. Alternatively, for each point $\boldsymbol{x}_i$ on the circle ℓ the sets of neighbouring points are globally circulant. This can be observed visually in Fig. 2 where we present a typical distribution of collocation points in black dots. Two consecutive centres are denoted by a cross (+) and their four nearest points, are denoted by blue and red circles. The point with a black circle belongs to both sets. A similar argument is valid for boundary points where Neumann conditions are imposed while the corresponding Dirichlet case is trivial.

Thus system (4.16) can be solved efficiently using the MDA proposed in [25] with some modifications exploiting the sparsity of the matrices A_{n_1,n_2}, $n_1,n_2=1,\cdots,N$.

4.2 Matrix decomposition algorithm

Following [25], if U is the unitary $M\times M$ Fourier matrix (see, e.g. [10]) and I_N is the $N\times N$ identity matrix, pre–multiplication of system (4.16) by $I_N\otimes U_M$ yields

$$(I_N\otimes U_M)A(I_N\otimes U_M^*)(I_N\otimes U_M)\boldsymbol{u}=\tilde{A}\tilde{\boldsymbol{u}}=(I_N\otimes U_M)\boldsymbol{b}=\tilde{\boldsymbol{b}}, \tag{4.17}$$

where

$$\tilde{A} = (I_N \otimes U_M)\,A\,(I_N \otimes U_M^*) = \begin{pmatrix} D_{1,1} & D_{1,2} & \cdots & D_{1,N} \\ D_{2,1} & D_{2,2} & \cdots & D_{2,N} \\ \vdots & \vdots & & \vdots \\ D_{N,1} & D_{N,2} & \cdots & D_{N,N} \end{pmatrix}, \tag{4.18}$$

and

$$\tilde{\boldsymbol{u}} = (I_N \otimes U_M)\boldsymbol{u} = \begin{pmatrix} \tilde{\boldsymbol{u}}_1 \\ \tilde{\boldsymbol{u}}_2 \\ \vdots \\ \tilde{\boldsymbol{u}}_N \end{pmatrix}, \quad \tilde{\boldsymbol{f}} = (I_N \otimes U_M)\boldsymbol{b} = \begin{pmatrix} \tilde{\boldsymbol{b}}_1 \\ \tilde{\boldsymbol{b}}_2 \\ \vdots \\ \tilde{\boldsymbol{b}}_N \end{pmatrix}. \tag{4.19}$$

From the properties of circulant matrices [10], each of the $M \times M$ matrices D_{n_1,n_2}, $n_1,n_2 = 1,\cdots,N$, is diagonal and if

$$D_{n_1,n_2} = \text{diag}\left(D_{n_1,n_{21}}, D_{n_1,n_{22}}, \cdots, D_{n_1,n_{2M}}\right) \quad \text{and} \quad A_{n_1,n_2} = \text{circ}\left(A_{n_1,n_{21}}, A_{n_1,n_{22}}, \cdots, A_{n_1,n_{2M}}\right), \tag{4.20}$$

we have, for $n_1,n_2 = 1,\cdots,N$,

$$D_{n_1,n_{2m}} = \sum_{k=1}^{M} A_{n_1,n_{2k}} \omega^{(k-1)(m-1)}, \quad m = 1,\cdots,M. \tag{4.21}$$

For convenience, we define the vectors describing the matrices D_{n_1,n_2} and A_{n_1,n_2} in (4.20) by

$$\boldsymbol{d}^{n_1,n_2} = \left[D_{n_1,n_{21}}, D_{n_1,n_{22}}, \cdots, D_{n_1,n_{2M}}\right]^T, \quad \boldsymbol{a}^{n_1,n_2} = \left[A_{n_1,n_{21}}, A_{n_1,n_{22}}, \cdots, A_{n_1,n_{2M}}\right]^T. \tag{4.22}$$

Since the matrix $\tilde{A}$ consists of N^2 blocks of order M, each of which is diagonal, system (4.17) can be decomposed into the M independent systems of order N

$$E_m \boldsymbol{x}_m = \boldsymbol{y}_m, \quad m = 1,\cdots,M, \tag{4.23}$$

where

$$(E_m)_{n_1,n_2} = D_{n_1,n_{2m}}, \quad n_1,n_2 = 1,\cdots,N,$$

and

$$(\boldsymbol{x}_m)_n = (\tilde{\boldsymbol{u}}_n)_m, \quad (\boldsymbol{y}_m)_n = \left(\tilde{\boldsymbol{b}}_n\right)_m, \quad n = 1,\cdots,N. \tag{4.24}$$

Having obtained the vectors $\boldsymbol{x}_m, m = 1,\cdots,M$, we can recover the vectors $\tilde{\boldsymbol{u}}_n, n = 1,\cdots,N$ and, subsequently, the vector $\boldsymbol{u}$ from (4.19), i.e.

$$\boldsymbol{u} = \begin{pmatrix} \boldsymbol{u}_1 \\ \boldsymbol{u}_2 \\ \vdots \\ \boldsymbol{u}_N \end{pmatrix} = (I_N \otimes U_M^*)\,\tilde{\boldsymbol{u}} = \begin{pmatrix} U_M^* \tilde{\boldsymbol{u}}_1 \\ U_M^* \tilde{\boldsymbol{u}}_2 \\ \vdots \\ U_M^* \tilde{\boldsymbol{u}}_N \end{pmatrix}. \tag{4.25}$$

The MDA for Poisson problems can be summarized as follows:

```
Algorithm 1
Step 1: Compute $\tilde{\boldsymbol{b}}_n = U_M \boldsymbol{b}_n$,   $n=1,\cdots,N$.
Step 2: Construct the diagonal matrices $D_{n_1,n_2}$ from (4.21).
Step 3: Solve the $M$, $N\times N$ systems (4.23) to obtain the $\{\boldsymbol{x}_m\}_{m=1}^M$,
        and subsequently the $\{\tilde{\boldsymbol{u}}_m\}_{m=1}^N$ from (4.24).
Step 4: Recover the vector of coefficients $\boldsymbol{u}$ from (4.25).
```

In Steps 1, 2 and 4, FFTs are used while the most expensive part of the algorithm is the solution of M linear systems, each of order N in Step 3. The FFTs are carried out using the MATLAB© [26] commands `fft` and `ifft`. Because of the sparsity of the matrix A in (4.16) several of the vectors $\boldsymbol{a}^{n_1,n_2}$ in (4.22) are zero vectors. Thus, by using an appropriate `if` statement we can avoid calculating the FFTs of zero vectors in Step 2 (i.e. (4.21)), which leads to substantial savings. In particular, we may check whether we have a zero vector by using the MATLAB© `nnz` command which determines the number of non-zero elements in the array. If the number of non-zero elements in the array is zero we do not perform the FFT in Step 2 and take the corresponding resulting vector $\boldsymbol{d}^{n_1,n_2}$ in (4.22) to be zero. Moreover, since each of the matrices E_m is therefore sparse, the solution of the systems in Step 3 can thus be solved using the MATLAB© `sparse` command leading to further substantial savings in the computational cost. In addition to the savings in computational cost, considerable savings in storage are achieved since only one row of the sparse circulant matrices involved needs to be stored.

