Marie Curie FP7 Program
Uppsala University Dept of Information Technology
Division of Scientific Computing
Box 337, SE-751 05 Uppsala, Sweden
http://www.it.uu.se

FEM-RBF: A Geometrically Flexible, Efficient Numerical Solution Technique for Partial Differential Equations with Mixed Regularity

Project Final Report
Final Publishable Summary Report

Project Acronym: **FEMRBF**
Project Reference: **235730**
Researchers: **Alfa Heryudono (alfa.heryudono@it.uu.se)**
Elisabeth Larsson (elisabeth.larsson@it.uu.se)
Project Website: **https://www.it.uu.se/research/project/rbf/**
Project Duration: **June 1, 2010 – June 30, 2012**

1 About This Report

The purpose of this document is to report research work done by the PI (Alfa Heryudono) and Co-PI (Elisabeth Larsson) at the Department of Information Technology, Division of Scientific Computing, Uppsala University in Sweden. The PI visited Uppsala in 3 time durations: June 2010 - August 2011 (15 months), January 2012 (1 month), and Summer 2012 (1 month) under Marie Curie FP7 program.

The main theme of the report is about numerical study in using hybrid method for solving elliptic partial differential equations whose solutions exhibit mixed regularities. The hybrid scheme is based on finite element and meshfree radial basis function collocation methods. Due to the size and complexity of the project, we created several subprojects. Each subproject leaded us to new observations and results for future publications. The subprojects along with collaborators are listed below:

Research Subprojects		
Subproject	Description	Collaborators
RBF stability and conditioning issue	Differentiation matrices 2D-3D MATLAB code development	Lehto, Fornberg Lehto
FEM-RBF hybrid methods	Coupling techniques MATLAB code development Preliminary theory	 Målqvist
Partition of unity	RBF collocation Coupling technique MATLAB code development Iterative solver Non-circular partitions Fortran package	 Von-Sydow, Ramage Safdari-Vaighani Tillenius
Multiple boundary conditions	Resampling technique Fictitious points	Safdari-Vaighani Safdari-Vaighani
Adaptivity	Adaptive RBF-PU Adaptive RBF-FD	 Driscoll

In this report, we put more emphasis on general overview, preliminary results, practical guides, and figures. Details of mathematical derivations are excluded in this report. For mathematical rigor, readers may consult references provided in the bibliography section. Some of new results will be included in our on-going work for future publications:

1. Stable computation of differentiation matrices and scattered node stencils based on Gaussian radial basis function. E. Larsson, E. Lehto, A. Heryudono, and B. Fornberg. Uppsala technical report 2012. To be submitted to SISC.
2. Partition of unity method for radial basis function collocation problems. E. Larsson and A. Heryudono. In preparation
3. Hybrid finite element and radial basis function collocation methods for elliptic partial differential equations. E. Larsson and A. Heryudono. In preparation.
4. Iterative domain decomposition scheme for RBF-based partition of unity method. E. Larsson, A. Ramage, L. Von-Sydow, and A. Heryudono. In preparation.
5. A numerical study of radial basis function methods for solving the generalized Rosenau equation. A. Safdari-Vaighani, A. Golbabai, A. Heryudono, and E. Larsson, Submitted to Numerical Methods for PDE, under revision.

Results of this research have also been disseminated in the form of talks at international conferences. Those conferences include the 2011 NSF-CBMS Radial Basis Function conference, Society of Industrial and Applied Mathematics (SIAM) sectional conference, and American Mathematical Society (AMS) sectional meeting.

The PI wants to thank Uppsala's Division of Scientific Computing for its hospitality and for providing the best research environment for visiting researchers. The PI feels that Uppsala is his second home and he will surely visit Uppsala again in the future. Special thanks also to Mrs. Carina Lindgren for helping us in administrative work related to this project. We want to thank UPPMAX for providing us access to MATLAB distributed computing engine in high performance computing cluster. Finally, the PI and co-PI want to thank CORDIS Marie Curie FP7 for providing generous support for this project.

2 Introduction

Problems that exhibit solutions with different regularities on localized regions are often encountered in many scientific applications related to solid mechanics, acoustics, and electromagnetic. In those applications, we often have to deal with solving non-periodic time-independent partial differential equations (PDEs) on complex geometries. Furthermore, due to the underlying physical properties such as defects, cracks, and/or material imperfections, solutions may be hard to resolve around the troubled regions. Methods for numerically simulating solutions of such problems must be able to efficiently resolve features around the defects. In the mean time, solutions away from the defects must also be computed as accurate as possible. The goal of this project is to develop and analyze techniques suitable for solving such problems with optimal computational costs and respectable accuracy by combining two powerful methods: finite element (FE) and radial basis function (RBF).

Global approximation methods such as radial basis function methods have proven extremely useful in the numerical solution of boundary value problems in high dimensions. They can be implemented on a flexible mesh and nodes adaptivity can be easily accomplished by adding or deleting points as necessary. RBF methods have been widely used for scattered data interpolation in high dimensions; see [4, 10, 39] and references therein for theory and implementation. Recently RBF collocation methods for PDEs, based on global, non-polynomial interpolants, have been developed [21, 22, 24, 25]. The RBF methods are referred to as *meshfree* methods since they may be implemented on scattered sets of collocation sites (commonly called *centers*) and are not tied to structured grids as are pseudospectral methods. In this way, RBF methods overcome some limitations of pseudospectral methods. RBF collocation methods for steady PDEs have become well established; see [6, 11, 23–25] and references therein.

Although the adaptive RBF [8, 18, 19] method is very promising to be used as a main solver for problems with defects, hybridizing it with finite element methods may have big impact on overall computational cost and efficiency. Around the defect regions, where solutions are less smooth and even "classical" solutions do not exist, the finite element methods will be utilized. Away from those regions, where solutions are smooth and classical solutions can be computed in collocation way, RBF methods will be used. The hybrid method can be constructed by decomposing the computational domains into FEM and RBF sub domains. Those sub domains will result in interface regions, i.e regions where two different methods meet. Solutions must match in terms of continuity and normal derivatives there. This approach combines the strengths of both the finite element methods in the regions, where solutions have less regularities, with the flexibility of RBF methods where required for smooth solution regions. We numerically study and analyze techniques to couple the two methods in a stable way.

3 Hybrid FEM-RBF scheme

In this section, we give a brief introduction to hybrid finite element and radial basis function collocation method for linear elliptic problems. In order to make our illustration simpler and easy to understand throughout this report, we use Poisson equation as examples. The method essentially involves three main steps:

1. The computational domain is decomposed into FEM subdomains and RBF subdomains. FEM subdomains are regions where the underlying solutions are less smooth and/or where classical solutions do not exist. FEM domains may contain cracks and/or defects due to physical anomalies of the underlying problems. On the other hand, RBF subdomains are regions where solutions are known to be smooth. We also assume that locations of FEM and RBF domains are determined beforehand, i.e. the method does not detect regularity of solution and does not automatically decompose domain into FEM and RBF subdomains in an adaptive way.

2. FEM subdomains are then tringulated using finite number of elements and the underlying elliptic problems in those subdomains are discretized in FEM way. RBF regions are discretized by placing RBF nodes. RBF interpolants generated using those nodes are then used to collocate the elliptic problems there. All necessary FEM matrices as well as RBF differentiation matrices are generated for later use in the assembling step. The domain decomposition will result in the formation of interface regions which connect unknown values u and the normal derivatives $\frac{\partial u}{\partial n}$ from different subdomains.

3. FEM matrices and RBF differentiation matrices are used to assemble global matrix operator of the PDE. Depending on whether the underlying problem is linear or non-linear, we obtain the global solution by solving system of linear or non-linear equations.

The following subsections 3.1 and 3.2 will describe the 3 steps above for simple 1D and 2D Poisson equations respectively for the case of 2 subdomains where solutions are smooth everywhere.

3.1 One Dimensional Case

Consider a 1D Poisson equation on interval $[a, c]$ with Dirichlet boundary conditions at each end points

$$\begin{aligned} -u'' &= f \quad \text{for} \quad a < x < c, \\ u(a) &= \alpha \quad u(c) = \beta. \end{aligned}$$

We decompose the interval into two subintervals: $[a, b]$ and $[b, c]$. Solution in $[a, b]$ is spanned by FEM basis and solution in $[b, c]$ is spanned by RBFs. This implies that the PDE in subinterval $[a, b]$ is formulated in weak sense and the PDE in subinterval $[b, c]$ is formulated in collocation sense. Note that in the 1D case, one may have more options in choosing the type of collocation method. Instead of using RBF, pseudospectral collocation or finite difference method may also be used. The point $x = b$ is the interface point, where we connect u and $\frac{\partial u}{\partial n}$ (normal derivative). The problem turns into:

$$\begin{aligned} -u''_{\text{FEM}} &= f \quad \text{for} \quad a < x \leq b, \\ u_{\text{FEM}}(a) &= \alpha, \\ \frac{\partial u_{\text{FEM}}}{\partial n} &= \frac{\partial u_{\text{RBF}}}{\partial n}, \quad \text{and} \quad u_{\text{FEM}} = u_{\text{RBF}} \quad \text{at} \quad x = b, \\ -u''_{\text{RBF}} &= f \quad \text{for} \quad b < x < c, \\ u_{\text{RBF}}(c) &= \beta. \end{aligned} \tag{1}$$

The domain decomposition of the interval $[a, c]$ can be illustrated by the figure below. In order to avoid confusion, vertices in the FE region are denoted by x and nodes in the RBF domain are denoted by ξ. Furthermore, solutions in the FE domain and RBF domain are denoted by u and w respectively.

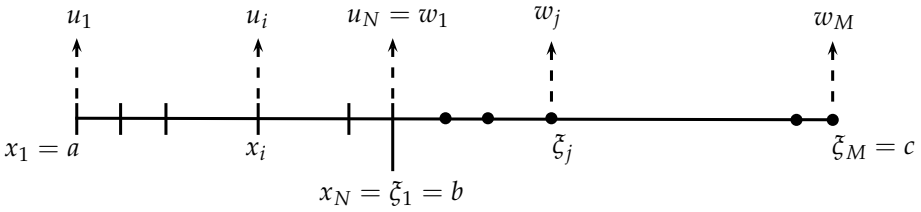

Figure 1: The 1D interval is decomposed into 2 subdomains. FE region on the left and RBF region on the right. Discretization nodes in each subdomain are denoted by symbols x and ξ respectively. Solutions in each subdomain are denoted by symbols u and w respectively.

From now on, we drop the subscript terms FEM and RBF on u to simplify our mathematical symbols. As usual, for the PDE in the FEM region, we multiply the solution by the test function v and integrate by parts to obtain

$$\int_a^b u'v'dx = \int_a^b fv + v(b)u'(b) - v(a)u'(a),$$

or by using inner product notation

$$< u', v' > = < f, v > + v(b)u'(b) - v(a)u'(a).$$

By letting ϕ_i to be a hat function centered at x_i

$$\phi_i(x) = \begin{cases} 1 + \frac{x - x_i}{x_i - x_{i-1}} & x_{i-1} \leq x < x_i \\ 1 - \frac{x - x_i}{x_{i+1} - x_i} & x_i \leq x \leq x_{i+1} \end{cases},$$

solution and its first derivative in the FEM region can be spanned by the *tent* bases as

$$\begin{aligned} u &= \sum_{i=1}^{N}{}' u_i \phi_i, \\ \frac{\partial u}{\partial x} &= \sum_{i=1}^{N}{}' u_i \frac{\partial \phi_i}{\partial x}. \end{aligned}$$

Note that the term $'$ in the sum means that functions centered at the end points of the interval: i.e. ϕ_1 and ϕ_N are *half* tent. If we discretize the FEM domain with equally-spaced points with equal spacing h, the FEM system matrix can be written as

$$\frac{1}{h}\begin{bmatrix} 2 & -1 & & & & \\ -1 & 2 & -1 & & & \\ & \ddots & \ddots & \ddots & & \\ & & -1 & 2 & -1 & \\ & & & -1 & 1 & \end{bmatrix}\begin{bmatrix} u_2 \\ u_3 \\ \vdots \\ u_{N-1} \\ u_N \end{bmatrix} = \begin{bmatrix} < f, \phi_2 > \\ < f, \phi_3 > \\ \vdots \\ < f, \phi_{N-1} > \\ < f, \phi_N > \end{bmatrix} + \begin{bmatrix} u_1/h \\ 0 \\ \vdots \\ 0 \\ u_x(x_N) \end{bmatrix}.$$

The FEM system matrix above cannot be solved independently as a standalone boundary value problem due to the unavailability of $u_x(x_N)$ value on the right hand side. That information will be supplied by matching $u_x(x_N)$ from the first derivative of RBF interpolant at the interface point $x_N = b$:

$$u_x(x_N = \xi_1 = b) = \sum_{j=1}^{M} d^1_{1j} w_j,$$

where d^1_{1j} are entries of RBF differentiation matrix (first derivative with respect to x, see section 5.1) at x_N and w_j are unknown RBF solutions at points ξ_j in the RBF domain. Note that discretization points in the RBF region does not have to be equally-spaced. RBF system matrix at $\xi_2, \cdots, \xi_{M-1}$ is given by

$$\begin{bmatrix} -d^2_{21} & \cdots & -d^2_{2M-1} \\ \vdots & \ddots & \vdots \\ -d^2_{M-11} & \cdots & -d^2_{M-1M-1} \end{bmatrix}\begin{bmatrix} w_2 \\ \vdots \\ w_{M-1} \end{bmatrix} = \begin{bmatrix} f(\xi_2) \\ \vdots \\ f(\xi_{M-1}) \end{bmatrix} + \begin{bmatrix} 0 \\ \vdots \\ d^2_{M-1M}\beta \end{bmatrix},$$

where d^2_{ij} are entries of RBF second derivative (with respect to x) matrix as explained in section 5.1. The FE and RBF system matrices along with condition at the interface are then assembled into "global" system matrices as

$$\begin{bmatrix} 2 & -1 & & & & & & \\ -1 & 2 & -1 & & & & & \\ & \ddots & \ddots & \ddots & & & & \\ & & -1 & 2 & -1 & & & \\ & & & -1 & 1 - hd^1_{11} & \cdots & -hd^1_{1M-1} \\ & & & & -d^2_{21} & \cdots & -d^2_{2M-1} \\ & & & & \vdots & \ddots & \vdots \\ & & & & -d^2_{M-11} & \cdots & -d^2_{M-1M-1} \end{bmatrix}\begin{bmatrix} u_2 \\ u_3 \\ \vdots \\ u_{N-1} \\ u_N \\ w_2 \\ \vdots \\ w_{M-1} \end{bmatrix} = \begin{bmatrix} h < f, \phi_2 > \\ h < f, \phi_3 > \\ \vdots \\ h < f, \phi_{N-1} > \\ h < f, \phi_N > \\ f(\xi_2) \\ \vdots \\ f(\xi_{M-1}) \end{bmatrix} + \begin{bmatrix} \alpha \\ 0 \\ \vdots \\ 0 \\ h\beta d^1_{1M} \\ 0 \\ \vdots \\ \beta d^2_{M-1M} \end{bmatrix},$$

which can then be solved with a suitable linear solver. In the global system matrix above, the continuity condition $u_N = w_1$ is simultaneously satisfied.

