J Sci Comput (2010) 44: 286–300
DOI 10.1007/s10915-010-9380-3

Radial Basis Function Interpolation on Irregular Domain through Conformal Transplantation

Alfa R.H. Heryudono · Tobin A. Driscoll

Received: 5 January 2010 / Accepted: 17 May 2010 / Published online: 3 June 2010

Abstract In this paper, Radial Basis Function (RBF) method for interpolating two dimensional functions with localized features defined on irregular domain is presented. RBF points located inside the domain and on its boundary are chosen such that they are the image of conformally mapped points on concentric circles on a unit disk. On the disk, a fast RBF solver to compute RBF coefficients developed by Karageorghis et al. (Appl. Numer. Math. 57(3):304–319, 2007) is used. Approximation values at desired points in the domain can be computed through the process of conformal transplantation. Some numerical experiments are given in a style of a tutorial and MATLAB code that solves RBF coefficients using up to 100,000 RBF points is provided.

Keywords Radial basis functions · Schwarz-Christoffel mapping · Meshfree · Interpolation

1 Introduction

Methods based on Radial Basis Functions (RBF) have been widely used for scattered data interpolation in higher dimension. In order to form the RBF interpolant, one requires only a set of nodes called *centers*, which define the basis functions, and its corresponding interpolation data values. These minimal requirements carry over to higher dimensions. The basis functions are radially symmetric with respect to its centers and their shapes remain invariant in all dimensions. The centers themselves do not have to be specially distributed in such a way where interpolation data is difficult to obtain. Moreover, the accuracy of approximation provided by the interpolant can be adjusted using parameters known as *shape parameters*. More in-depth study about RBF methods can be found in [4, 12, 33] and references therein.

A.R.H. Heryudono (✉)
Department of Mathematics, University of Massachusetts Dartmouth, Dartmouth, MA 02747, USA
e-mail: aheryudono@umassd.edu

T.A. Driscoll
Department of Mathematical Sciences, University of Delaware, Newark, DE 19716, USA
e-mail: driscoll@math.udel.edu

The flexibilities of RBF method described above regarding its meshfree and/or mesh-tolerant nature and the availability of adjustable parameters make the RBF interpolant a viable choice for an approximant for solutions of partial differential equations on irregular geometry [23, 25]. The freedom of placing nodes wherever needed is a great deal if the function to be interpolated has some localized features. In that case, nodes can be denser around high activity areas and coarser in other region. Indeed, study about adaptive process for automatically refining/coarsening RBF centers based on the profile of the functions/solutions is becoming an active research area [10, 21, 22, 29].

On the other hand, those flexibilities combined with several drawbacks of the RBF method can make the process of selecting the best centers' distributions and choosing the best parameters a frustrating task. Moreover, as the number of centers grows, the method needs to solve $Ax = b$ with relatively large and dense matrix A. Accuracy of RBF approximations is plagued by severe ill-conditioning of the interpolation matrix that makes achieving higher accuracy difficult. This instability behavior leads to the famous classical accuracy and stability trade-off: RBF uncertainty principle by Schaback [30].

Either one goes for a small error and gets a bad sensitivity, or one wants a stable algorithm and has to take a comparably large error.

Several techniques have been devised to tackle this issue such as complex contour integration technique [17], RBF-QR [15, 16], and RBF Matrix Decomposition [24]. If one wishes to use an iterative method, ill-conditioning can also create a serious convergence issue. In such cases good preconditioners are needed [3, 13]. In addition to instability from linear algebra, node and center locations also play a crucial role. The classical problem of interpolation stability is usually measured by Lebesgue constants and manifested through the Runge phenomenon. In one dimensional case, the interpolant generated by Gaussian RBFs with equally-spaced centers can be transformed into a polynomial [27]. Just as in polynomial potential theory, as the number of centers grows to infinity, an interpolated function must be analytic in a complex domain larger than the interval of approximation, unless a special node density is used. This density clusters nodes toward the ends of the interval in order to avoid Runge oscillations. The existence and construction of stable node sets in higher dimensions and general geometries, in particular with regards to proper clustering near the boundary, remains a very challenging problem today.

In this article, we focus on using the RBF method for interpolating two dimensional functions with some degree of localization defined on irregular domains. The objective of this paper is to combine two techniques, an FFT-based fast RBF solver by Karageorghis et al. [24] and conformal mapping to handle the irregular geometry. Although all functions used in our numerical experiments can be extended outside the domain, we only use centers inside the domain and on the boundary. The irregular domains are currently chosen to be simply connected polygons. RBF centers are not chosen randomly. Instead, they are the images of conformally mapped points on concentric circles on a unit disk. On the disk, a fast RBF solver [24] that takes advantage of block circulant structures of the interpolation matrix is available. Once the RBF interpolant on the disk is obtained, interpolants between the disk and the irregular domain can be communicated through the process of conformal transplantation. The idea of conformal transplantation between two domains is not new. It has been used in Chebyshev pseudospectral methods [14, 20, 32] as well. One of striking applications is the eigenvalue problems on fractal regions [2].