5 Biharmonic problems

We define the collocation points in the same way as described in Section 3.2, see (3.2)-(3.4). However, the interior points $(\boldsymbol{x}_i)_{i=1}^{\mathcal{K}_{\rm int}}$ are defined from

$$\boldsymbol{x}_{(n-3)M+m} = (x_{mn}, y_{mn}), \quad m=1,\cdots,M, \quad n=3,\cdots,N-2, \tag{5.1}$$

while the boundary points $(\boldsymbol{x}_i)_{i=\mathcal{K}_{\rm int}+1}^{\mathcal{K}_{\rm int}+\mathcal{K}_{\rm bry}}$ from

$$\boldsymbol{x}_{(N-3)M+m} = (x_{m1}, y_{m1}) \quad \text{and} \quad \boldsymbol{x}_{(N-2)M+m} = (x_{mN}, y_{mN}), \quad m=1,\cdots,M. \tag{5.2}$$

The reason for taking fewer interior points than in the Poisson case is that in the biharmonic case we need to impose two boundary conditions on the boundary instead of one. Thus, $\mathcal{K}_{\rm int} = (N-4)M$, $\mathcal{K}_{\rm bry} = 2M$ and $\mathcal{K} = NM$. The points on $\partial\Omega_1$ and $\partial\Omega_2$ are defined as in Section 3.2.

As in Section 4.1 we use the local approximation (4.1). In the application of the LRBF method for the solution of boundary value problem (2.3), we satisfy the biharmonic equation (2.3a) at each interior point of Ω in $\mathcal{X}_i$, that is

$$\Delta^2 u_{\mathcal{K}}(\boldsymbol{x}) = f(\boldsymbol{x}), \quad \boldsymbol{x} \in \mathcal{X}_i. \tag{5.3}$$

The local formulation of (5.3) is similar to (4.7)-(4.9) by replacing Δ by Δ^2.

Regarding the Dirichlet and Neumann boundary conditions in the first biharmonic problem, we impose these as in the Poisson case, namely equations (4.14) and (4.13), respectively, at each of the N_{bry} boundary points. In the second biharmonic problem, we apply the Dirichlet boundary condition as in (4.14). For the Laplacian boundary condition, the local formulation is practically the same as (4.7)-(4.9). Note that in the second biharmonic problem, the problem may be decoupled into two Poisson problems which could be subsequently solved by the method described in Section 4.1, see [24].

In both biharmonic problems, we have a total of $\mathcal{K}=\mathcal{K}_{\text{int}}+2\mathcal{K}_{\text{bry}}$ (taking into account the two conditions on the boundary) equations in $\mathcal{K}$ unknowns, leading to a system of the form (4.15) where the matrix A is block circulant and sparse. With reference to system (4.15), the right hand side vector $\boldsymbol{b}$ is now defined by

$$\begin{aligned}
&b_i=f(\boldsymbol{x}_i), \quad i=1,\cdots,\mathcal{K}_{\text{int}},\\
&b_i=g_{_D}(\boldsymbol{x}_i), \quad i=\mathcal{K}_{\text{int}}+1,\cdots,\mathcal{K}_{\text{int}}+\mathcal{K}_{\text{bry}},\\
&b_{\mathcal{K}_{\text{bry}}+i}=g_{_N}(\boldsymbol{x}_i) \quad \text{or} \quad g_{_L}(\boldsymbol{x}_i), \quad i=\mathcal{K}_{\text{int}}+1,\cdots,\mathcal{K}_{\text{int}}+\mathcal{K}_{\text{bry}}.
\end{aligned}$$

Clearly, the resulting system can be solved efficiently using the MDA presented in Section 4.2.

6 Linear elasticity problems

We define the collocation points in the same way as described in Section 3.2, see (3.2)-(3.4), and $\mathcal{K}_{\text{int}}$, $\mathcal{K}_{\text{bry}}$, $\mathcal{K}_{\text{bry}_1}$, $\mathcal{K}_{\text{bry}_2}$ and $\mathcal{K}$ are also defined as in Section 3.2.

In this case we have two unknown functions, namely the displacements (u_1,u_2), which we approximate locally, for $k=1,2$, as in (4.1), by

$$u_{k_{\mathcal{K}}}(\boldsymbol{x})=\sum_{\ell=1}^{\kappa}\alpha^i_{k_\ell}\Phi(||\boldsymbol{x}-\boldsymbol{x}^i_\ell||), \quad \boldsymbol{x}\in\mathcal{X}_i, \qquad \text{or c.f. (4.4)}, \; u_{k_{\mathcal{K}}}(\boldsymbol{x})=\left(\boldsymbol{w}^i\right)^T\boldsymbol{u}^i_{k_{\mathcal{K}}}, \tag{6.1}$$

where $\boldsymbol{w}^i$ is defined as in (4.5) and

$$\boldsymbol{u}^i_{k_{\mathcal{K}}}=\left[u_{k_{\mathcal{K}}}(\boldsymbol{x}_{1(i)}),u_{k_{\mathcal{K}}}(\boldsymbol{x}_{2(i)}),\cdots,u_{k_{\mathcal{K}}}(\boldsymbol{x}_{n(i)})\right]^T.$$

We satisfy the Cauchy-Navier equations (2.4a) at each interior point of Ω in $\mathcal{X}_i$, that is, for $k=1,2$,

$$\mathcal{L}_k(u_{1_{\mathcal{K}}}(\boldsymbol{x}),u_{2_{\mathcal{K}}}(\boldsymbol{x}))=f_k(\boldsymbol{x}), \; \boldsymbol{x}\in\mathcal{X}_i. \tag{6.2}$$

By collocating the local approximation of all points $\boldsymbol{x}_i \in \mathcal{X}_i$, we have

$$\begin{aligned}\mathcal{L}_k(u_{1_\mathcal{K}}(\boldsymbol{x}_i),u_{2_\mathcal{K}}(\boldsymbol{x}_i)) &= \sum_{\ell=1}^{\kappa}\alpha^i_{1_\ell}\mathcal{L}^1_k\Phi(||\boldsymbol{x}_i-\boldsymbol{x}^i_\ell||)+\sum_{\ell=1}^{\kappa}\alpha^i_{2_\ell}\mathcal{L}^2_k\Phi(||\boldsymbol{x}_i-\boldsymbol{x}^i_\ell||)\\ &=(\boldsymbol{h}^i_{\mathcal{L}^1_k})^T B_i^{-1}\boldsymbol{u}^i_{1_\mathcal{K}}+(\boldsymbol{h}^i_{\mathcal{L}^2_k})^T B_i^{-1}\boldsymbol{u}^i_{2_\mathcal{K}}\\ &=\left(\boldsymbol{w}^i_{\mathcal{L}^1_k}\right)^T\boldsymbol{u}^i_{1_\mathcal{K}}+\left(\boldsymbol{w}^i_{\mathcal{L}^2_k}\right)^T\boldsymbol{u}^i_{2_\mathcal{K}}\\ &=f_k(\boldsymbol{x}_i),\quad i=1,\cdots,\mathcal{K}_{\text{int}},\end{aligned}\tag{6.3}$$

where

$$\boldsymbol{h}^i_{\mathcal{L}^{k_2}_{k_1}}=\left[\mathcal{L}^{k_2}_{k_1}\Phi(||\boldsymbol{x}_i-\boldsymbol{x}^i_1||),\mathcal{L}^{k_2}_{k_1}\Phi(||\boldsymbol{x}_i-\boldsymbol{x}^i_2||),\cdots,\mathcal{L}^{k_2}_{k_1}\Phi(||\boldsymbol{x}_i-\boldsymbol{x}^i_\kappa||)\right]^T,\quad k_1,k_2=1,2,$$

and

$$\boldsymbol{w}^i_{\mathcal{L}^{k_2}_{k_1}}=B_i^{-1}\boldsymbol{h}^i_{\mathcal{L}^{k_2}_{k_1}},\quad k_1,k_2=1,2.$$