3.2 Two Dimensional Case

Hybridizing FEM and RBF-FD method in 2D is similar as in the 1D case. Consider the 2D Poisson equation:

$$\begin{aligned} -\Delta u &= f \quad \text{in} \quad \Omega, \\ u &= g \quad \text{on} \quad \partial\Omega, \end{aligned} \tag{2}$$

where Ω (see the Figure 2 as an example) consists of 2 regions: Ω_1 (Finite element region) and Ω_2 (RBF region). The interface curve Ω_{12} separates mesh and meshfree regions. Points at the interface are both FEM nodes and RBF nodes. The Poisson equation (2) can be written as the following:

$$\begin{aligned} -\Delta u_{\text{FEM}} &= f \quad \text{in} \quad \Omega_1, \\ u_{\text{FEM}} = u_{\text{RBF}}, \quad \text{and} \quad \frac{\partial u_{\text{FEM}}}{\partial n} &= \frac{\partial u_{\text{RBF}}}{\partial n} \quad \text{at the interface} \quad \partial\Omega_{12}, \\ -\Delta u_{\text{RBF}} &= f \quad \text{in} \quad \Omega_2, \\ u_{\text{RBF}} &= g \quad \text{on} \quad \partial\Omega. \end{aligned}$$

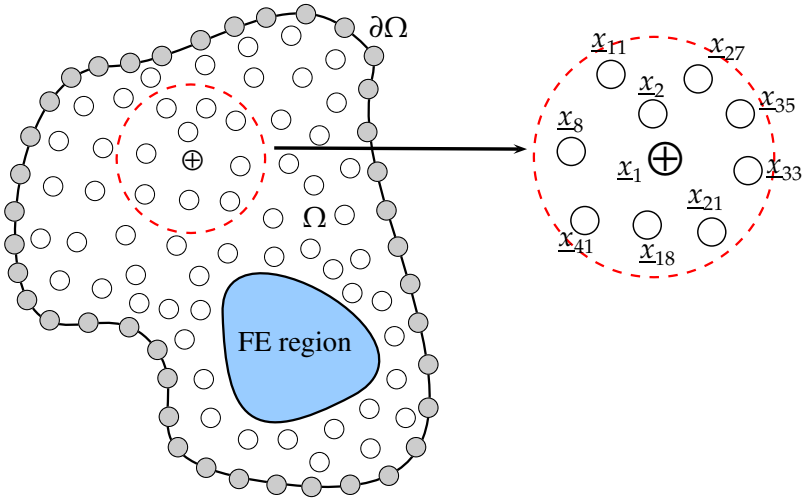

Figure 2: A 2D irregular domain is decomposed into 2 subdomains: FE region and RBF (meshfree) region. White nodes are RBF points inside the domain and grey nodes are RBF boundary points. Note that sizes of points are made bigger for easy visualization.

As usual, the solution in the FEM domain can be written as

$$u_{\text{FEM}} = \sum_{i=1}^{N}{}' u_i^F v_i(\underline{x}),$$
$$\frac{\partial u_{\text{FEM}}}{\partial n} = \sum_{i=1}^{N}{}' u_i^F \frac{\partial v_i(\underline{x})}{\partial n}.$$

Note that the term $'$ in the sum means that basis functions centered at points which lie on the interface are *cutted* tents. The $v_i(\underline{x})$ is linear on each triangle and take the value 0 at all nodes $\underline{x}_j$ except at $\underline{x}_i$. The PDE is then multiplied with test function v (v is spanned by $v_i(\underline{x})$) and integrate over domain Ω. Due to the compact support property of the test function v_i's, we end up only to integrate over Ω_1

$$-\int_{\Omega_1} \Delta u_{\text{FEM}} v dA = \int_{\Omega_1} f v dA.$$

By using Green's identity applied to the left hand side term of the equation above, the weak formulation of the PDE on Ω_1 can be written as

$$\int_{\Omega_1} \nabla u_{\text{FEM}} \cdot \nabla v dA - \int_{\partial\Omega_{12}} \frac{\partial u_{\text{FEM}}}{\partial n} v d\Gamma = \int_{\Omega_1} f v dA. \tag{3}$$

To couple with RBF method, we need to replace $\frac{\partial u_{\text{FEM}}}{\partial n}$ with $\frac{\partial u_{\text{RBF}}}{\partial n}$ in the equation above. Note that $u_{\text{FEM}} = u_{\text{RBF}}$ will automatically be satisfied at interface points. Assembling the system matrix can be done in similar way as in the 1D case. FEM discretization of (3) will result in under-determined FEM system matrix. What we need now are differentiation matrices to form RBF system matrix to be augmented to the FEM system matrix such that the global system matrix for the hybrid method can be solved. The following sections describe how to generate RBF interpolants and differentiation matrices.

4 RBF Interpolant and Differentiation Matrices

In this section, we briefly discuss how to form an RBF interpolant and to compute its derivatives. Given a set Ξ of n distinct RBF nodes called *centers* $\underline{x}_1^c, \ldots, \underline{x}_n^c$ in $\mathbb{R}^d$, the RBF interpolant takes the form

$$s(\underline{x}) = \sum_{k=1}^{n} \lambda_k \phi^k(\underline{x}), \tag{4}$$

where $\phi^k(\underline{x})$ is a radial basis function centered at $\underline{x}_k^c$. While there are a large number of known RBFs, the RBFs that are infinitely differentiable and contain a free *shape* parameter ε have been the most widely used. The inverse multiquadric (IMQ) and Gaussian (GA) which represent this category of RBF are given respectively by

$$\text{(IMQ):} \quad \phi^k(\underline{x}) = \frac{1}{\sqrt{1+\varepsilon^2\|\underline{x}-\underline{x}_k\|^2}}, \qquad \text{(GA):} \quad \phi^k(\underline{x}) = e^{-\varepsilon^2\|\underline{x}-\underline{x}_k\|^2}, \tag{5}$$

where $\|\cdot\|$ represents Euclidean distance.

The coefficients λ_k are determined by enforcing the interpolation condition

$$s(\underline{x}_k) = u(\underline{x}_k), \tag{6}$$

at a set of nodes that coincide with the centers. Enforcing the interpolation conditions at n centers results in a $n \times n$ linear system

$$A\boldsymbol{\lambda} = \mathbf{u}, \tag{7}$$

where

$$A = \begin{bmatrix} \phi^1(\underline{x}_1^c) & \cdots & \phi^n(\underline{x}_1^c) \\ \vdots & \ddots & \vdots \\ \phi^1(\underline{x}_n^c) & \cdots & \phi^n(\underline{x}_n^c) \end{bmatrix}, \quad \text{and} \quad \mathbf{u} = \begin{bmatrix} u(\underline{x}_1) \\ \vdots \\ u(\underline{x}_n) \end{bmatrix}. \tag{8}$$

The matrix A is known as the *interpolation matrix*. In the case of Gaussian RBF, A is positive definite and therefore invertible [27]. For further study of RBF, one may consult books by Buhmann [4], Wendland [39], and Fasshauer [10].

Differentiation operator matrices can be formed in a straightforward way. As an example, first derivative operator with respect to x can be written as

$$D = A_x A^{-1}, \tag{9}$$

where

$$A_x = \begin{bmatrix} \phi_x^1(\underline{x}_1^c) & \cdots & \phi_x^n(\underline{x}_1^c) \\ \vdots & \ddots & \vdots \\ \phi_x^1(\underline{x}_n^c) & \cdots & \phi_x^n(\underline{x}_n^c) \end{bmatrix}. \tag{10}$$

The matrix vector product $D\mathbf{u}$ will result in an approximation of the first derivative of $u(x)$ at $\underline{x}_1, \underline{x}_2, \cdots, \underline{x}_n$, respectively. Note that the first row of D in (9) contains the differentiation matrix elements needed to compute $s_x(\underline{x}_1)$, the second row for $s_x(\underline{x}_2)$, and so on. In other words, at any desired point $\underline{x}$, evaluating $s_x(\underline{x})$ is the same as multiplying first derivative weights at $\underline{x}$ and function values given by

$$s_x(\underline{x}) = \underbrace{\begin{bmatrix} \phi_x^1(\underline{x}) & \cdots & \phi_x^n(\underline{x}) \end{bmatrix} \begin{bmatrix} \phi^1(\underline{x}_1^c) & \cdots & \phi^n(\underline{x}_1^c) \\ \vdots & \ddots & \vdots \\ \phi^1(\underline{x}_n^c) & \cdots & \phi^n(\underline{x}_n^c) \end{bmatrix}^{-1}}_{\text{first derivative weights of } s(\underline{x})} \begin{bmatrix} u(\underline{x}_1) \\ \vdots \\ u(\underline{x}_n) \end{bmatrix}. \tag{11}$$

Differentiation matrices D_k for higher derivatives of order k can be formed in a similar fashion. The use of RBF differentiation matrices to solve time-independent and/or time dependent problem is the core of RBF-Pseudospectral (RBF-PS) methods [9]. Compared to polynomial-based PS method [2, 5, 12, 20, 37], RBF-PS do not need to be in its polynomial limit and may use non-smooth bases or bases with larger ε if needed. Based on this fact, one may see RBF-PS method as the generalization of polynomial-based PS method.

In this project, we have developed stable and accurate spatial discretization operators formed by RBF differentiation matrices based on RBF-QR [14, 17]. The RBF-QR technique is used to circumvent the ill-conditioning issue [3, 31] of the interpolation matrix due to the use of flat RBFs (cases when $\epsilon \to 0$). Manuscript 1 listed in the project summary is the outcome of this project.

5 PDE Collocation in the RBF Regions

The way we collocate PDEs in the RBF regions are based on the choice between two methods: finite difference mode [13, 36, 40] and partition of unity.

5.1 Finite Difference Mode

This method has the following features:

1. Flexibility in laying out points for discretizing a domain.
2. Unstructured local stencils with flexible stencil sizes.
3. Simple method for computing stencil weights: i.e. elements of differentiation matrices.
4. Similar computational costs with finite difference.

5.1.1 Discretization

The discretize step consists of three ingredients: laying out points, compute local interpolants, and form differentiation matrices. As a simple example, Figure 2 illustrates $N = 101$ collocation points (40 of them on the boundary) in the RBF region on a irregular domain in 2D. Let $\mathcal{X} = \{\underline{x}_1, \ldots, \underline{x}_N\}$ be a set of RBF collocation points and let $\mathcal{L}_j = \{\underline{x}_k\}$, where $k \in \{1, \ldots, N\}$, be a set of RBF points to form stencil weights at $\underline{x}_j$. Note that $\mathcal{L}_j \subset \mathcal{X}$ contains several neighbor points of $\underline{x}_j$ including $\underline{x}_j$ itself. The number of local points n_{loc} in each $\mathcal{L}_j$ can be the same for all j or different at each j. In our case, we set n_{loc} to be the same in order to guarantee that all local interpolants provide more or less the same approximation order. Moreover, we call the point $\underline{x}_j$ the *master* node of the set $\mathcal{L}_j$. All other $n_{\text{loc}} - 1$ points in the set $\mathcal{L}_j$ are *slave* nodes.

A particular stencil on Figure 2 has been magnified to show an example of a stencil formed by the set $\mathcal{L}_1$. In that case, $\underline{x}_1$ is the master node of $\mathcal{L}_1 = \{\underline{x}_1, \underline{x}_{21}, \underline{x}_8, \underline{x}_{18}, \underline{x}_2, \underline{x}_{27}, \underline{x}_{33}, \underline{x}_{35}, \underline{x}_{11}, \underline{x}_{41}\}$ with $n_{\text{loc}} = 10$. Slave nodes in $\mathcal{L}_1$ can be obtained using neighbor search algorithm. Notice that points in $\mathcal{L}_1$ are still listed with respect to their global indices, i.e $\mathbf{globInd}(\mathcal{L}_1) = \{1, 21, 8, 18, 2, 27, 33, 35, 11, 41\}$. However, when working locally, for convenience one may want to reindex using local indices $\mathbf{locInd}(\mathcal{L}_1)$ which runs from 1 to n_{loc}. Mapping from local indices $\mathbf{locInd}(\mathcal{L}_1)$ to $\mathbf{globInd}(\mathcal{L}_1)$ and vice versa can be done using pointers.