2 Radial Basis Functions in Two Dimensional Case

We briefly describe the method of interpolation by RBF. For further study, one should consult the three references we mentioned in the beginning paragraph of the previous section. Given a set of N distinct *centers* $(x_1^c, y_1^c), \dots, (x_N^c, y_N^c)$ in $\mathbb{R}^2$, the RBF interpolant takes the form

$$s(x, y) = \sum_{j=1}^{N} \lambda_j \phi^j(x, y), \tag{1}$$

where $\phi^j(x, y)$ is a radial basis function centered at (x_j^c, y_j^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 due to (under certain conditions [5, 7, 26, 34]) their high-order (even spectrally) accurate approximations. As an example, the Inverse Multiquadric (IMQ)

$$\phi^j(x, y) = \frac{1}{\sqrt{1 + \varepsilon^2 r_j^2(x, y)}}, \tag{2}$$

represents this category of RBF where $r_j^2(x, y) = (x - x_j^c)^2 + (y - y_j^c)^2$ is the square of Euclidean distance from the center (x_j^c, y_j^c). The coefficients λ are determined by enforcing the interpolation condition

$$s(x_i, y_i) = F(x_i, y_i) \tag{3}$$

at a set of nodes that typically coincide with the centers, where $F(x, y)$ is the function to be interpolated. Enforcing the interpolation conditions at N centers results in solving a $N \times N$ linear system

$$A\lambda = \mathbf{F}. \tag{4}$$

The matrix A with entries

$$a_{ij} = \phi^j\left(x_i^c, y_i^c\right), \quad i, j = 1, \dots, N \tag{5}$$

is called the *interpolation matrix*.

An interesting connection exists between spectral that are based on polynomials and the non-polynomial RBF approximations. In the limit $\varepsilon \to 0$ the RBF interpolant is equivalent to the minimal-degree Lagrange interpolating polynomial [9]. In higher dimensions, the limit may not exist but when it does it is a low degree multivariate polynomial [31]. How fast RBF approximations converge is recently discussed in [26].

3 RBF Centers on Concentric Circles

Suppose we arrange RBF centers in a systematic way on the concentric circles

$$x_{i,j} = R_j \cos\left[\frac{2(i-1)\pi}{n} + \frac{2\beta_j \pi}{n}\right], \qquad y_{i,j} = R_j \sin\left[\frac{2(i-1)\pi}{n} + \frac{2\beta_j \pi}{n}\right], \tag{6}$$

where $i = 1, \dots, n$, $j = 1, \dots, m$ with $R_1 < R_2 < \dots < R_m$ and $0 \le \beta_j \le 1$. The $N \times N$ RBF interpolation matrix A in (4), where $N = nm$, provides a nice structure in the form of block circulant submatrices

$$A = \begin{bmatrix} A_{11} & A_{12} & \cdots & A_{1m} \\ A_{21} & A_{22} & \cdots & A_{2m} \\ \vdots & \vdots & \ddots & \vdots \\ A_{m1} & A_{m2} & \cdots & A_{mm} \end{bmatrix}. \tag{7}$$

Following the convention in (5), the entries of the $n \times n$ circulant submatrix A_{kl} are formed by using the RBFs centered at points on the circle with radius R_l and evaluated at points on the circle with radius R_k.

In this case, RBF coefficients, which are the solution of (4), can be efficiently computed by taking advantage of circulant matrix decomposition via Fast Fourier Transform (FFT). Regarding this method, one should consult the paper by Karageorghis et al. [24]. We briefly describe their method.