The Dirichlet boundary conditions in (2.4b) and (2.4c) are applied separately for u_1 and u_2 as in (4.14). The Neumann boundary conditions in (2.4c) are applied as follows

$$\begin{aligned}t_{1_\mathcal{K}}(\boldsymbol{x}_i)=&\sum_{\ell=1}^{\kappa}\alpha^i_{1_\ell}\left[2\mu\left(\frac{1-\nu}{1-2\nu}\right)\frac{\partial\Phi(||\boldsymbol{x}_i-\boldsymbol{x}^i_\ell||)}{\partial x}\mathrm{n}_x(\boldsymbol{x}_i)+\mu\frac{\partial\Phi(||\boldsymbol{x}_i-\boldsymbol{x}^i_\ell||)}{\partial y}\mathrm{n}_y(\boldsymbol{x}_i)\right]\\ &+\sum_{\ell=1}^{\kappa}\alpha^i_{2_\ell}\left[2\mu\left(\frac{\nu}{1-2\nu}\right)\frac{\partial\Phi(||\boldsymbol{x}_i-\boldsymbol{x}^i_\ell||)}{\partial y}\mathrm{n}_x(\boldsymbol{x}_i)+\mu\frac{\partial\Phi(||\boldsymbol{x}_i-\boldsymbol{x}^i_\ell||)}{\partial x}\mathrm{n}_y(\boldsymbol{x}_i)\right]\\ =&(\boldsymbol{h}^i_{t^1_1})^T B_i^{-1}\boldsymbol{u}^i_{1_\mathcal{K}}+(\boldsymbol{h}^i_{t^2_1})^T B_i^{-1}\boldsymbol{u}^i_{2_\mathcal{K}}\\ =&\left(\boldsymbol{w}^i_{t^1_1}\right)^T\boldsymbol{u}^i_{1_\mathcal{K}}+\left(\boldsymbol{w}^i_{t^2_1}\right)^T\boldsymbol{u}^i_{2_\mathcal{K}}\\ =&g_{N_1}(\boldsymbol{x}_i),\end{aligned}$$

and

$$\begin{aligned}t_{2_\mathcal{K}}(\boldsymbol{x}_i)=&\sum_{\ell_1}^{\kappa}\alpha^i_{1_\ell}\left[\mu\frac{\partial\Phi(||\boldsymbol{x}_i-\boldsymbol{x}^i_\ell||)}{\partial y}\mathrm{n}_x(\boldsymbol{x}_i)+2\mu\left(\frac{\nu}{1-2\nu}\right)\frac{\partial\Phi(||\boldsymbol{x}_i-\boldsymbol{x}^i_\ell||)}{\partial x}\mathrm{n}_y(\boldsymbol{x}_i)\right]\\ &+\sum_{\ell=1}^{\kappa}\alpha^i_{2_\ell}\left[\mu\frac{\partial\Phi(||\boldsymbol{x}_i-\boldsymbol{x}^i_\ell||)}{\partial x}\mathrm{n}_x(\boldsymbol{x}_i)+2\mu\left(\frac{1-\nu}{1-2\nu}\right)\frac{\partial\Phi(||\boldsymbol{x}_i-\boldsymbol{x}^i_\ell||)}{\partial y}\mathrm{n}_y(\boldsymbol{x}_i)\right]\\ =&(\boldsymbol{h}^i_{t^1_2})^T B_i^{-1}\boldsymbol{u}^i_{1_\mathcal{K}}+(\boldsymbol{h}^i_{t^2_2})^T B_i^{-1}\boldsymbol{u}^i_{2_\mathcal{K}}\\ =&\left(\boldsymbol{w}^i_{t^1_2}\right)^T\boldsymbol{u}^i_{1_\mathcal{K}}+\left(\boldsymbol{w}^i_{t^2_2}\right)^T\boldsymbol{u}^i_{2_\mathcal{K}}\\ =&g_{\mathcal{K}_2}(\boldsymbol{x}_i),\quad i=\mathcal{K}_{\text{int}}+\mathcal{K}_{\text{bry}_1}+1,\cdots,\mathcal{K}_{\text{int}}+\mathcal{K},\end{aligned}$$

where

$$\boldsymbol{h}^i_{t^1_1}=2\mu\left(\frac{1-\nu}{1-2\nu}\right)\mathrm{n}_x(\boldsymbol{x}_i)\boldsymbol{h}^i_{\Phi_x}+\mu\mathrm{n}_y(\boldsymbol{x}_i)\boldsymbol{h}^i_{\Phi_y},\quad \boldsymbol{h}^i_{t^2_1}=2\mu\left(\frac{\nu}{1-2\nu}\right)\mathrm{n}_x(\boldsymbol{x}_i)\boldsymbol{h}^i_{\Phi_y}+\mu\mathrm{n}_y(\boldsymbol{x}_i)\boldsymbol{h}^i_{\Phi_x},$$
$$\boldsymbol{h}^i_{t^1_2}=\mu\mathrm{n}_x(\boldsymbol{x}_i)\boldsymbol{h}^i_{\Phi_y}+2\mu\left(\frac{\nu}{1-2\nu}\right)\mathrm{n}_y(\boldsymbol{x}_i)\boldsymbol{h}^i_{\Phi_x},\quad \boldsymbol{h}^i_{t^2_2}=\mu\mathrm{n}_x(\boldsymbol{x}_i)\boldsymbol{h}^i_{\Phi_x}+2\mu\left(\frac{1-\nu}{1-2\nu}\right)\mathrm{n}_y(\boldsymbol{x}_i)\boldsymbol{h}^i_{\Phi_y},$$

$\boldsymbol{h}^i_{\Phi_x}$ and $\boldsymbol{h}^i_{\Phi_y}$ are defined as in (4.12), and $\boldsymbol{w}^i_{t^{k_2}_{k_1}}=B_i^{-1}\boldsymbol{h}^i_{t^{k_2}_{k_1}}$, $k_1,k_2=1,2$.