5.1.2 Local RBF Interpolants

Working in local indices, the RBF local interpolant with master node $\underline{x}_j$ takes the form

$$s_j(\underline{x}) = \sum_{k=1}^{n_{\text{loc}}} \lambda_k \phi^k(\underline{x}), \tag{12}$$

where $\phi^k(\underline{x})$ is a radial basis function centered at $\underline{x}_k$. Suppose we want to interpolate a function f on $\mathcal{L}_j$, the coefficients λ are determined by enforcing the interpolation condition

$$s_j(\underline{x}_k) = f(\underline{x}_k), \tag{13}$$

resulting in a $n_{\text{loc}} \times n_{\text{loc}}$ linear system

$$A\underline{\lambda} = \underline{f}, \tag{14}$$

to be solved for RBF expansion coefficients λ. The matrix A with entries

$$a_{\ell k} = \phi^k(\underline{x}_\ell), \qquad \ell, k = 1, \ldots, n_{\text{loc}} \tag{15}$$

is called the *local interpolation matrix*.

Since $\underline{\lambda} = A^{-1}\underline{f}$, the local interpolant $s_j(\underline{x})$ evaluated at any point $\underline{x}$ can be written in Lagrange formulation as

$$s_j(\underline{x}) = \sum_{k=1}^{n_{\text{loc}}} \Psi^k(\underline{x}) f_k, \tag{16}$$

where

$$\underline{\Psi} = \begin{bmatrix} \Psi^1(\underline{x}) & \cdots & \Psi^{n_{\text{loc}}}(\underline{x}) \end{bmatrix} = \begin{bmatrix} \phi^1(\underline{x}) & \cdots & \phi^{n_{\text{loc}}}(\underline{x}) \end{bmatrix} \begin{bmatrix} & A^{-1} & \end{bmatrix}, \tag{17}$$

and $\underline{\Psi}$ is a vector containing interpolation weights at $\underline{x}$. RBF basis shapes can also be changed by tweaking ε. Dealing with small ε is one of the challenging problems in RBF research since the basis becomes flatter and therefore closer to being linearly dependent. This eventually leads to severe ill-conditioning issue of A. Small ε cases unravel interesting connection between spectral and RBF approximations. In the limit $\varepsilon \to 0$ the RBF interpolant is equivalent to the minimal-degree Lagrange interpolating polynomial [7]. In higher dimensions, the limit may not exist but when it does it is a multivariate polynomial [15, 26, 32]. Conditions where RBF approximations produce spectral accuracy are discussed in [16, 17, 29, 30, 41].

5.1.3 Differentiation Matrices

Computing derivatives of local interpolants s_j is straightforward. As an example, weights of first derivative with respect to x evaluated at $\underline{x}$ is given by

$$[s_j(\underline{x})]_x = \sum_{k=1}^{n_{\text{loc}}} \Psi_x^k(\underline{x}) f_k, \tag{18}$$

where

$$\underline{\Psi}_x := \begin{bmatrix} \Psi_x^1(\underline{x}) & \cdots & \Psi_x^{n_{\text{loc}}}(\underline{x}) \end{bmatrix} = \begin{bmatrix} \phi_x^1(\underline{x}) & \cdots & \phi_x^{n_{\text{loc}}}(\underline{x}) \end{bmatrix} \begin{bmatrix} & A^{-1} & \end{bmatrix}, \tag{19}$$

Computing higher derivatives or derivatives with respect to other independent variables can be done in the same manner. As an example

$$\underline{\Psi}_{xx} := \begin{bmatrix} \Psi_{xx}^1(\underline{x}) & \cdots & \Psi_{xx}^{n_{\text{loc}}}(\underline{x}) \end{bmatrix} = \begin{bmatrix} \phi_{xx}^1(\underline{x}) & \cdots & \phi_{xx}^{n_{\text{loc}}}(\underline{x}) \end{bmatrix} \begin{bmatrix} & A^{-1} & \end{bmatrix}, \tag{20}$$

$$\underline{\Psi}_y := \begin{bmatrix} \Psi_y^1(\underline{x}) & \cdots & \Psi_y^{n_{\text{loc}}}(\underline{x}) \end{bmatrix} = \begin{bmatrix} \phi_y^1(\underline{x}) & \cdots & \phi_y^{n_{\text{loc}}}(\underline{x}) \end{bmatrix} \begin{bmatrix} & A^{-1} & \end{bmatrix}. \tag{21}$$

Note that since the local interpolation matrix A only needs to be inverted once, one may want to compute weights $\Psi_x, \Psi_y, \Psi_{xx}, \cdots$ at $\underline{x}$ and stack them as a matrix

$$\underline{\Psi}^M := \begin{bmatrix} \phi_x^1(\underline{x}) & \cdots & \phi_x^{n_{\text{loc}}}(\underline{x}) \\ \phi_y^1(\underline{x}) & \cdots & \phi_y^{n_{\text{loc}}}(\underline{x}) \\ \phi_{xx}^1(\underline{x}) & \cdots & \phi_{xx}^{n_{\text{loc}}}(\underline{x}) \\ & \cdots & \end{bmatrix} \begin{bmatrix} & A^{-1} & \end{bmatrix}. \tag{22}$$

Derivative weights of local interpolants s_j can be assembled to form a global differentiation matrices. As an example, suppose we want to form the first derivative matrix operator with respect to x. The steps for computing elements of the sparse differentiation matrix D can be described as the following: First, for every $\underline{x}_j$, we create its interpolation matrix A_j using RBF centers in the neighborhood set $\mathcal{L}_j$. Second, when the inverse of A_j is known, elements of the local derivative operator of row j of D can be easily computed. D will have $n_{\text{loc}}N$ nonzero entries. As an example, let us take the stencil $\mathcal{L}_1$ with master node $\underline{x}_1$ from Figure 2. Local interpolation matrix for the stencil $\mathcal{L}_1$ and differentiation weights at $\underline{x}_1$ are given by

$$A_1 = \begin{bmatrix} \phi^1(\underline{x}_1) & \phi^{21}(\underline{x}_1) & \cdots & \cdots & \phi^{41}(\underline{x}_1) \\ \phi^1(\underline{x}_{21}) & \phi^{21}(\underline{x}_{21}) & \cdots & \cdots & \phi^{41}(\underline{x}_{21}) \\ \vdots & & \ddots & & \vdots \\ \vdots & & & \ddots & \vdots \\ \phi^1(\underline{x}_{41}) & \phi^{21}(\underline{x}_{41}) & \cdots & \cdots & \phi^{41}(\underline{x}_{41}) \end{bmatrix}, \tag{23}$$

$$\begin{bmatrix} d_{1,1} & d_{1,21} & \cdots & d_{1,41} \end{bmatrix} = \begin{bmatrix} \phi_x^1(\underline{x}_1) & \phi_x^{21}(\underline{x}_1) & \cdots & \phi_x^{41}(\underline{x}_1) \end{bmatrix} \begin{bmatrix} & A_1^{-1} & \end{bmatrix}.$$

The next step is to fill in entries of the first row of D, where only entries at column $\{1, 21, 8, 18, 2, 27, 33, 35, 11, 41\}$ are non zeros. We redo the same procedure to fill in elements in other rows of D. Higher order differentiation matrices can be computed in the same manner. Note that computing entries for each row of D is an embarrassingly parallel process. Thus, the computation can be distributed to multiple computing cores. The differentiation matrices are also sparse containing $n_{loc}N$ non zero entries.

5.2 Partition of Unity

The partition of unity method for numerically solving Boundary Value Problems (BVPs) on regular and/or irregular geometry has gained popularity due to the work of Babuška and Melenk [1]. In this method, the geometry Ω is covered by overlapping patches $\{\Omega_j\}$ that correspond to a partition of unity $\{w_j\}$ subordinate to the cover. The global solution on the domain is then approximated by ansatz

$$u(\underline{x}) = \sum_{j=1}^{M} w_j(\underline{x}) u_j(\underline{x}), \tag{24}$$

where u_j is the local approximant on patch Ω_j and

$$\sum_{j=1}^{M} w_j(\underline{x}) = 1. \tag{25}$$

The formulation (24) offers greater flexibilities to adjust local approximants u_j by any combinations of the following:

1. Enlarging or shrinking patches. [h type version].
2. Choosing function spaces for the local approximants u_j tailored to the properties of the PDEs. [p type version]

In the same reference, Babuška and Melenk also pointed out that the global solution u inherits not only the approximation properties of the local spaces u_j but also the smoothness of the partition of unity.

Incorporating localized RBF interpolation into the partition of unity method to solve boundary value problems is straightforward. We can call the method RBF-PU method. It essentially uses RBF interpolants as local approximants u_j. The ansatz can then be collocated to satisfy the BVPs. Without loss of generality, let us describe the method in solving equation (2). We proceed with the backbone of PUM collocation scheme of four procedures:

Discretize - - Cover - - Collocate - - Solve

to solve u on Ω. The details are described below.

5.2.1 Discretize

Domain of computation is discretized by laying out points inside the domain and on its boundary. Figure 3 shows an example of a discretized irregular domain Ω with an almost uniform distribution of points.

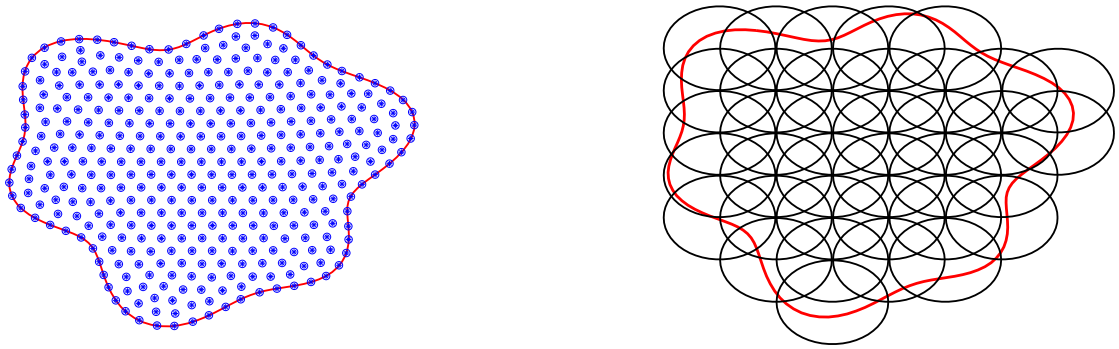

Figure 3: **Left:** Irregular domain is discretized with an almost uniform distribution points. **Right:** Domain is covered by overlapping disks generated from boxes of level 3.

5.2.2 Cover

The goal of this partition covering scheme is to cover the domain with patches. In two dimensional cases, quadtree type subdivisions can be used as a simple partitioning scheme. We begin by covering the domain with a square box. We call this box a **level** 0 box. We then naturally divide the level 0 box into 4 boxes of level 1 and continue this subdivision process up to the desired level k. At this point, the domain is covered by 4^k non-overlapping boxes of level k. Any boxes, which are lying entirely outside the domain, are deleted. To make the scheme even simpler, any boxes, whose midpoints are outside the domain, are also deleted. As an alternative, one may also subdivide/refine boxes which are crossed by boundary.

The next step is to create some overlapping of boxes. This can be done by enlarging the area of each box. In addition to square overlapping patches, we can also form overlapping disks by setting their radiuses to be half of diameters of the corresponding boxes. The bookkeeping process must also be done. For every discretization point, we have to record which disks it belongs to. For every disk partition, we have to record which points lie inside.

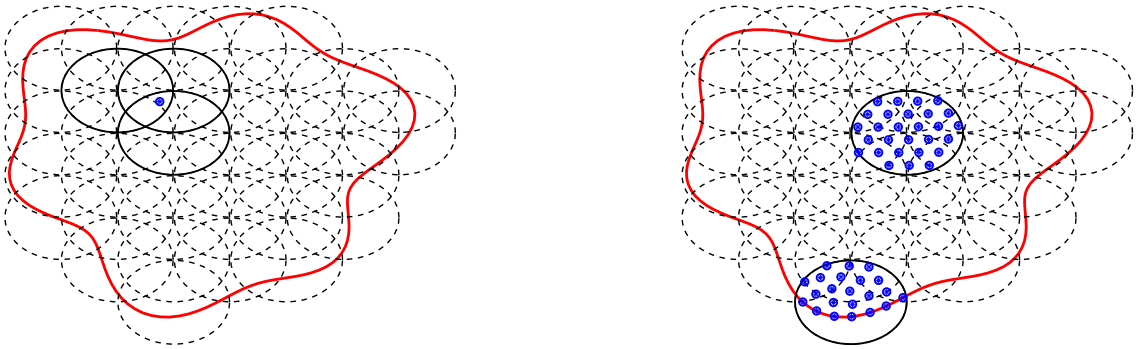

Figure 4: **Left:** A point can be a member for at least one disk. **Right:** Each disk has its own points to generate local RBF interpolants.

Finding a *good* cover for irregular domain can be a challenge. In the case where the domain is quasi uniformly discretized, we prefer to have covers which have consistent percentage of overlapping and consistent node density (i.e. number of nodes per partition) throughout the domain. This makes our life easier in guarantying identical quality of approximations on each partition. However, this ideal case breaks down in regions near boundaries. For example, if covers away from boundary curves can enjoy 10% overlap with approximately n_{loc} nodes per partition, covers that are crossed by boundaries usually have bigger overlaps with some of their neighboring covers and larger node density to accommodate uncovered nodes or uncovered area. Uncovered nodes in most cases lie inside deleted boxes described in previous section.

In addition to discretize and then cover, another idea is to cover the domain first and then *fill in* each box with approximately n_{loc} points. If we fill each partition with identical distribution of nodes, then boxes away from boundaries will have *periodic/isotropic* like structures. RBF approximation on cover with this special case can be computed once and used on other covers of the same type thus saving computational costs. As usual, covers near boundaries are specially treated. If needed, one may also freely fill in each partition with different distribution of nodes.

We also have to pay attention on regions in the domain as well as line segments on boundary curves with no RBF nodes in them. Those regions and line segments must also be covered. In short, nothing in the domain and on the boundary is left uncovered. Figure 5 shows a simple procedure to cover the boundary of an irregular domain.

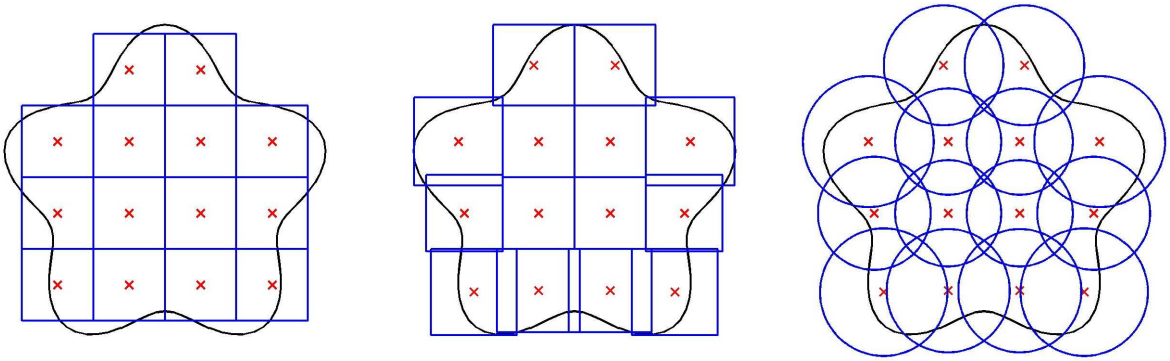

Figure 5: The domain is covered by non overlapping boxes. Due to curve boundary, some boundary regions are not covered. Boxes that cross boundary curve are then enlarged to guarantee full covering. Diameter of those corrected boxes are then used to determine radiuses of overlapping disks.

5.2.3 Collocate and Solve

At every discretization point we collocate ansatz (24) to satisfy (2):

$$\begin{aligned} -\Delta u(\underline{x}_i) = \sum_{j=1}^{M} w_j(\underline{x}_i)\Delta u_j(\underline{x}_i) + 2\nabla u_j(\underline{x}_i)\cdot\nabla w_j(\underline{x}_i) + \\ u_j(\underline{x}_i)\Delta w_j(\underline{x}_i) = f(\underline{x}_i) \quad \underline{x}_i \in \Omega, \\ u(\underline{x}_i) = \sum_{j=1}^{M} w_j(\underline{x}_i)u_j(\underline{x}_i) = g(\underline{x}_i) \quad \underline{x}_i \in \partial\Omega. \end{aligned} \tag{26}$$

All operators Δ and ∇ for u are generated using RBF-FD differentiation matrices. Values of Δw and ∇v at each point can be computed exactly. Equation (26) leads us to solve sparse linear system either using direct or iterative methods. Manuscript 2 and 4 listed in the project summary are currently in preparation to disseminate our results about RBF collocation-based partition of unity method. We are also preparing MATLAB educational toolbox for this purpose (see the screenshot in section 8).

6 Numerical Experiments

All our numerical experiments are carried out in MATLAB (tested on version 2008 - 2012) on a MacPro workstation. Since RBF local interpolants and RBF-FD differentiation matrices for a particular stencil can be computed separately from other stencils, we use commands such as **parfor** and **spmd** in MATLAB's parallel computing toolbox to distribute independent jobs. Those embarrassingly parallel processes carry over to the RBF-PU since local interpolants and differentiation matrices for a particular cover can also be computed separately from other covers.

6.1 RBF-FD as a Full Poisson Solver

The RBF-FD differentiation matrices described in section 5.1 can easily be utilized for solving boundary value problems. As a basic test problem, we solve equation (2) where $\partial\Omega$ (see the left part of Figure 6) is a starfish like shape with parametric equation

$$r_b(\theta) = 0.8 + 0.1(\sin(6\theta) + \sin(3\theta)), \quad \theta \in [0, 2\pi). \tag{27}$$

The collocation process involves three steps. First, we discretize the domain with $N = N_i + N_b$ points, where N_i is the number of interior points and N_b is the number of boundary points. The Poisson equation is then collocated

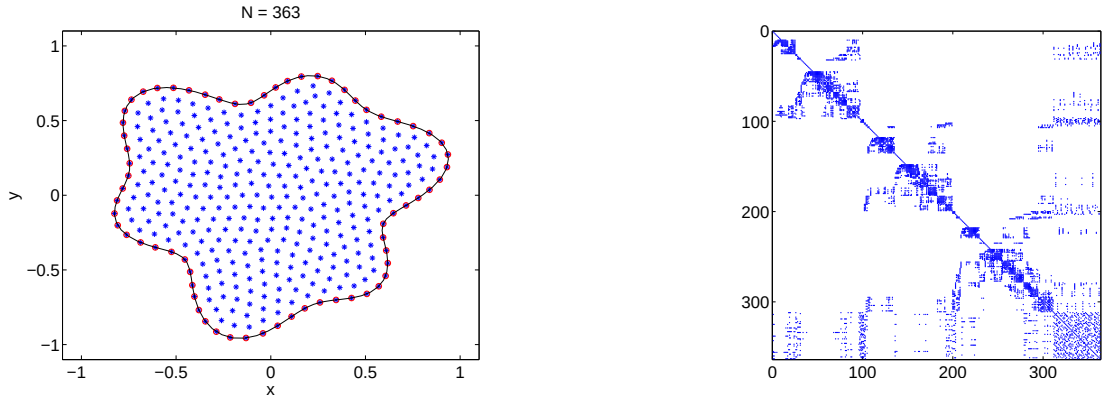

Figure 6: **Left:** An example of starfish like domain discretized with $N = 363$ uniformly distributed nodes. **Right:** Sparsity distribution of the system matrix with stencil size $n_{\text{loc}} = 21$.

at the interior points using RBF-FD stencils for approximating the Laplacian. This results in an under-determined system of size $N_i \times N$. By augmenting the system with N_b equations from the boundary condition we end up with an $N \times N$ linear system of equations. Finally, we solve the linear system to obtain the nodal solution values. As in the finite difference case, the system matrix is sparse. The right part of Figure 6 shows an example of the sparsity distribution of a system matrix with $N = 363$ and stencil size $n_{\text{loc}} = 21$ after a minimal degree reordering.

The process of distributing RBF points uniformly can for example be carried out by treating nodes as being connected by springs that repel and attract one another until an equilibrium state is achieved [28]. The forcing function and boundary condition of (2) are chosen such that the exact solutions are the following smooth functions

$$u_1(x,y) = \sin(\pi x)\sin(\pi y), \qquad \text{(Test case 1)}$$
$$u_2(x,y) = (x^2 + y^2 - 0.25)^2. \qquad \text{(Test case 2)}$$

Figure 7 shows numerical experiments for the two test cases, where the total number of points N is kept fixed. The accuracy of the numerical solutions of (2) with respect to the stencil sizes (n_{loc}) for $\varepsilon = 0.1$ can be seen in the two leftmost subfigures.

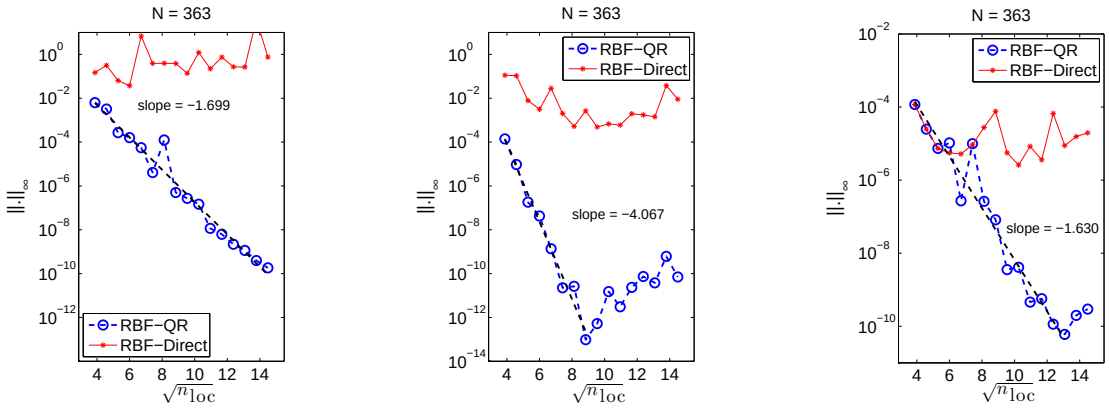

Figure 7: Comparisons of convergence trends for RBF-QR and RBF-Direct with respect to the square root of the stencil sizes $\sqrt{n_{\text{loc}}}$ when solving the Poisson equation. The two leftmost subfigures are for Test case 1 and Test case 2 respectively with $\varepsilon = 0.1$. The right subfigure is for Test case 1 with $\varepsilon = 1$.

For RBF-QR the convergence trends in both cases are spectral of the form

$$\|\cdot\|_\infty \propto \exp(s\sqrt{n_{\text{loc}}}), \quad s < 0,$$

with slopes s around -1.7 for Test case 1 and -4.1 for Test case 2. The results are compared with using RBF-Direct (in double precision). As can be seen in the figures, RBF-Direct does not converge due to severe ill-conditioning. The rightmost subfigure shows the result for Test case 1 with the larger shape parameter value $\varepsilon = 1$. In that case,

RBF-Direct agrees with the results obtained with RBF-QR for smaller stencil sizes before leveling off again due to the conditioning issue.

6.2 Notes on Cases when $n_{\mathbf{loc}} = N$

With RBF-FD, one may experiment to enlarge all stencil sizes all the way up to the total number of points N. In that case, the RBF-FD method will turn into global RBF method with a dense system matrix. In [17], Fornberg, Larsson, and Flyer pointed out that clustering points toward the boundary is a crucial step to maintain the stability of the global RBF interpolant especially when small shape parameter is used. In other words, the convergence trends should remain *flat* after hitting the machine precision.

The question of *When do RBF interpolants behave like polynomial interpolants ?* has stimulated research activity in this subject. At least in one dimensional case, this is indeed the case. When the basis is *flat*, i.e cases where $\varepsilon \to 0$, Driscoll and Fornberg [7] pointed out that one dimensional RBF interpolant is equivalent to Lagrange interpolating polynomial. Their surprising observations open the door to analyze the stability of RBF interpolant using potential theory. When Gaussian RBFs are used, Platte and Driscoll [30] showed that RBF interpolant generated using equally-spaced centers on unit interval can be transformed into polynomials. Their method also allows us to determine best distribution of nodes suitable for one particular shape parameter to prevent the Runge phenomenon. Cases for higher dimensions are much subtler. In [15] Fornberg, Wright, and Larsson made some observations and conjectured that Gaussian RBF interpolants will never diverge as ε vanishes. Schaback [32] also showed that the RBF interpolant is equivalent to multivariate Boor/Ron polynomial.

In addition to clustering inner points towards the boundary, our numerical experiments also show that one should avoid overcrowding boundary points. As a simple example, suppose we solve equation (2) on a unit disk with $N = N_i + N_b$, where N_i inner points (a subset of Halton points) and N_b boundary points (equally-spaced). Figure 8 shows the discretization points before and after clustering for the case $N_i = 721$ and $N_b = 154$.

Figure 8: **Left:** A subset of 721 Halton points inside a unit disk is used as RBF inner nodes. In addition, 154 boundary points are distributed equally on the perimeter. **Right:** The Halton points are clustered toward the boundary with a mapping function $\tilde{r}_i = \sin(\frac{\pi}{2} r_i)$. The boundary points are left intact.

For testing convergence, we vary inner points N_i. The boundary points are equally-spaced with $N_b = \frac{2\pi}{q_i}$, where q_i can be chosen as filling distance of the unclustered/clustered inner points or average minimum distance of inner points. For Poisson equation with smooth known exact solutions as references, Figure 9 shows that the convergence is evidently spectral with respect to $\sqrt{N}$, i.e. the convergence trends follow

$$\|\cdot\|_\infty \propto \exp(a\sqrt{N}), \quad a < 0. \tag{28}$$

However, as can be observed in Figure 9, the spacing rule $N_b = \frac{2\pi}{q_i}$ seems not to be working well when $\sqrt{N} \geq 20$. The error is growing and the location of $\|\cdot\|_\infty$ seems to occur at random locations, i.e not always at inner points near boundary points. This problem might be created by putting too many points on the boundary. We experiment by fixing the number of N_i, use the rule $N_b = \frac{2\pi}{q_i}$ as usual, and then remove the boundary points one by one until we find the best minimum global error of the solution. Figure 10 shows that the best accuracy is achieved when the ratio N_b/N_i is around 10% for cases $N_i = 721, 910, 1218$.