Assembling all equations for the $\mathcal{K}$ centres $\boldsymbol{x}_i$, we obtain a system of the form (4.15) where, now, the matrix $A\in\mathbb{R}^{2\mathcal{K}\times 2\mathcal{K}}$ is sparse but not block circulant. With reference to system (4.15) the right hand side $\boldsymbol{b}$ is defined by

$$\begin{aligned}
&b_i=f_1(\boldsymbol{x}_i),\quad b_{i+1}=f_2(\boldsymbol{x}_i),\quad i=1,\cdots,\mathcal{K}_{\text{int}},\\
&b_i=g_{D_1}(\boldsymbol{x}_i)\ \ (\text{or }\ b_i=g_{N_1}(\boldsymbol{x}_i))\quad\text{and}\quad b_{i+1}=g_{D_2}(\boldsymbol{x}_i)\ \ (\text{or }\ b_{i+1}=g_{N_2}(\boldsymbol{x}_i)),\\
&\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad i=\mathcal{K}_{\text{int}}+1,\cdots,\mathcal{K}_{\text{int}}+\mathcal{K}_{\text{bry}_1},\\
&b_i=g_{D_1}(\boldsymbol{x}_i)\,\text{and}\,b_{i+1}=g_{D_2}(\boldsymbol{x}_i),\quad i=\mathcal{K}_{\text{int}}+\mathcal{K}_{\text{bry}_1}+1,\cdots,\mathcal{K}.
\end{aligned}$$

The solution vector $\boldsymbol{u}_{\mathcal{K}}\in\mathbb{R}^{2\mathcal{K}\times 1}$ yields the approximations to the solution (u_1,u_2) at the centres. As described in [25], by an appropriate transformation, the sparse, non-block circulant matrix can be transformed into a sparse block circulant matrix and the system solved efficiently by an appropriate MDA. This process is described next.

6.1 Matrix decomposition algorithm

As in [25] we define the $2M\times 2M$ matrix

$$R=\left(\begin{array}{c|c|c|c|c|c}
R_{\vartheta_1} & 0 & 0 & \cdots & 0 & 0\\ \hline
0 & R_{\vartheta_2} & 0 & \cdots & 0 & 0\\ \hline
\vdots & \vdots & \ddots & \vdots & \vdots & \vdots\\ \hline
0 & 0 & 0 & \cdots & R_{\vartheta_{M-1}} & 0\\ \hline
0 & 0 & 0 & \cdots & 0 & R_{\vartheta_M}
\end{array}\right),\tag{6.4}$$

where

$$R_{\vartheta_k}=\begin{pmatrix}\cos\vartheta_k & \sin\vartheta_k\\ \sin\vartheta_k & -\cos\vartheta_k\end{pmatrix},\quad \vartheta_k=\frac{2\pi(k-1)}{M},\quad \text{clearly satisfies } R^2_{\vartheta_k}=I_2 \text{ and hence } R^2=I_{2N}.$$

We premultiply the $2MN\times 2MN$ Cauchy-Navier system (4.15) by the $2MN\times 2MN$ matrix $I_N\otimes R$ to get

$$(I_N\otimes R)A\boldsymbol{u}=\tilde{A}\tilde{\boldsymbol{u}}=(I_N\otimes R)\boldsymbol{b}=\tilde{\boldsymbol{b}},\quad\text{where}\quad \tilde{\boldsymbol{u}}=(I_N\otimes R)\boldsymbol{u},\tag{6.5}$$

and

$$\tilde{A} = (I_N \otimes R) A (I_N \otimes R) = \begin{pmatrix} \tilde{A}_{1,1} & \tilde{A}_{1,2} & \cdots & \tilde{A}_{1,N} \\ \tilde{A}_{2,1} & \tilde{A}_{2,2} & \cdots & \tilde{A}_{2,N} \\ \vdots & \vdots & \ddots & \vdots \\ \tilde{A}_{N,1} & \tilde{A}_{N,2} & \cdots & \tilde{A}_{N,N} \end{pmatrix}, \tag{6.6}$$

with $\tilde{A}_{m,\ell} = R A_{m,\ell} R$, $m,\ell = 1,\cdots M$.

The elements $\big(\tilde{A}_{n_1,n_2}\big)_{m_1,m_2} = \big(\big(\tilde{A}_{n_1,n_2}\big)_{m_1,m_2}\big)_{i,j=1}^{2}$ (which are 2×2 arrays), are given by

$$\big(\tilde{A}_{n_1,n_2}\big)_{m_1,m_2} = R_{\vartheta_{m_1}} \big(A_{n_1,n_2}\big)_{m_1,m_2} R_{\vartheta_{m_2}}, \quad m_1,m_2 = 1,\cdots,M, \quad n_1,n_2 = 1,\cdots,N, \tag{6.7}$$

and each submatrix $\tilde{A}_{n_1,n_2}, n_1,n_2 = 1,\cdots,N$, is sparse and has a block 2×2 block circulant structure.

System (6.5) may be written in the form

$$\left(\begin{array}{c|c} B_{11} & B_{12} \\ \hline B_{21} & B_{22} \end{array}\right) \left(\begin{array}{c} \boldsymbol{c}_1 \\ \hline \boldsymbol{c}_2 \end{array}\right) = \left(\begin{array}{c} \boldsymbol{d}_1 \\ \hline \boldsymbol{d}_2 \end{array}\right), \quad \text{where } B_{ij} = \begin{pmatrix} \tilde{B}^{ij}_{1,1} & \tilde{B}^{ij}_{1,2} & \cdots & \tilde{B}^{ij}_{1,N} \\ \tilde{B}^{ij}_{2,1} & \tilde{B}^{ij}_{2,2} & \cdots & \tilde{B}^{ij}_{2,N} \\ \vdots & \vdots & \ddots & \vdots \\ \tilde{B}^{ij}_{N,1} & \tilde{B}^{ij}_{N,2} & \cdots & \tilde{B}^{ij}_{N,N} \end{pmatrix}, \; i,j = 1,2, \tag{6.8}$$

and each $M\times M$ submatrix $\tilde{B}^{ij}_{n_1,n_2}$, $i,j = 1,2$, $n_1,n_2 = 1,\cdots,N$, is circulant and defined by

$$\Big(\tilde{B}^{ij}_{n_1,n_2}\Big)_{m_1,m_2} = \Big(\big(\tilde{A}_{n_1,n_2}\big)_{m_1,m_2}\Big)_{i,j}, \quad m_1,m_2 = 1,\cdots,M. \tag{6.9}$$

We premultiply system (6.8) by the matrix $I_2 \otimes I_N \otimes U_M$ to get

$$\begin{aligned} (I_2 \otimes I_N \otimes U_M) \left(\begin{array}{c|c} B_{11} & B_{12} \\ \hline B_{21} & B_{22} \end{array}\right) \left(\begin{array}{c} \boldsymbol{c}_1 \\ \hline \boldsymbol{c}_2 \end{array}\right) &= \left(\begin{array}{c|c} \tilde{B}_{11} & \tilde{B}_{12} \\ \hline \tilde{B}_{21} & \tilde{B}_{22} \end{array}\right) \left(\begin{array}{c} \boldsymbol{p}_1 \\ \hline \boldsymbol{p}_2 \end{array}\right) \\ &= (I_2 \otimes I_N \otimes U_M) \left(\begin{array}{c} \boldsymbol{d}_1 \\ \hline \boldsymbol{d}_2 \end{array}\right) = \left(\begin{array}{c} \boldsymbol{q}_1 \\ \hline \boldsymbol{q}_2 \end{array}\right), \end{aligned} \tag{6.10}$$

where

$$\left(\begin{array}{c} \boldsymbol{p}_1 \\ \hline \boldsymbol{p}_2 \end{array}\right) = (I_2 \otimes I_N \otimes U_M) \left(\begin{array}{c} \boldsymbol{c}_1 \\ \hline \boldsymbol{c}_2 \end{array}\right) \quad \text{and} \quad \tilde{B}_{ij} = (I_N \otimes U_M) B_{ij} (I_N \otimes U_M^*), \quad i,j = 1,2. \tag{6.11}$$

Each of the matrices $B_{ij}, i,j = 1,2$ is block circulant, and from (4.18) it follows that

$$\tilde{B}_{ij} = \begin{pmatrix} D^{ij}_{1,1} & D^{ij}_{1,2} & \cdots & D^{ij}_{1,N} \\ D_{2,1} & D^{ij}_{2,2} & \cdots & D^{ij}_{2,N} \\ \vdots & \vdots & & \vdots \\ D^{ij}_{N,1} & D^{ij}_{N,2} & \cdots & D^{ij}_{N,N} \end{pmatrix}, \tag{6.12}$$

where each $M\times M$ matrix $D^{ij}_{n_1,n_2}$, $n_1,n_2=1,\cdots,N$, is diagonal.