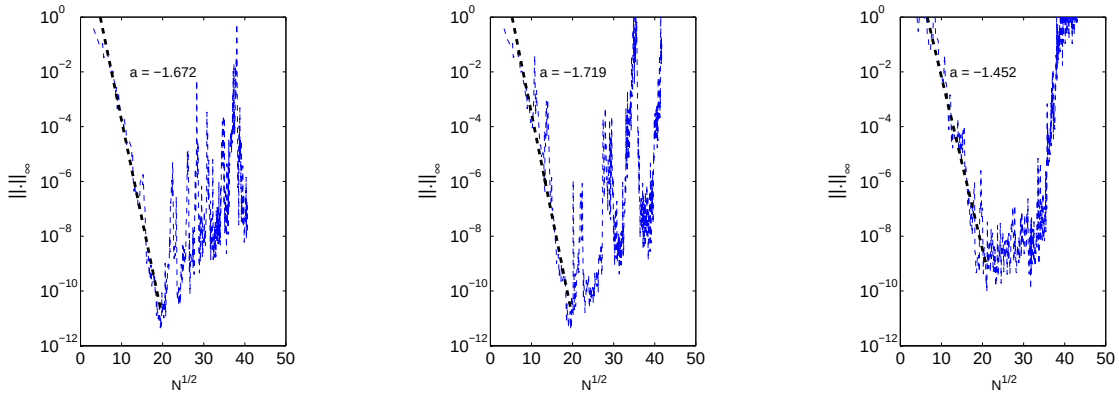

Figure 9: **Left:** Spectral convergence trend where the q_i is set using filling distance of clustered inner points. **Middle:** Case where q_i is set using filling distance of unclustered inner points. **Right:** Case where q_i is set using average minimum distance of inner points.

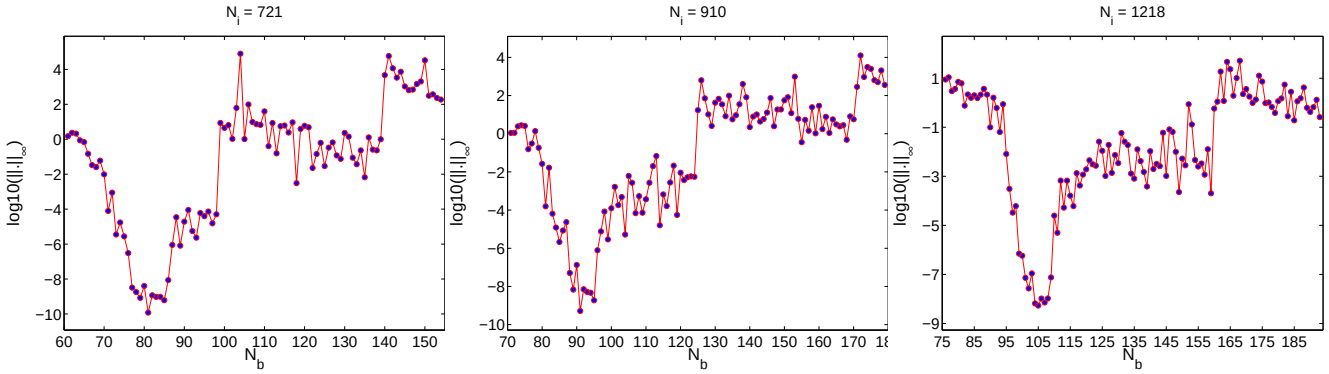

Figure 10: Given a fixed N_i clustered points, overcrowding boundary points seems to result in poor accuracy for the global solution unless the ratio N_b/N_i are kept to about 10%.

6.3 RBF-PU as a Full Poisson Solver

We test RBF-PU method for solving 2D and 3D of equation (2) on irregular domains. Partition of unity weight function w_j for each cover Ω_j is constructed using Shepard's formulation [33] as the following:

$$w_j(\underline{x}) = \frac{\psi_j(\underline{x})}{\sum_{\ell=1}^{M} \psi_j(\underline{x})},$$

where compactly supported Wendland's functions [38] are used for $\psi_j(\underline{x})$. In particular, we use (in terms of radius measured from disk center)

$$\psi(r) = \begin{cases} (4r+1)(1-r)^4 & r < R \\ 0 & \text{otherwise,} \end{cases} \tag{29}$$

where R is radius of support. One may also try another Wendland's function such as

$$\psi(r) = \begin{cases} (35r^2+18r+3)(1-r)^6 & r < R \\ 0 & \text{otherwise.} \end{cases}$$

In all our numerical experiments, we use (29) since we only need up to continuous second derivatives for w_j.

For our problems, we have freedom in choosing node distribution. Therefore, the process of covering domains is separated from the process of discretizing them. One have a choice to lay out RBF points either "globally" throughout the domain based on a particular node distribution or "locally" by filling out each partition with its own node

distribution. One of many widely used examples for globally generating scattered points is by using Halton points on a unit box. By scaling the unit box to enclose an irregular domain, we remove Halton points outside the domain and keep the inside points. Boundary points are generated independently. For testing purposes, we also use vertices from mesh generators such as DISTMESH [28] to create uniformly distributed RBF nodes. In practice, however, costly mesh generation should be avoided to keep the spirit of meshfree methods alive. Even though one prefers to use available mesh generation software (due to nice GUI and solid object manipulation capability) to discretize a complicated 2D/3D objects, good quality meshes are not needed since RBF method is also *mesh tolerant*.

6.3.1 Two Dimensional Case

As a test problem, we solve (2) on a domain with parametric equation

$$r_b(\theta) = 0.9 + 0.1\sin(6\theta) + 0.02\sin(9\theta), \tag{30}$$

where the forcing function f and boundary condition g are chosen such that the exact solution is

$$u(r,\theta) = \frac{1}{0.25r^2+1}. \tag{31}$$

Sparsity distribution of the RBF-PU system matrix and the solution can be seen in Figure 11.

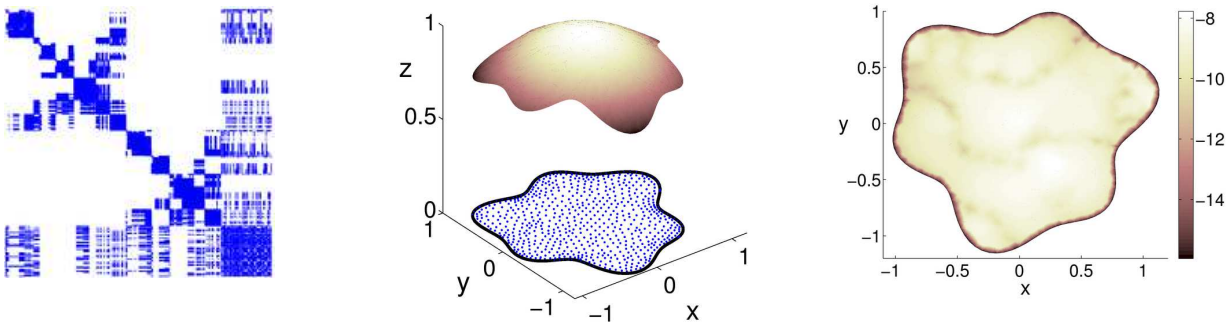

Figure 11: **Left:** An example of sparsity distribution of RBF-PU system matrix. **Middle:** Solution on starfish domain. **Right:** $\log_{10}$ error distribution on the domain for $N = 1105$.

We divide our error convergence tests into 3 cases. One case for showing algebraic convergence and two cases for spectral convergence. The convergence test measures how error of numerical solutions (compared to known exact solutions) decreases with respect to total number of nodes N or total number of local nodes n_{loc}. Note that for dimension greater than one, we should measure the error convergence with respect $N^{1/d}$ or $n_{\text{loc}}^{1/d}$, where d is dimension. However, if we prefer to keep using N or n_{loc}, then we should expect the rates to be d times slower compared to one dimensional cases.

We are now showing how absolute error decreases algebraically with respect to N. In this numerical experiment, quasi-uniform points are used. Each cover (disk) should contain the same n_{loc} points. However, due to the shape of irregular geometry, some covers may contain couple of points greater than n_{loc} due to boundary control process described in section 5.2.2. For a fixed number n_{loc}, we vary N by setting $N = N_\varrho = \varrho N_0$, where N_0 is a chosen fixed integer number and ϱ is the desired positive integer multiplier.

The next step is to choose the size of nonoverlapping boxes that cover our domain such that the corresponding overlapping disks (each has radius r_{loc}) contain at least n_{loc} RBF nodes each. This can be done by satisfying the condition such that local node density in each partition is the same as the global node density. The density relation is given by

$$\frac{N}{\mathcal{A}} = \frac{n_{\text{loc}}}{\pi r_{\text{loc}}^2} \quad \rightarrow \quad \frac{\varrho N_0}{\mathcal{A}} = \frac{n_{\text{loc}}}{\pi r_{\text{loc}}^2}, \tag{32}$$

where $\mathcal{A}$ is the total area of the domain. r_{loc} can then be written as

$$r_{\text{loc}} = \frac{r_0}{\sqrt{\varrho}}, \quad \text{where} \quad r_0^2 = \frac{n_{\text{loc}}\mathcal{A}}{\pi N_0}. \tag{33}$$

Thus, setting N to be ϱ times of N_0 will result in having radius of partitions $\varrho^{-1/2}$ smaller than r_0 in order to keep constant density relation. Covering the domain with nonoverlapping boxes of level k with side length $\sqrt{2}r_{\text{loc}}$ is essentially the same as having a bounding box of level 0 with side length $\ell = 2^{(k-0.5)}$.

The left figure in Figure 12 shows algebraic convergence trend with $N_0 = 160$ using $n_{\text{loc}} = 21, 28, 45, 66$ corresponds to one-half up to fourth order of convergence, i.e. $N^{-3/2}, N^{-2}, N^{-3}, N^{-4}$ respectively.

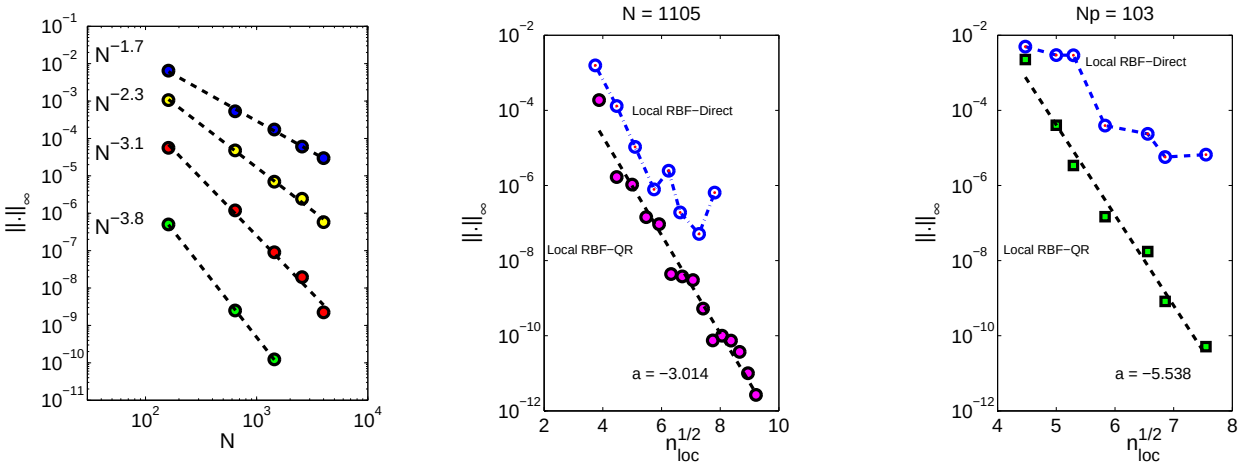

Figure 12: Error convergence trends of local RBF based on partition of unity method. **Left:** Algebraic convergence trend where domain is partitioned with overlapping disks using $21, 28, 45, 66$ local points respectively. **Middle:** Spectral convergence obtained by using a fixed total number of points ($N = 1105$) and disks overlapping are increased based on number of local points n_{loc}. **Right:** Spectral convergence obtained by using a fixed total number of partitions ($N_p = 103$) and local points n_{loc} are varied. Without RBF-QR, rounding errors kick in sooner to prevent convergence.

For the first spectral convergence test, the domain is quasi-uniformly discretized using a fixed $N = 1105$ nodes. The domain is covered with partitions of level 4 that each contains $n_{\text{loc}} = 15$. In other words, the nonoverlapping boxes are generated with

$$r_{\text{loc}}^2 = \frac{n_{\text{loc}}\mathcal{A}}{\pi N} = \frac{15\mathcal{A}}{\pi 1105}.$$

The $\|\cdot\|_\infty$ is then computed. We then recompute $\|\cdot\|_\infty$ for cases $n_{\text{loc}} = 20, 25, 30, \cdots, 85$. Note that while incrementally increasing n_{loc} (therefore increasing r_{loc}) using k–nearest neighbor search, the center of partitions are left intact. This will result in heavily overlapped partitions especially for larger n_{loc}. Figure 12 shows that the convergence is evidently spectral.

For the second test, we are going with different route. The domain is first covered with fixed number of partitions N_p. In our example, we choose $N_p = 103$. These partitions have fixed radiuses and overlaps. Boxes of corresponding partitions are then *filled in* with RBF nodes such that each partition will contain n_{loc}. Figure 12 shows the spectral convergence.

6.3.2 Three Dimensional Case

Our preliminary results for solving Poisson equations on three dimensional solid domain are computed without using boundary control technique and RBF-QR as in our two dimensional examples. The absent of boundary control technique is purely for simplicity. With the absence of boundary control technique, there are possibilities where some regions on the surface of the domain are uncovered. In order to make sure that our three dimensional domain is fully covered, we examine projection of the 3D domain onto several two dimensional planes.

Our test is carried out on a solid sphere. The sphere is discretized as the following: On the surface, RBF nodes are laid out based on minimum energy distribution. In this case, we can think of RBF nodes as point charges of same types. Their mutual repulsive forces allow them to freely move (restricted on the surface) to find equilibrium positions until minimum total potential energy is achieved. Sloan and Womersley [34, 35] have used this technique to obtain good interpolation points on a sphere.

For inner points, we can follow similar approaches as in two dimensional cases. For example, we can choose uniform nodes distribution where nodes density for each partition is roughly the same as the global node density.

A second choice will be to use non-uniformly distributed Halton points in a box $[-1,1] \times [-1,1] \times [-1,1]$ where points located outside the unit ball are discarded. The third choice will be by filling each partition with different distribution of points. For simplicity and testing purposes, we choose the first approach. Figure 13 shows an example of cover partitions of a solid sphere and $\log_{10}$ error distribution of numerical solution compared to known exact solution.