If

$$D^{ij}_{n_1,n_2}=\text{diag}\left(D^{ij}_{n_1,n_{21}},D^{ij}_{n_1,n_{22}},\cdots,D^{ij}_{n_1,n_{2M}}\right) \quad \text{and} \quad \tilde{B}^{ij}_{n_1,n_2}=\text{circ}\left(\tilde{B}^{ij}_{n_1,n_{21}},\tilde{B}^{ij}_{n_1,n_{2M}},\cdots,\tilde{B}^{ij}_{n_1,n_{2M}}\right),$$

we have, for $n_1,n_2=1,\cdots,N$,

$$D^{ij}_{n_1,n_{2m}}=\sum_{k=1}^{M}\tilde{B}^{ij}_{n_1,n_{2k}}\omega^{(k-1)(m-1)}, \quad m=1,\cdots,M. \tag{6.13}$$

Since each matrix $\tilde{B}_{ij}$, $i,j=1,2$, consists of N^2 diagonal blocks of order M, system (6.10) is equivalent to the M systems of order $2N$

$$\left(\begin{array}{c|c} E^m_{11} & E^m_{12} \\ \hline E^m_{21} & E^m_{22}\end{array}\right)\left(\begin{array}{c} \boldsymbol{x}^m_1 \\ \hline \boldsymbol{x}^m_2\end{array}\right)=\left(\begin{array}{c} \boldsymbol{y}^m_1 \\ \hline \boldsymbol{y}^m_2\end{array}\right), \quad m=1,\cdots,M, \tag{6.14}$$

where

$$\left(E^m_{ij}\right)_{n_1,n_2}=D^{ij}_{n_1,n_{2m}}, \quad n_1,n_2=1,\cdots,N.$$

From the vectors $\boldsymbol{x}^m_i$, $i=1,2$, $m=1,\cdots,M$, we can obtain the vectors $\boldsymbol{p}_1,\boldsymbol{p}_2$ and the vectors $\boldsymbol{c}_1,\boldsymbol{c}_2$, and subsequently the vector $\tilde{\boldsymbol{u}}$, before finally obtaining the vector $\boldsymbol{u}$.

The MDA for linear elasticity problems can be summarized as follows:

Algorithm 2

```
Step 1: Compute  $\tilde{\boldsymbol{b}}=(I_N\otimes R)\boldsymbol{b}$.
Step 2: Calculate  the 2×2 arrays $\left(\tilde{A}_{n_1,n_2}\right)_{1,m_2}$ from (6.7).
Step 3: Compute $\boldsymbol{q}_1,\boldsymbol{q}_2$ in (6.10) and hence $\boldsymbol{y}^m_i$, $m=1,\cdots,N$, $i=1,2$ in (6.14).
Step 4: Construct the diagonal matrices $D^{ij}_{n_1,n_2}$ from (6.13) and matrices $E^m_{ij}$ in
        (6.14).
Step 5: Solve the $M$, $2N\times 2N$ systems (6.14) to obtain the $\boldsymbol{x}^m_i$, $i=1,2$,
        $m=1,\cdots,M$, and subsequently the vectors $\boldsymbol{p}_i$, $i=1,2$.
Step 6: Recover the vectors $\boldsymbol{c}_i$, $i=1,2$ from (6.11).
Step 7: Reorder vectors $\boldsymbol{c}_i$, $i=1,2$ to obtain vector $\tilde{\boldsymbol{u}}$.
Step 8: Compute  $\boldsymbol{u}=(I_N\otimes R)\tilde{\boldsymbol{u}}$.
```

In Steps 3, 4 and 6 FFTs are used while the most expensive part of the algorithm is the solution of M linear systems, each of order $2N$ in Step 5. Because of the sparsity of the matrices B_{ij}, $i,j=1,2$, in (6.8), the corresponding matrices $\tilde{B}_{ij}$, $i,j=1,2$, are also sparse and several of the vectors $[\tilde{B}^{ij}_{n_1,n_{21}},\tilde{B}^{ij}_{n_1,n_{22}},\cdots,\tilde{B}^{ij}_{n_1,n_{2M}}]^T$ in (6.13) are zero vectors. Thus, as in the Poisson case, the FFT calculations of these zero vectors in Step 4 can be avoided by appropriately using the `nnz` command. Also, as in the Poisson case, each of the matrices E^m_{ij} is sparse, hence the coefficient matrices in the systems of Step 5 are sparse and further savings can be achieved by using the `sparse` command in their solution. Again, substantial savings in storage are obtained as only the first line of the circulant matrices involved needs to be constructed and stored.

7 Numerical examples

In all numerical examples considered, we took the inner and outer radii of the annular domain Ω to be $\varrho_1 = 0.3$, $\varrho_2 = 1$, respectively, and, unless otherwise stated, we chose collocation points described by $\alpha_n = (-1)^n/4$, $n = 1,\cdots,N$ (cf. (3.4)).

The accuracy of the approximations was assessed by calculating the LRBF approximations at the interior centres and then the root mean square error E, defined by

$$E = \frac{||u - u_{\mathcal{K}}||_{2,\Omega}}{\sqrt{\mathcal{K}_{\text{int}}}}, \tag{7.1}$$

where $\mathcal{K}_{\text{int}}$ is the number of interior collocation points.

The numerical computations were carried out on a MATLAB© 2014 platform on a desktop PC with 8x Intel(R) Core(TM) i7-2600k CPU@3.40 GHZ, 16 GB memory, in Linux OS Buntu 14.04.1 LTS.

In the following examples, we took the RBF Φ in (4.1) to be the normalized MQ basis function

$$\Phi(r_j) = \sqrt{(cr_j)^2 + 1}, \quad r_j^2 = (x - x_j)^2 + (y - y_j)^2,$$

where c is the shape parameter.

7.1 Example 1

We consider the Dirichlet boundary value problem (2.2) for the Poisson equation corresponding to the exact solution $u = \sin(\pi x)\cos(\pi y/2)$.

First, we chose $M = N = 80$ and in Fig. 3 we present the error E versus the shape parameter c for the case $\kappa = 9$. From this figure, the observed optimal c and corresponding E are 0.920 and 3.703(-6) respectively. We then applied the LOOCV algorithm to search for

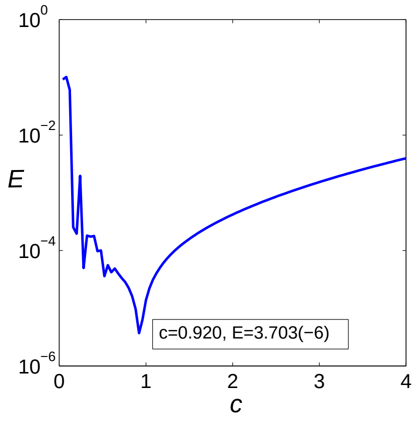


Figure 3: Example 1: Dirichlet case. Error versus c for the case $M = N = 80$, $\kappa = 9$.