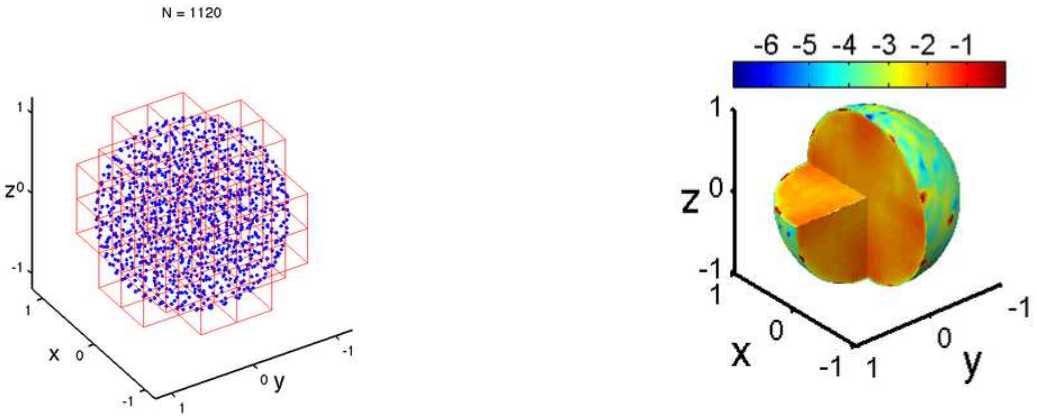

Figure 13: **Left:** Discretization points and partition covers of a solid sphere. **Right:** Distribution of $\log_{10}$ errors for the case $N = 1120$.

Figure 14 shows spectral convergence trends

$$\|\cdot\|_\infty \propto \exp(a \cdot n_{\text{loc}}^{1/3}), \quad a < 0. \tag{34}$$

for 2 numerical tests: (1) Fixed total number of points N and then enlarge partition sizes, (2) Fixed number of partition and their radiuses and then enlarge the number of local points n_{loc}. Note that in experiment (2), the total number of points N changes.

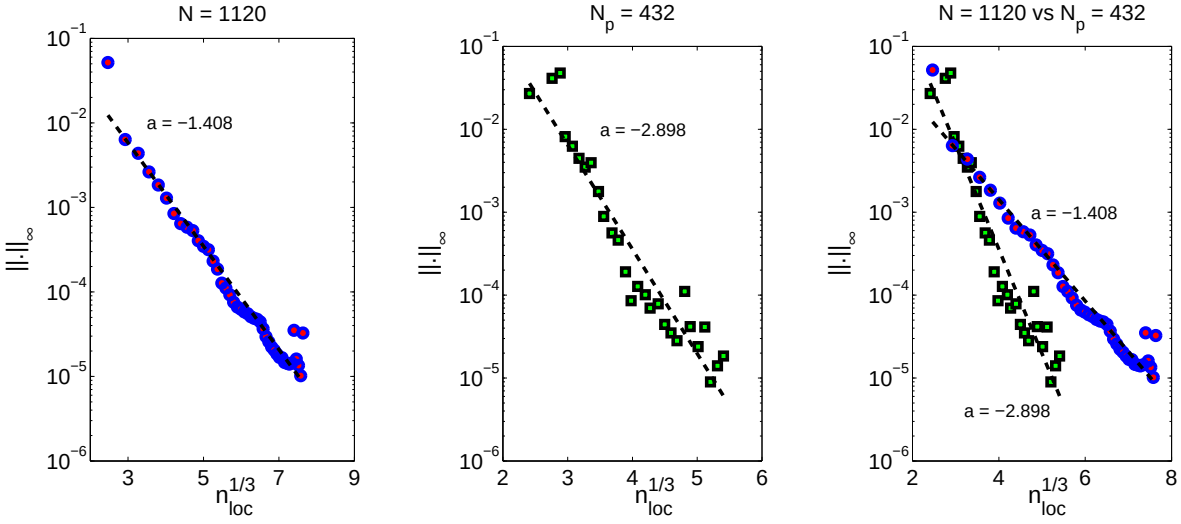

Figure 14: Convergence trends for solving 3D Poisson equation on a sphere with RBF-PU.

7 Hybrid FEM-RBF for Smooth Problems

For 1D case, the code is very simple and is provided in section 10. In the 2D case, we use MATLAB PDE toolbox to generate mesh in the finite element domain and to easily access stiff matrices and all needed functions (e.g. forcing functions), which are already computed on those meshes.

7.1 Numerical Experiments in 1 Dimension

Let us first solve equation (1) with hybrid finite element and pseudospectal method on interval $[-1, 1]$ with interface point at $x = -0.5$. The finite element region is discretized with 61 equally spaced points and the pseudospectral region is discretized with 51 Chebyshev points. The forcing function f is chosen based on taking the negative Laplacian of the exact solution

$$u(x) = \exp(-\cos(3\pi(x - 0.5))) - 2x - \sin(\pi(x + 0.5)).$$

Dirichlet boundary conditions are enforced based on the exact solution value at the two end points. Figure 15 shows numerical result and error distribution. As expected the errors are typically higher in the neighborhood of the interface point.

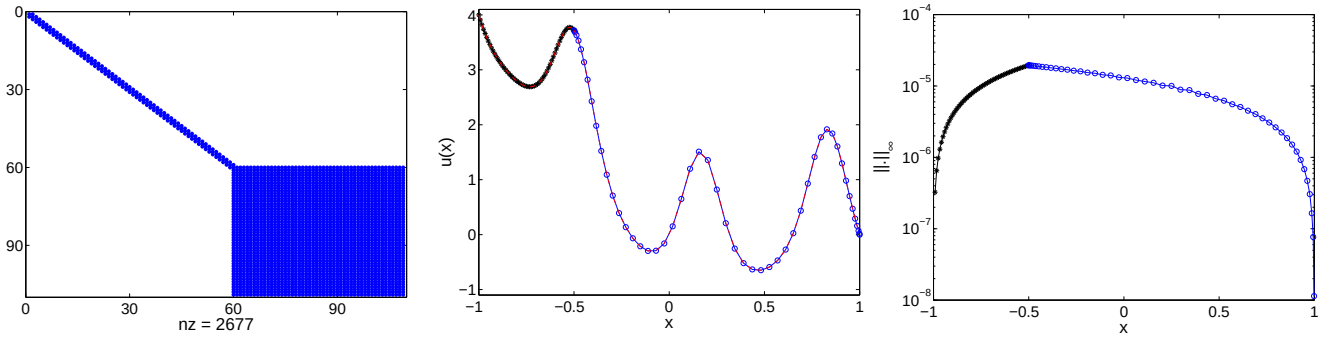

Figure 15: **Left:** Sparsity profile of the 1D FEM-PS system matrix. **Middle:** Numerical solution. **Right:** Error distribution on the interval. Notice that the error is typically higher at and around the interface point.

For the second experiment, we replace pseudospectral method with RBF-FD method. The interface point is slightly changed to $x = -0.325$. The finite element region is discretized with 28 equally-spaced points and RBF region is discretized with 54 points. Points in the RBF region are not tied to specific grid. Indeed, we first generate equally spaced points and then jiggle them randomly. The stencil size for RBF-FD is $n_{\mathbf{loc}} = 11$. Figure 16 shows numerical solution and error distribution using the hybrid method.

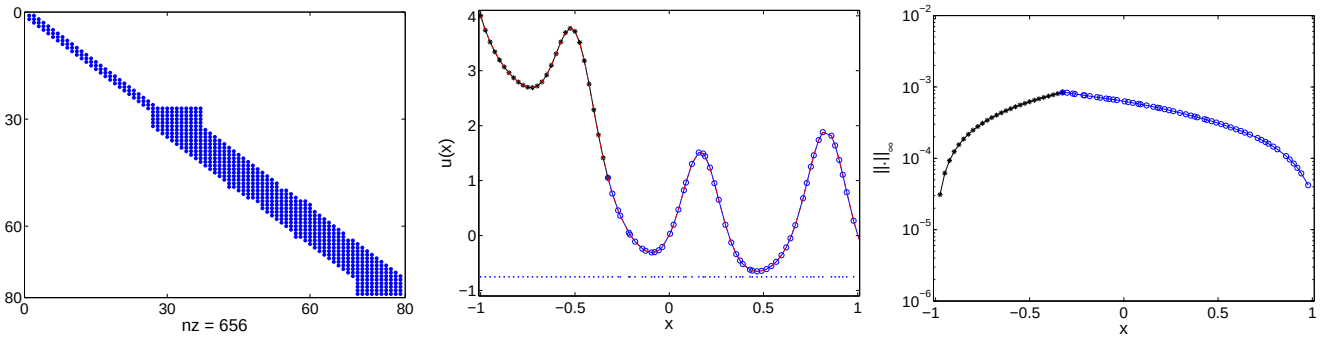

Figure 16: **Left:** Sparsity profile of the 1D FEM-RBF system matrix. The stencil size for the RBF-FD method is 11. **Middle:** Numerical solution. **Right:** Error distribution on the interval. As usual, the error is typically higher at and around the interface point.

Note that finite element and RBF only share a point at the interface. This creates a one sided difference style stencils for the RBF-FD in the neighborhood of the interface point. One may instead have an option to let the RBF method taking several FE points inward as RBF points to form centered difference style stencils around the interface region. We do not have this option for the FEM-PS since the Chebyshev points cannot be altered. To generate Figure 16, one can use MATLAB codes provided in section 10.

7.2 Numerical Experiments in 2 Dimension

We utilize MATLAB PDE toolbox for getting all triangular meshes and matrices needed for creating FEM system matrix in the finite element region. In the PDE toolbox, basic elliptic equation with generalized Neumann (Robin

BCs) is given by

$$\begin{aligned} -\nabla \cdot (c\nabla u) + au &= f \quad \text{on} \quad \Omega, \\ n \cdot (c\nabla u) + \bar{q}u &= \bar{g} \quad \text{on} \quad \partial\Omega_{12}, \end{aligned}$$

where n is the outward unit normal, c is a constant and $\bar{g}, \bar{q}$ are functions defined on $\partial\Omega_{12}$. Again, deriving from the Green's identity, we will have

$$\int_{\Omega} (c\nabla u) \cdot \nabla v + auv dA - \int_{\partial\Omega_{12}} (-\bar{q}u + \bar{g})v d\Gamma = \int_{\Omega} fv dA$$

to get the matrix system in the form

$$(K + M + Q)U = F + G, \tag{35}$$

where entries of those matrices are:

$$K_{ij} = \int_{\Omega} (c\nabla v_j) \cdot \nabla v_i dA \quad \text{(Stiffness matrix)}, \qquad M_{ij} = \int_{\Omega} a v_j v_i dA \quad \text{(Mass matrix)},$$

$$Q_{ij} = \int_{\partial\Omega_{12}} \bar{q} v_j v_i d\Gamma, \qquad F_i = \int_{\Omega} f v_i dA, \qquad G_i = \int_{\partial\Omega_{12}} \bar{g} v_i d\Gamma.$$

In our case, $c = 1, a = 0, \bar{q} = 0$ and $\bar{g} = n \cdot (c\nabla u_{\text{RBF}})$ at the interface. The discretized values of $\bar{g} = \bar{g}_1, \bar{g}_2, \cdots, \bar{g}_{N_b}$ are constants that contain normal derivatives of u_{RBF} at points on the interface. Hence,

$$G_i = \int_{\partial\Omega} \bar{g}_i v_i d\Gamma.$$

If we let $V_i = \int_{\partial\Omega} v_i d\Gamma$, we can write G in matrix form as

$$G = \begin{bmatrix} V_1 & & 0 \\ & \ddots & \\ 0 & & V_{N_b} \end{bmatrix} \begin{bmatrix} \quad \text{RBF normal derivative matrix at the interface} \quad \end{bmatrix} \begin{bmatrix} u_1 \\ \vdots \\ u_{Nb} \\ w_2 \\ \vdots \\ w_M \end{bmatrix}.$$

As usual, the under-determined system of equations (35) will be augmented by RBF system matrix. In this preliminary result, we will use RBF-FD (without RBF-QR) to discretize the PDE in the RBF region instead of RBF-PU.

As a simple test problem, we solve equation (2) where $\partial\Omega$ (see the most left figure of Figure 17) is a starfish like shape with parametric equation

$$r_b(\theta) = 1 + 0.1(\sin(6\theta) + \sin(3\theta)), \quad \theta \in [0, 2\pi). \tag{36}$$

The domain is decomposed into 2 subdomains: A disk with radius $r = 0.5$ for FE and the rest of the domain for RBF. Those subdomains only share points at the interface. Figure 17 shows the case with 146 FE vertices and 392 RBF nodes. The middle and right figures of Figure 17 shows sparsity distribution of the assembled system matrix (RBF stencil size is $n_{\text{loc}} = 35$) before and after minimal degree ordering. In MATLAB, the command for applying approximate minimal degree ordering to a sparse matrix is **amd**. The forcing function of the Poisson equation and Dirichlet boundary condition are chosen based on the exact solution

$$u(x, y) = \exp(-20(x^2 + y^2)) - x^2 + y^3. \tag{37}$$

Numerical solution and error distribution obtained from this hybrid method can be seed in Figure 18.

Following the numerical experiment in 1 dimensional case, one may want to avoid the one sided difference stencils for the RBF-FD in the region around interface by letting RBF method to take several points inward the FE region as interpolation points. The left part of Figure 19 shows nodes distribution with 224 FE points and 492 RBF points. The RBF is allowed to use FE points between radiuses $r = 0.35$ and $r = 0.5$. Numerical solution and error distribution are shown in Figure 20.

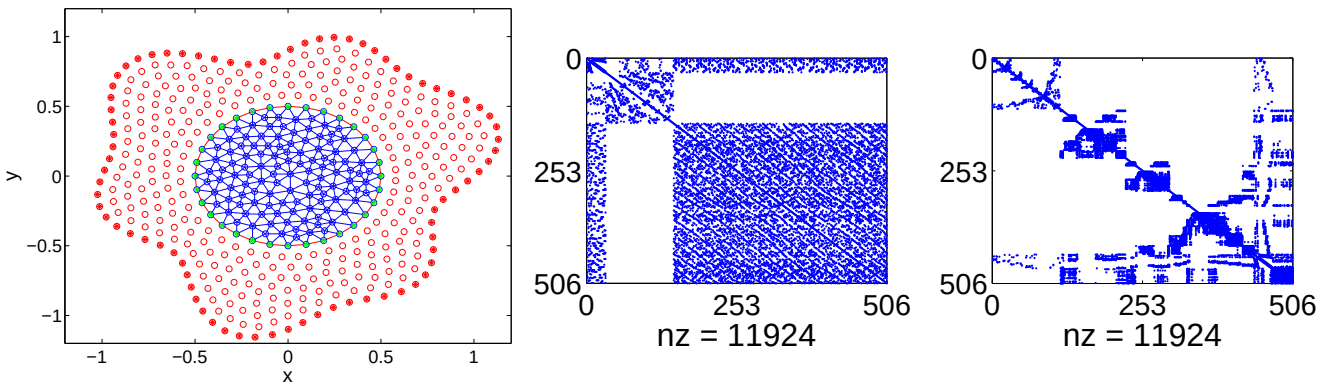

Figure 17: **Left:** A 2D irregular domain is decomposed into 2 subdomains: a disk with radius 0.5 (FE region) and starfish with a hole (RBF region). Note that FE and RBF only share points at the interface (the perimeter of the disk). **Right:** Sparsity distribution of the system matrix of the hybrid method before and after minimal degree ordering.

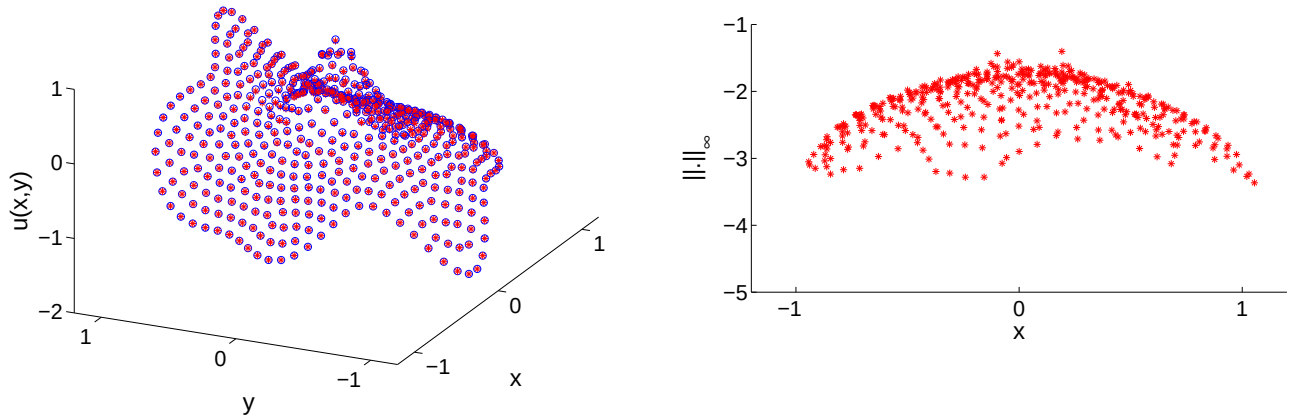

Figure 18: Numerical solution and error distribution obtained using hybrid FE-RBF with one sided difference RBF-FD stencils in the neighborhood of interface.

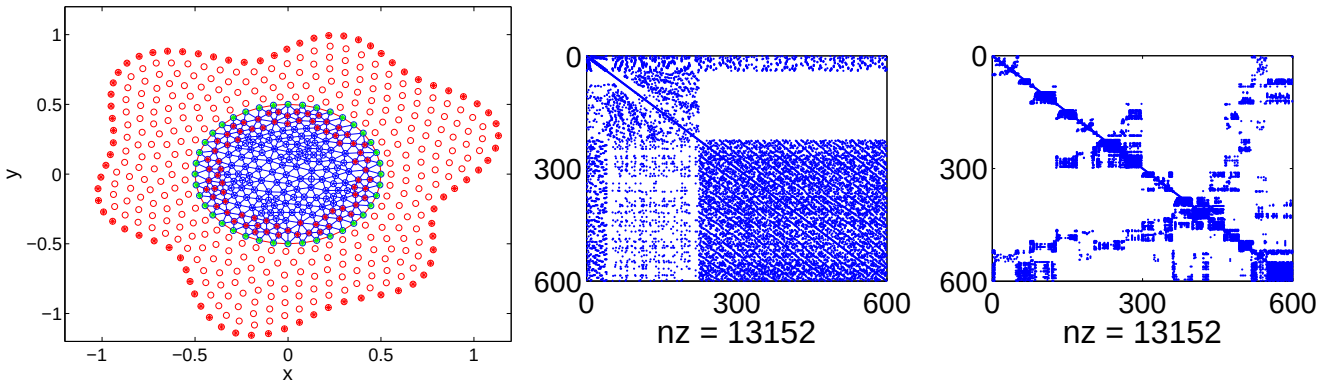

Figure 19: **Left:** A 2D irregular domain is decomposed into 2 subdomains: a disk with radius 0.5 (FE region) and starfish with a hole (RBF region). RBF method is allowed to use FE points between radiuses $r = 0.35$ and $r = 0.5$ to form more *centered difference* like stencils. **Right:** Sparsity distribution of the system matrix of the hybrid method before and after minimal degree ordering.

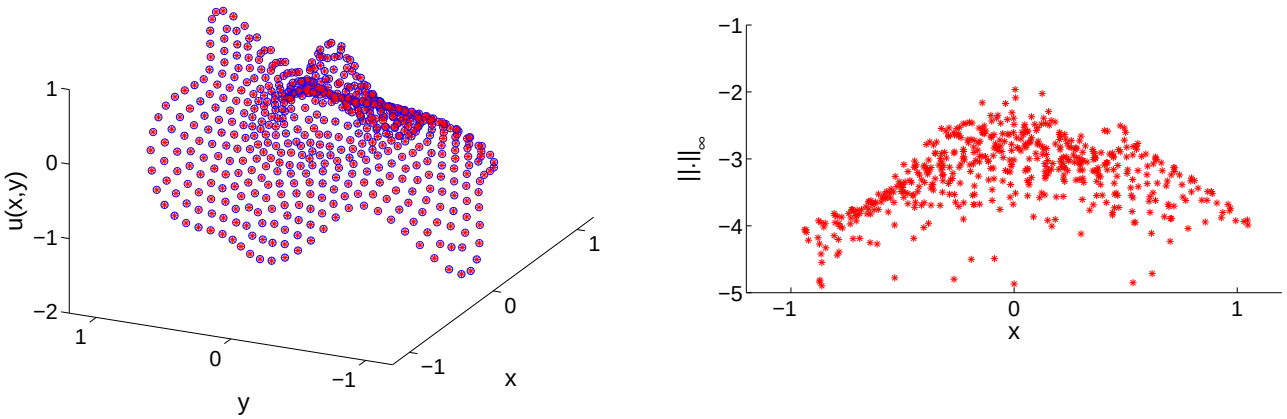

Figure 20: Numerical solution and error distribution obtained using hybrid FE-RBF with centered difference RBF-FD stencils in the neighborhood of the interface.

8 On-going Software Project

All codes as part of this research will be integrated into the MATLAB toolbox called RBF-PDETool (currently developed by the PI and E. Larsson) based on Adaptive local RBF, RBF-QR, and RBF partition of unity method. The software will provide an easy to use GUI as well as access to its MATLAB commands. The software screenshot for our very early version can be seen in Figure 21. Users should be able to generate RBF differentiation matrices, to select solver for solving boundary and/or initial boundary value problems, to specify boundary conditions, and to simulate solutions. If users have accesses to MATLAB Parallel Computing Toolbox, the differentiation matrices can be assembled in embarrassingly parallel way using multiple CPU and GPU cores to speed up computations. The toolbox will have options to run ready to use example problems (templates) which can be used as teaching and research tools. Results and codes will be disseminated on our publicly accessible website https://www.it.uu.se/research/project/rbf/ in the future.

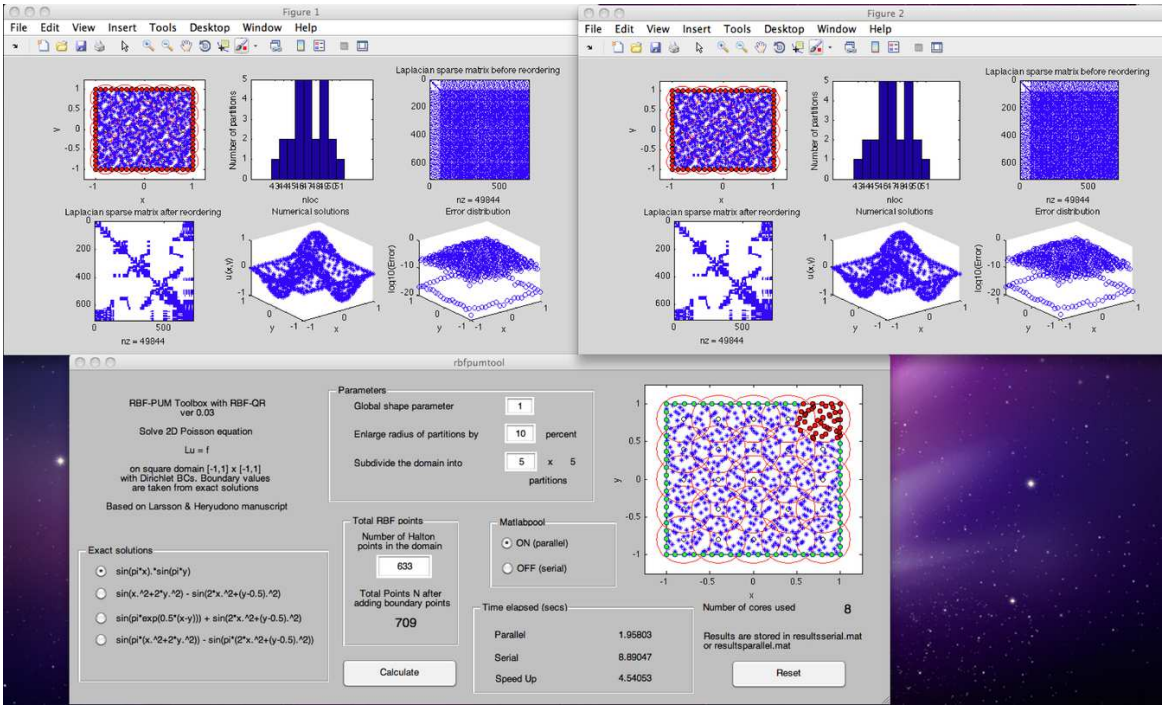

Figure 21: Version 0.03 of our MATLAB Toolbox called RBF-PDETool

9 Conclusion

Researchers who are working in the field of numerical methods for PDEs may find this project interesting. Indeed, the stable RBF-QR algorithm for generating differentiation matrices is the bread and butter of all our numerical experiments using RBF-FD, RBF-PU, and FEM-RBF hybrid on irregular geometry. RBF-QR algorithm provides highly accurate approximation in regions where solutions are smooth. Furthermore, by using either RBF-FD or RBF-PU as a standalone solver for smooth problems, we can now numerically observe algebraic and spectral convergence of the RBF collocation method. In the past, these convergence trends were hindered by ill-conditioning issues.

However, there are still many things unexplored in this project that may lead to new findings and directions. Our on-going projects include:

1. Finding better coupling techniques among partitions that will lead to efficient iterative solver.
2. Experimenting with multiple boundary conditions.
3. Testing the FEM-RBF with non-smooth solutions in the FEM regions.
4. 3D FEM-RBF using COMSOL FEM modules.
5. Eigenvalue stability for time-dependent problems.
6. Theory and error estimates.

At last, this report may look a bit like our day to day practical guide research diary without detailed mathematics. We like to make it that way such that researchers and students who are new to RBF can easily experiment with this new hybrid method. We provide the one dimensional FEM-RBF MATLAB code and we hope that readers (or their students) may try it. Other codes are available upon requests and we will include them in our future publications or on our website to support reproducible research. We hope that researchers and students who read this report may find this project useful and will eventually become our future collaborators!