Table 1: Example 1: Shape parameters and corresponding errors obtained using various LOOCV initial search intervals for the Dirichlet boundary value problem with $M=N=80$.

	$\kappa=9$		$\kappa=30$	
`[min, max]`	c	E	c	E
[0,3]	0.5838	5.254(−5)	1.4064	3.898(−6)
[0,4]	0.5827	4.986(−5)	2.3792	5.674(−6)
[0,5]	0.5838	4.827(−5)	2.4324	7.015(−6)
[0,6]	0.5823	5.699(−5)	2.5168	8.060(−6)
[0,7]	0.5809	4.107(−5)	2.1995	3.922(−6)
[0,8]	0.5845	5.198(−5)	2.4964	7.601(−6)

Table 2: Example 1: Accuracy and CPU times for a large number of interpolation points using $\kappa=9$ and initial search interval [0,5].

$M=N$	c	E	CPU (sec)
200	1.5800	3.000(−5)	0.51
300	2.1607	4.425(−5)	1.53
400	1.2450	8.928(−6)	3.16
500	1.3207	6.083(−6)	6.02
600	1.5903	8.297(−6)	10.08
700	1.9804	1.340(−5)	15.14
800	1.6084	4.837(−5)	25.05
900	1.5095	2.332(−5)	30.16
1000	2.1841	2.564(−5)	44.97

an appropriate shape parameter and in Table 1 we present the shape parameters obtained by using various lengths of initial search intervals `[min, max]` and their corresponding errors for the cases $\kappa=9$ and $\kappa=30$. We observe that the shape parameters obtained are consistent for different initial search intervals. The accuracy can be further improved by increasing the number of collocation points in the local influence domain to $\kappa=30$.

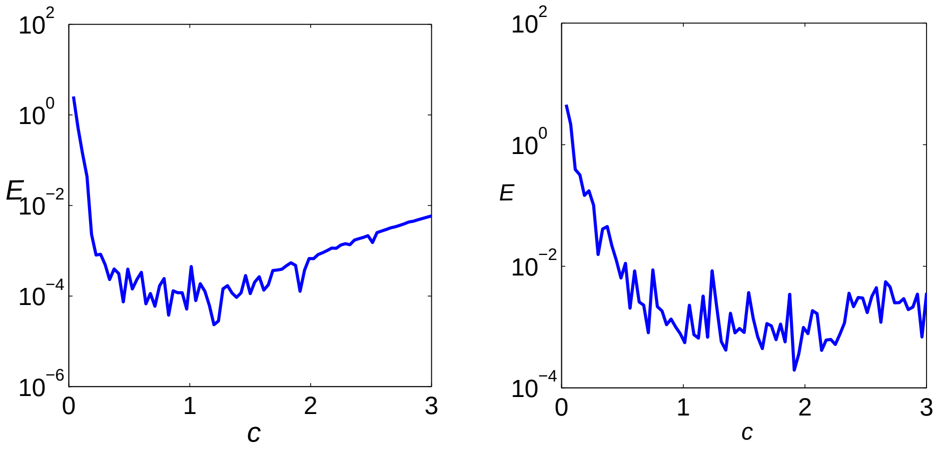
In Table 2 we present the results obtained using a large number of collocation points which is often required in the solution of large-scale problems. An indication of the efficiency of the proposed LRBF-MDA is that, in this example, it takes only 45 seconds of computer time to process a cloud of one million collocation points. Apparently, we can easily move beyond one million points. Note that after the accuracy reaches a certain level, it stabilizes with the increase of the collocation points. The reason that we implement such a large number of collocation points is to demonstrate that our algorithm is capable of handling problems with extremely large numbers of collocation points in a stable manner without problems of ill-conditioning, memory space, and computational time.

7.2 Example 2

In this example we consider the first biharmonic problem (2.3a)-(2.3b) corresponding to the exact solution $u(x,y)=\mathrm{e}^{x+y}$. We chose collocation points described by $\alpha_n=(-1)^n/5$, $n=1,\cdots,N$ (cf. (3.4)).

In Fig. 4 we present the profile of the error versus the shape parameter c for $M=N=80,200$, $\kappa=30$. The shape parameters and corresponding errors obtained using LOOCV for various initial search intervals [`min,max`], in the cases $M=N=80$, $\kappa=9,30,50$, are given in Table 3. Despite of the fluctuation of the curve in Fig. 4(a), the estimation of a good shape parameter shown in Table 3 for $\kappa=30$ is quite stable for various initial search intervals. In the case $\kappa=9$, the solution is stable in terms of shape parameter, but the accuracy is poor. The situation is reversed for the case $\kappa=50$ where the obtained shape parameters are not stable but the accuracy is much improved. When the number of collocation points is increased ($M=N=200$), the fluctuation of the curve becomes more intense as shown in Fig. 4(b), and the corresponding values of the shape parameter obtained using LOOCV Table 4 are unstable for various initial search intervals. In addition, the accuracy does not necessarily improve when increasing the number of collocation points. The main difficulty in this problem is the determination of a suitable shape parameter.

The difficulties encountered when local methods are used to solve biharmonic problems are well-document in the literature [3,24]. Hence, the relatively low accuracy of the results obtained in this example is not unusual. Improved results can be obtained when the second biharmonic problem consisting of (2.3a) and (2.3c) is split into two Poisson Dirichlet problems and these are subsequently solved using Algorithm 1, described in Section 4.2. Further details regarding this approach may be found in [24].



(a) $M=N=80$, $\kappa=30$ (b) $M=N=200$, $\kappa=30$

Figure 4: Example 2: Biharmonic problem. Error versus c for the cases $M=N=80,200$, $\kappa=30$.

Table 3: Example 2: Shape parameters and corresponding errors obtained using various LOOCV initial search intervals for solving the biharmonic problem with $M=N=80$.

	$\kappa=9$		$\kappa=30$		$\kappa=50$	
[min, max]	c	E	c	E	c	E
[0,3]	0.7692	1.977(−1)	1.3392	4.807(−4)	1.4990	1.578(−4)
[0,4]	0.7670	1.840(−1)	2.1417	1.115(−3)	1.8621	3.465(−4)
[0,5]	0.7675	2.026(−1)	2.2020	1.411(−3)	2.4882	1.117(−3)
[0,6]	0.7682	2.019(−1)	2.2791	1.669(−3)	2.7767	2.916(−3)
[0,7]	0.7682	1.641(−1)	2.1556	1.259(−3)	3.0092	2.867(−3)
[0,8]	0.7664	2.079(−1)	2.1653	1.148(−3)	2.8660	3.230(−3)

Table 4: Example 2: Shape parameters and corresponding errors obtained using various LOOCV initial search intervals for solving the biharmonic problem with $M=N=200$.