10 MATLAB Codes

Program 1: femrbf1d.m

```
N = 81; % Total number of points
a = -1; c = 1; % Interval end points
Nf = 28; % Number of FE points
jiggleRBFpts = true; % Set to true to randomly jiggle RBF points
No = 3; % Number of FE point taken as RBF points (counting exclude x=b)
nloc = 11; % RBF-FD stencil size

x = linspace(a,c,N).'; % Generate equally-spaced points [a,c]
h = (c-a)/(N-1); % spacing.
ep = 0.075/h; % RBF shape parameter.
b = x(Nf); % point at interface.
if jiggleRBFpts, x(Nf+1:end) = x(Nf+1:end) + 0.5*h*(2*rand(N-Nf,1)-1); end

% FEM Stiffness Matrix
e = ones(Nf-1,1); K1 = spdiags([-e 2*e -e], -1:1, Nf-1, Nf-1);
K1(end,end) = 1; % Correct due to half hat function on the right.

% Generate RBF-FD Differentiation matrices
xrbf = x(Nf-No:end); xrbf = xrbf(:); Nr = length(xrbf);
[D1,D2]=deal(spalloc(Nr,Nr,nloc*Nr));
for i=1:Nr
    dx = xrbf(i) - xrbf.'; r2 = dx.^2;
    [sortr2,idx] = sort(r2); idx = idx(1:nloc);
    r2 = r2(idx); dx = dx(idx);
    dxmat = bsxfun(@minus,xrbf(idx),xrbf(idx).');
    A = 1./sqrt(1 + (ep*dxmat).^2); % RBF Interpolation matrix (IMQ-RBF)
    D1(idx,i) = (-ep^2*dx.*A(1,:).^3)/A; % 1st Deriv matrix
    D2(idx,i) = (ep^2*(-1+2*(ep*dx).^2).*A(1,:).^5)/A; % 2nd Deriv matrix
end
D1 = D1.'; D2 = D2.'; D2 = -D2;

% Assemble FEM+RBF matrices
A = blkdiag(K1,D2((No+2):Nr-1,(No+2):Nr-1));
A(Nf-1,Nf-(No+1):end) = A(Nf-1,Nf-(No+1):end)-h*D1(No+1,1:Nr-1);
A(Nf:end,Nf-(No+1):Nf-1) = D2((No+2):Nr-1,1:(No+1));

% FEM RHS
xfem = x(1:Nf); Ff = zeros(Nf,1);
Ff(1) = quad(@(y)Fint(y,xfem,1),xfem(1),xfem(2));
Ff(Nf) = quad(@(y)Fint(y,xfem,Nf),xfem(Nf-1),xfem(Nf));
for i=2:Nf-1
    Ff(i) =  quad(@(y)Fint(y,xfem,i),xfem(i-1),xfem(i+1));
end
Ff = h*Ff(2:end); Ff(1) = Ff(1) + uexact(xfem(1)); % add left BC
Ff(end) = Ff(end) + h*D1(No+1,Nr)*uexact(xrbf(Nr));
% RBF RHS
Fr = f(xrbf((No+2):Nr-1)) - D2((No+2):Nr-1,Nr)*uexact(xrbf(Nr));

% Combine RHS and solve
F = [Ff;Fr]; u = A\F;

% FEM Solution
uf = zeros(Nf,1); uf(1) = uexact(xfem(1)); uf(2:Nf) = u(1:Nf-1);
% RBF Solution
ur = zeros(Nr,1); ur(1:(No+1)) = u(Nf-(No+1):Nf-1);
ur((No+2):Nr-1) = u(Nf:end); ur(Nr) = uexact(xrbf(Nr));

% Plot system matrix
figure('Position',[100 100 1000 300])
subplot(1,3,1); spy(A)
% Plot FE and RBF solutions
subplot(1,3,2)
plot(xfem,uf,'-*k',xfem,uexact(xfem),'--r'); hold on;
plot(xrbf,ur,'-ob',xrbf,uexact(xrbf),'--r')
plot(x,min([uf;ur])*ones(N,1)-0.1,'.','markerfacecolor','k')
xlim([a-0.01 c+0.01]);xlabel('x'); ylabel('u(x)')
% Plot error distribution
subplot(1,3,3)
semilogy(xfem,abs(uf-uexact(xfem)),'-*k'); hold on
semilogy(xrbf,abs(ur-uexact(xrbf)),'-ob'); grid on;
xlim([a-0.01 c+0.01]);xlabel('x'); ylabel('||.||_{\infty}')
```

11 MATLAB Codes

Program 2: subroutines needed by femrbf1d.m

```
function y = uexact(x)
y = exp(-cos(3*pi*(x-0.5))) - 2*x - sin(pi*(x+0.5));

function ff = f(x)
ff = (-pi^2)*(cos(pi*x) + 4.5*exp(sin(3*pi*x)).*(1 + cos(6*pi*x) - 2*sin(3*pi*x)));

function yy = Hat1D(xx,x,i)
    % Equally spaced x
    N = length(x);
    yy = zeros(size(xx));

    switch i
        case 1
            flagrig = xx >= x(1) & xx < x(2);
            yy(flagrig) = 1 - (xx(flagrig)-x(1))./(x(2)-x(1));
        case N
            flaglef = xx <= x(N) & xx > x(N-1);
            yy(flaglef) = 1 + (xx(flaglef)-x(N))./(x(N)-x(N-1));
        otherwise
            flagrig = xx >= x(i) & xx < x(i+1);
            yy(flagrig) = 1 - (xx(flagrig)-x(i))./(x(i+1)-x(i));
            flaglef = xx <= x(i) & xx > x(i-1);
            yy(flaglef) = 1 + (xx(flaglef)-x(i))./(x(i)-x(i-1));
    end

function F = Fint(xx,x,i)
    F = f(xx).*Hat1D(xx,x,i);
```

Bibliography

[1] I. Babuška and J.M. Melenk. The partition of unity method. *Internat. J. Numer. Methods Engrg.*, 40(4):727–758, 1997. ISSN 0029-5981. 5.2

[2] J.P. Boyd. *Chebyshev and Fourier spectral methods*. Dover Publications Inc., Mineola, NY, second edition, 2001. ISBN 0-486-41183-4. 4

[3] J.P. Boyd and K. Gildersleeve. Numerical experiments on the condition number of the interpolation matrices for radial basis functions. *Appl. Numer. Math.*, 61(4):443–459, 2011. ISSN 0168-9274. 4

[4] M. D. Buhmann. *Radial basis functions: theory and implementations*, volume 12 of *Cambridge Monographs on Applied and Computational Mathematics*. Cambridge University Press, Cambridge, 2003. ISBN 0-521-63338-9. 2, 4

[5] C. Canuto, M.Y. Hussaini, A. Quarteroni, and T.A. Zang. *Spectral methods in fluid dynamics*. Springer Series in Computational Physics. Springer-Verlag, New York, 1988. ISBN 0-387-17371-4. 4

[6] A.H.-D. Cheng, M.A. Golberg, E.J. Kansa, and G. Zammito. Exponential convergence and *h-c* multiquadric collocation method for partial differential equations. *Numer. Methods Partial Differential Equations*, 19(5):571–594, 2003. ISSN 0749-159X. 2

[7] T. Driscoll and B. Fornberg. Interpolation in the limit of increasingly flat radial basis functions. *Computers and Mathematics with Applications*, 43:413–422, 2002. 5.1.2, 6.2

[8] T. A. Driscoll and A. Heryudono. Adaptive residual subsampling methods for radial basis function interpolation and collocation problems. *Computers and Mathematics with Applications*, 53:927–939, 2007. http://www.mathworks.com/matlabcentral/fileexchange/authors/23817. 2

[9] G.E. Fasshauer. RBF collocation methods as pseudospectral methods. In *Boundary elements XXVII*, volume 39 of *WIT Trans. Model. Simul.*, pages 47–56. WIT Press, Southampton, 2005. 4

[10] G.E. Fasshauer. *Meshfree approximation methods with MATLAB*, volume 6 of *Interdisciplinary Mathematical Sciences*. World Scientific Publishing Co. Pte. Ltd., Hackensack, NJ, 2007. ISBN 978-981-270-634-8; 981-270-634-8. With 1 CD-ROM (Windows, Macintosh and UNIX). 2, 4

[11] A.I. Fedoseyev, M.J. Friedman, and E.J. Kansa. Improved multiquadric method for elliptic partial differential equations via PDE collocation on the boundary. *Comput. Math. Appl.*, 43(3-5):439–455, 2002. ISSN 0898-1221. Radial basis functions and partial differential equations. 2

[12] B. Fornberg. *A practical guide to pseudospectral methods*, volume 1 of *Cambridge Monographs on Applied and Computational Mathematics*. Cambridge University Press, Cambridge, 1996. ISBN 0-521-49582-2. 4

[13] B. Fornberg and E. Lehto. Stabilization of RBF-generated finite difference methods for convective PDEs. *J. Comput. Phys.*, 230(6):2270–2285, 2011. ISSN 0021-9991. 5

[14] B. Fornberg and C. Piret. A stable algorithm for flat radial basis functions on a sphere. *SIAM J. Sci. Comput.*, 30 (1):60–80, 2007/08. ISSN 1064-8275. 4

[15] B. Fornberg, G. Wright, and E. Larsson. Some observations regarding interpolants in the limit of flat radial basis functions. *Comput. Math. Appl.*, 47(1):37–55, 2004. ISSN 0898-1221. 5.1.2, 6.2

[16] B. Fornberg, N. Flyer, and J.M. Russell. Comparisons between pseudospectral and radial basis function derivative approximations. *IMA J. Numer. Anal.*, 30(1):149–172, 2010. ISSN 0272-4979. 5.1.2

[17] B. Fornberg, E. Larsson, and E. Flyer. Stable computations with Gaussian radial basis functions. *SIAM J. Sci. Comput.*, 33(2):869–892, 2011. ISSN 1064-8275. 4, 5.1.2, 6.2

[18] A.R.H. Heryudono. *Adaptive radial basis function methods for the numerical solution of partial differential equations, with application to the simulation of the human tear film*. ProQuest LLC, Ann Arbor, MI, 2008. ISBN 978-0549-81391-0. Thesis (Ph.D.)–University of Delaware. 2

[19] A.R.H. Heryudono and T.A. Driscoll. Radial basis function interpolation on irregular domain through conformal transplantation. *J. Sci. Comput.*, 44(3):286–300, 2010. ISSN 0885-7474. 2

[20] J. Hesthaven, S. Gottlieb, and D. Gottlieb. *Spectral Methods for Time-Dependent Problems*. Cambridge University Press, 2007. ISBN 0521792118. 4

[21] E. J. Kansa. Multiquadrics - a scattered data approximation scheme with applications to computational fluid dynamics I: Surface approximations and partial derivative estimates. *Computers and Mathematics with Applications*, 19(8/9):127–145, 1990. 2

[22] 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. *Computers and Mathematics with Applications*, 19(8/9):147–161, 1990. 2

[23] 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(7-8):123–137, 2000. ISSN 0898-1221. 2

[24] E.J. Kansa and Y.C. Hon, editors. *Radial basis functions and partial differential equations*. Elsevier Science B.V., Amsterdam, 2002. Comput. Math. Appl. **4**3 (2002), no. 3-5. 2

[25] E. Larsson and B. Fornberg. A numerical study of some radial basis function based solution methods for elliptic PDEs. *Comput. Math. Appl.*, 46(5-6):891–902, 2003. ISSN 0898-1221. 2

[26] E. Larsson and B. Fornberg. Theoretical and computational aspects of multivariate interpolation with increasingly flat radial basis functions. *Comput. Math. Appl.*, 49(1):103–130, 2005. ISSN 0898-1221. 5.1.2

[27] C. Micchelli. Interpolation of scattered data: Distance matrices and conditionally positive definite functions. *Constructive Approximation*, 2:11–22, 1986. 4

[28] P-O. Persson and G. Strang. A simple mesh generator in Matlab. *SIAM Rev.*, 46(2):329–345 (electronic), 2004. ISSN 0036-1445. 6.1, 6.3

[29] R.B. Platte. How fast do radial basis function interpolants of analytic functions converge? *IMA J. Numer. Anal.*, 31(4), 2011. 5.1.2

[30] R.B. Platte and T.A. Driscoll. Polynomials and potential theory for Gaussian radial basis function interpolation. *SIAM J. Numer. Anal.*, 43(2):750–766 (electronic), 2005. ISSN 0036-1429. 5.1.2, 6.2

[31] R. Schaback. Error estimates and condition numbers for radial basis function interpolation. *Advances in Computational Mathematics*, 3:251–264, 1995. 4

[32] R. Schaback. Limit problems for interpolation by analytic radial basis functions. *Journal of Computational and Applied Mathematics*, 212:127–149, 2008. 5.1.2, 6.2

[33] D. Shepard. A two-dimensional interpolation function for irregularly-spaced data. In *Proceedings of the 1968 23rd ACM national conference*, ACM '68, pages 517–524, New York, NY, USA, 1968. ACM. 6.3

[34] I.H. Sloan and R.S. Womersley. The search for good polynomial interpolation points on the sphere. In *Numerical analysis 1999 (Dundee)*, volume 420 of *Chapman & Hall/CRC Res. Notes Math.*, pages 211–229. Chapman & Hall/CRC, Boca Raton, FL, 2000. 6.3.2

[35] I.H. Sloan and R.S. Womersley. Extremal systems of points and numerical integration on the sphere. *Adv. Comput. Math.*, 21(1-2):107–125, 2004. ISSN 1019-7168. 6.3.2

[36] 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(1):68–79, 2003. ISSN 0178-7675. 5

[37] L.N. Trefethen. *Spectral methods in MATLAB*, volume 10 of *Software, Environments, and Tools*. Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, 2000. ISBN 0-89871-465-6. 4

[38] H. Wendland. Piecewise polynomial, positive definite and compactly supported radial functions of minimal degree. *Adv. Comput. Math.*, 4(4):389–396, 1995. ISSN 1019-7168. 6.3

[39] H. Wendland. *Scattered data approximation*, volume 17 of *Cambridge Monographs on Applied and Computational Mathematics*. Cambridge University Press, Cambridge, 2005. ISBN 978-0521-84335-5; 0-521-84335-9. 2, 4

[40] G.B. Wright and B. Fornberg. Scattered node compact finite difference-type formulas generated from radial basis functions. *J. Comput. Phys.*, 212(1):99–123, 2006. ISSN 0021-9991. 5

[41] J. Yoon. Spectral approximation orders of radial basis function interpolation on the Sobolev space. *SIAM J. Math. Anal.*, 33(4):946–958 (electronic), 2001. ISSN 0036-1410. 5.1.2