	$\kappa=30$		$\kappa=40$	
[min, max]	c	E	c	E
[0,3]	1.797	1.027(−3)	0.9393	2.204(−4)
[0,4]	2.5047	7.763(−4)	2.6169	1.535(−4)
[0,5]	3.9463	1.599(−3)	1.9098	7.789(−3)
[0,6]	2.2918	1.446(−2)	3.6594	3.231(−3)
[0,7]	2.1335	4.284(−3)	2.0375	1.416(−2)
[0,8]	2.5234	2.092(−2)	3.6375	2.539(−2)

7.3 Example 3

In this example we first consider the Dirichlet Cauchy-Navier problem (2.4a)-(2.4b) corresponding to the exact solutions $u_1(x,y)=\mathrm{e}^{x+y}$, $u_2(x,y)=\cos(x+y)$.

In our numerical experiments, we first chose $M=N=80$, $\kappa=9$ and in Fig. 5 we show how the errors behave with respect to the shape parameter. As was the case in Example 1, the two error curves E_1 and E_2 (which are defined as in (7.1)) are relatively smooth and predictable which is ideal for the application of LOOCV. The values of the shape parameters obtained using LOOCV and the corresponding errors which are presented in Table 5, are in very good agreement with the optimal solutions in Fig. 5 for the same input data. As shown in Table 6, we obtain similar results for the Cauchy-Navier problem with mixed Neumann/Dirichlet boundary conditions, that is, the boundary value problem consisting of (2.4a) and (2.4c).

In Table 7 we present the results obtained for the Dirichlet Cauchy-Navier problem using a large number of collocation points. These indicate that as the number of degrees of freedom increases, the numerical solutions remain stable and accurate.

Another noteworthy observation from Table 7 is that for 360,000 collocation points, it takes only 71.09 seconds of CPU time to complete the task, which highlights the efficiency of the proposed algorithm.

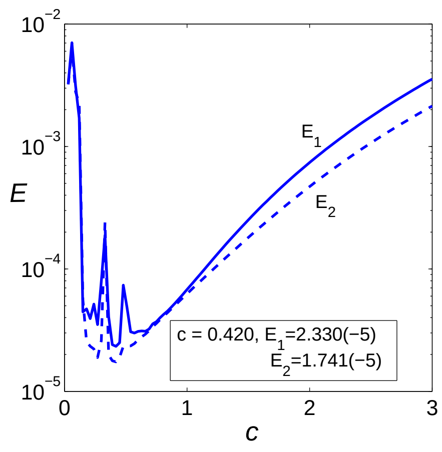


Figure 5: Example 3: Dirichlet Cauchy-Navier problem. Error versus c for the case $M=N=80$, $\kappa=9$.

Table 5: Example 3: Shape parameters and corresponding errors obtained using various LOOCV initial search intervals for solving the Dirichlet Cauchy-Navier problem with $M=N=80$, $\kappa=9$.

[min, max]	c	E_1	E_2
[0,3]	0.5751	2.854(−5)	2.485(−5)
[0,4]	0.5792	3.067(−5)	2.530(−5)
[0,5]	0.5744	2.746(−5)	2.440(−5)
[0,6]	0.5748	3.074(−5)	2.509(−5)
[0,7]	0.5781	2.625(−5)	2.418(−5)
[0,8]	0.5766	3.051(−5)	2.529(−5)

Table 6: Example 3: Shape parameters and corresponding errors obtained using various LOOCV initial search intervals for solving the mixed Neumann/Dirichlet Cauchy-Navier problem with $M=N=80$, $\kappa=9$.

[min, max]	c	E_1	E_2
[0,3]	0.5831	3.989(−5)	3.284(−5)
[0,4]	0.5814	4.193(−5)	3.231(−5)
[0,5]	0.5821	4.145(−5)	3.263(−5)
[0,6]	0.5807	4.367(−5)	3.203(−5)
[0,7]	0.5810	4.161(−5)	3.224(−5)
[0,8]	0.5793	4.094(−5)	3.244(−5)

We also performed two stress tests for the Dirichlet Cauchy-Navier problem by moving the inner boundary very close to the exterior boundary and by moving the inner boundary very close to the center of the concentric circles, respectively. First, we let the radii of the inner and outer circles be 0.97 and 1.0, respectively. The numerical solution of boundary value problems by traditional global methods in thin domains such as this, is

Table 7: Example 3: Accuracy and CPU times for a large number of interpolation points for the Dirichlet Cauchy-Navier problem using $\kappa=8$ and initial search interval [0,5].

$M=N$	c	E_1	E_2	CPU (sec)
200	0.6695	1.979(−5)	2.194(−5)	2.10
300	0.9754	1.570(−5)	2.742(−5)	5.53
400	1.3013	1.885(−5)	2.884(−5)	15.15
500	1.5227	4.244(−5)	2.694(−5)	26.18
600	1.7764	5.815(−5)	3.712(−5)	71.09

most challenging. With the proposed LRBF-MDA and LOOCV, however, with $M=1000$, $N=50$ and $\kappa=8$ we obtained high accuracy with errors $E_1=1.114(-7)$, $E_2=2.949(-7)$, $c=2.1105$. In the other extreme case, we reduced the radius of the inner circle to 0.01 and taking $M=N=500$, $\kappa=20$, we obtained $E_1=8.540(-5)$, $E_2=1.107(-4)$ for $c=2.4275$.

The dense concentration of collocation points in the thin annulus in the first case and on the circles close to the centre in the second, could create difficulties due to severe ill-conditioning. The proposed LRBF-MDA, however, appears to overcome this.

We also observe that the shape parameters obtained in Tables 5-7 are very similar to those reported in Example 1. This leads us to the conjecture that the shape parameter of second order elliptic equations is heavily dependent on the density and distribution of the collocation points.

8 Conclusions

The LRBF method has been applied to boundary value problems for the Poisson, inhomogeneous biharmonic and inhomogeneous Cauchy-Navier equations in annular domains. By appropriately choosing the collocation points, and in the case of the Cauchy-Navier equations by an appropriate transformation, the LRBF discretization leads to systems in which the coefficient matrices are sparse block circulant. These systems lend themselves for the application of MDAs which lead to substantial savings in computer time and storage. The LOOCV algorithm is used to select a shape parameter which yields accurate results and a modification of the kdtree algorithm is used to efficiently select the neighbouring points in the local influence domain.

In the proposed method we do not exploit one of the main advantages of the LRBF method which is its ability to tackle problems with irregularly scattered points [14] as well as regularly distributed points. However, by appropriately selecting the parameters α_n and the radii r_n in (3.4), we can obtain unlimited collocation point distribution patterns. In addition, we should state that it is not the purpose of this study to compare the proposed method with other methods but to propose an alternative efficient and elegant technique for the solution of the problems in question. It could well be the case that other methods perform better for some of the proposed distributions of collocation points.

Possible areas of future research could include the extension of the current LRBF algorithms for the solution of three-dimensional axisymmetric problems as was achieved in the case of the global Kansa-RBF method in [18]. The proposed method could also be adapted for the solution of problems in annular domain in which the types of boundary conditions alternate on the same segment of the boundary. This was examined in [17] using the method of fundamental solutions which is a boundary RBF method. The LRBF discretization in the proposed MDAs leads to several independent sparse systems which we solve using the MATLAB© `sparse` command. The performance of iterative solvers such as `bicgstab` for the solution of these sparse systems could be a subject of future investigation.

An outstanding issue is the efficient calculation of the FFTs of sparse vectors which occurs repeatedly in the application of the proposed algorithm. Clearly, it is not possible to use the `fft` (or the `ifft`) command with a sparse vector. One can, however, select the columns of the Fourier matrix which multiply the non-zero elements of the vector in question via the MATLAB© `bsxfun` command and then carry out the multiplications with just the non-zero elements of the vector via the command `nonzeros`. Extensive experimentation has revealed that the `fft` command for the full vector is still considerably less expensive.

Acknowledgments

The authors are most grateful to Dr Olivier Macchion for his help and advice. The first author acknowledges HPC at The University of Southern Mississippi supported by the National Science Foundation under the Major Research Instrumentation (MRI) program via Grant # ACI 1626217.

References

[1] Y. Bai, Y. Wu and X. Xie, Uniform convergence analysis of a higher order hybrid stress quadrilateral finite element method for linear elasticity, Adv. Appl. Math. Mech., 8(2016), 399–425.

[2] Y. H. Bai, Y. K. Wu and X. P. Xie, Superconvergence and recovery type a posteriori error estimation for hybrid stress finite element method, Sci. China Math., vol. 59(2016), 1835–1850.

[3] M. Ben-Artzi, I. Chorev, J.-P. Croisille and D. Fishelov, A compact difference scheme for the biharmonic equation in planar irregular domians, SIAM J. Numer. Anal., 47(2009), 3087–3108.

[4] B. Bialecki and G. Fairweather, Matrix decomposition algorithms for separable elliptic boundary value problems in two space dimensions, J. Comput. Appl. Math., 46(1993), 369–386.

[5] B. Bialecki, G. Fairweather and A. Karageorghis, Matrix decomposition algorithms for elliptic boundary value problems: a survey, Numer. Algorithms, 56(2011), 253–295.

[6] M. D. Buhmann, Radial basis functions, Acta Numer., vol. 9, Cambridge Univ. Press, Cambridge, 2000, 1–38.
[7] M. D. Buhmann, Radial Basis Functions: Theory and Implementations, Cambridge Monographs on Applied and Computational Mathematics, vol. 12, Cambridge University Press, Cambridge, 2003.
[8] W. Chen, Z. J. Fu and C. S. Chen, Recent Advances in Radial Basis Function Collocation Methods, Springer Briefs in Applied Sciences and Technology, Springer, Heidelberg, 2014.
[9] A. H.-D. Cheng, Multiquadric and its shape parameter – a numerical investigation of error estimate, condition number, and round-off error by arbitrary precision computation, Eng. Anal. Bound. Elem., 36(2012), 220–239.
[10] P. J. Davis, Circulant Matrices, Second edition, AMS Chelsea Publishing, Providence, Rhode Island, 1994.
[11] G. E. Fasshauer, Meshfree Approximation Methods with MATLAB, Interdisciplinary Mathematical Sciences, vol.6, World Scientific Publishing Co. Pte. Ltd., Hackensack, NJ, 2007.
[12] G. E. Fasshauer and J. G. Zhang, On choosing optimal shape parameters for rbf approximation, Numer. Algorithms, 45(2007), 345–368.
[13] F. Hartmann, Elastostatics, Progress in Boundary Element Methods. Vol. 1 (C. A. Brebbia, ed.), Pentech Press, London, 1981, pp. 84–167.
[14] A. R. H. Heryudono and T. A. Driscoll, Radial basis function interpolation on irregular domain through conformal transplantation, J. Sci. Comput., 44(2010), 286–300.
[15] 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.
[16] E. J. Kansa and Y. C. Hon, Circumventing the ill-conditioning problem with multiquadric radial basis functions: applications to elliptic partial differential equations, Comput. Math. Appl., 39(2000), no. 7-8, 123–137.
[17] A. Karageorghis, The method of fundamental solutions for elliptic problems in circular domains with mixed boundary conditions, Numer. Algorithms, 68(2015), 185–211.
[18] A. Karageorghis, C. S. Chen, and X. Y. Liu, Kansa-RBF algorithms for elliptic problems in axisymmetric domains, SIAM J. Sci. Comput., 38(2016), pp. A435–A470.
[19] A. Karageorghis, C. S. Chen, and Y. S. Smyrlis, A matrix decomposition RBF algorithm: approximation of functions and their derivatives, Appl. Numer. Math., 57(2007), 304–319.
[20] A. Karageorghis, C. S. Chen and Y. S. Smyrlis, Matrix decomposition RBF algorithm for solving 3D elliptic problems, Eng. Anal. Bound. Elem., 33(2009), 1368–1373.
[21] C. K. Lee, X. Liu and S. C. Fan, Local multiquadric approximation for solving boundary value problems, Comput. Mech., 30(2003), 396–409.
[22] J. Le, H. Jin, X. G. Lv and Q. S. Cheng, A preconditioned method for the solution of the Robbins problem for the Helmholtz equation, ANZIAM J., 52(2010), 87–100.
[23] H. B. Li, T. Z. Huang, Y. Zhang, X. P. Liu and T. X. Gu, Chebyshev-type methods and preconditioning techniques, Appl. Math. Comput., 218(2011), 60–270.
[24] M. Li, G. Amazzar, A. Naji and C. S. Chen, Solving biharmonic equation using the localized method of approximate particular solutions, Int. J. Comput. Math. 91(2014), 1790–1801.
[25] X. Y. Liu, A. Karageorghis and C.S. Chen, A Kansa-radial basis function method for elliptic boundary value problems in annular domains, J. Sci. Comput., 65(2015), 1240–1269.
[26] MATLAB, the MathWorks, Inc., 3 Apple Hill Dr., Natick, MA, USA.
[27] S. Rippa, An algorithm for selecting a good value for the parameter c in radial basis function interpolation, Adv. Comput. Math., 11(1999), 193–210.

[28] C. Shu, H. Ding and K. S. Yeo, Local radial basis function-based differential quadrature method and its application to solve two-dimensional incompressible Navier-Stokes equations, Comput. Methods Appl. Mech. Engrg., 192(2003), 941–954.

[29] S. S. Skiena, The Algorithm Design Manual, Second Edition, Springer-Verlag, London.

[30] A. I. Tolstykh, On using RBF-based differencing formulas for unstructured and mixed structured-unstructured grod calculations, in Proceedings of the 16th IMACS World Congress 228, Lausanne, 2000, 4606–4624.

[31] A. I. Tolstykh, D. A. Lipavskii and D. A. Shirobokov High-accuracy discretization methods for solid mechanics, Arch. Mech., 55(2003), 531–553.

[32] A. I. Tolstykh and D. A. Shirobokov, On using radial basis functions in a "finite difference mode" with applications to elasticity problems, Comput. Mech., 33(2003), 68–79.

[33] C. Wang, T. Z. Huang and C. Wen, A new preconditioner for indefinite and asymmetric matrices, A new preconditioner for indefinite and asymmetric matrices, Appl. Math. Comput., 219(2013), 11036–11043.

[34] G. B. Wright, Radial basis function interpolation: Numerical and analytical developments, Ph.D. thesis, University of Colorado, Boulder, 2003.

[35] G. Yao, Local radial basis function methods for solving partial differential equations, Ph.D. thesis, University of Southern Mississippi, 2010.

[36] G. Yao, J. Kolibal and C. S. Chen, A localized approach for the method of approximate particular solutions, Comput. Math. Appl., 61(2011), 2376–2387.