A circulant matrix $C = \text{circ}(c_1, \dots, c_n)$ has a nice property such that it can be decomposed as $C = U^*DU$, where $D = \text{diag}(d_1, \dots, d_n)$ with $d_j = \sum_{k=1}^n c_k \omega^{(k-1)(j-1)}$, and U (Fourier matrix) is the conjugate of an $n \times n$ matrix

$$U^* = \frac{1}{\sqrt{n}} \begin{bmatrix} 1 & 1 & 1 & \cdots & 1 \\ 1 & \omega & \omega^2 & \cdots & \omega^{n-1} \\ 1 & \omega^2 & \omega^4 & \cdots & \omega^{2(n-1)} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & \omega^{n-1} & \omega^{2(n-1)} & \cdots & \omega^{(n-1)(n-1)} \end{bmatrix}, \quad \text{where } \omega = e^{2\pi i/n}.$$

Hence, the interpolation matrix (7) that contains block circulant structures can be decomposed as

$$A = \begin{bmatrix} U^* & 0 & \cdots & 0 \\ 0 & U^* & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & U^* \end{bmatrix} \begin{bmatrix} D_{11} & D_{12} & \cdots & D_{1m} \\ D_{21} & D_{22} & \cdots & D_{2m} \\ \vdots & \vdots & \ddots & \vdots \\ D_{m1} & D_{m2} & \cdots & D_{mm} \end{bmatrix} \begin{bmatrix} U & 0 & \cdots & 0 \\ 0 & U & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & U \end{bmatrix}, \tag{8}$$

where D_{kl} in (8) and A_{kl} in (7) satisfy $A_{kl} = U^* D_{kl} U$.

With I_m is an $m \times m$ indentity matrix, and $\otimes$ defines a Kronecker product operations of two matrices, RBF coefficients in (4) can be found (or solved for) as presented in Algorithm 1. The algorithm can be coded in MATLAB (provided in Appendix) in under 50 lines with the name of **diskRBF.m**. The MATLAB code **diskRBF.m** uses $R_j = \frac{j}{m} r_{\max}$, $j = 1, \dots, m$ with $r_{\max} = 1$, and special rotation angles $\beta_j = 0$ for j odd and 0.5 for j even. The code can take up to 100,000 RBF centers on concentric circles and takes only couple of seconds to run.

Algorithm 1

```
1. Apply FFT to compute $\hat{\mathbf{F}} = (I_m \otimes U)\mathbf{F}$ at a cost of $O(mn \log n)$.
2. Apply FFT to compute diagonal elements of submatrices $D$'s at a cost of $O(m^2 n \log n)$.
3. Transform $D$ to block diagonal matrix with approximate minimum degree (amd) permu-
   tation [1].
4. Solve $m$ linear systems of order $n$ to compute $\hat{\lambda}$ at a cost of $O(mn^2)$.
5. Apply inverse FFT to compute $\lambda = (I_m \otimes U^*)\hat{\lambda}$ at a cost of $O(mn \log n)$.
```

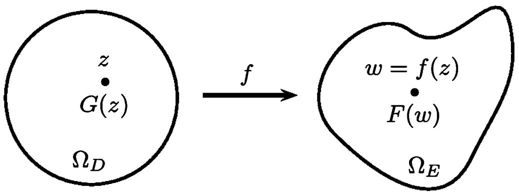


Fig. 1 Conformal map f from a disk to a simply connected domain on a complex plane

4 Conformal Transplantation

We briefly describe the process of conformal transplantation between two domains on a complex plane. For in-depth study, one can consult [19]. Let Ω_D be a region of the z-plane ($z = x + iy$) and let $f : z \to w = f(z)$ be a holomorphic function defining a one-to-one mapping of Ω_D onto a region Ω_E of the w-plane ($w = u + iv$). An illustration of a conformal map f from a disk to a simply connected domain can be seen in Fig. 1.

Let a real-valued function

$$G : (x, y) \to G(x, y) = G(z) \tag{9}$$

be defined in Ω_D. We can then define in Ω_E a function F as follows: for any $w \in \Omega_E$

$$F(w) := G(f^{-1}(w)) = G(x(u, v), y(u, v)). \tag{10}$$

Thus the value of F at w is equal to the value of G at the preimage of w under the map f. The function F is called the *conformal transplant* of G under the mapping f and its process of reconstruction is the *conformal transplantation*.

5 Statement of Our Problem

Our goal is to conformally transplant the RBF interpolant $S(x, y)$ on irregular domain. The steps are described in Algorithm 2 and its illustration is shown in Fig. 2. In all our cases, we choose simply connected polygons as our irregular domains. The special case of conformal mapping from a unit disk to a polygon is widely known as Schwarz-Christoffel (SC) mapping.

Algorithm 2

```
1. Find a conformal map from the unit disk to the irregular domain.
2. Find RBF centers on the domain. The centers are images (under map in step 1) of centers
   on the unit disk.
3. Compute function values at centers on irregular domain.
4. Solve RBF coefficients to form RBF interpolant $S$ on a disk using function values in
   step 3.
5. Pick points on the domain at which values to be approximated.
6. Find the preimage of points in step 5 on the unit disk.
7. Evaluate interpolant $S$ on the preimage points.
```

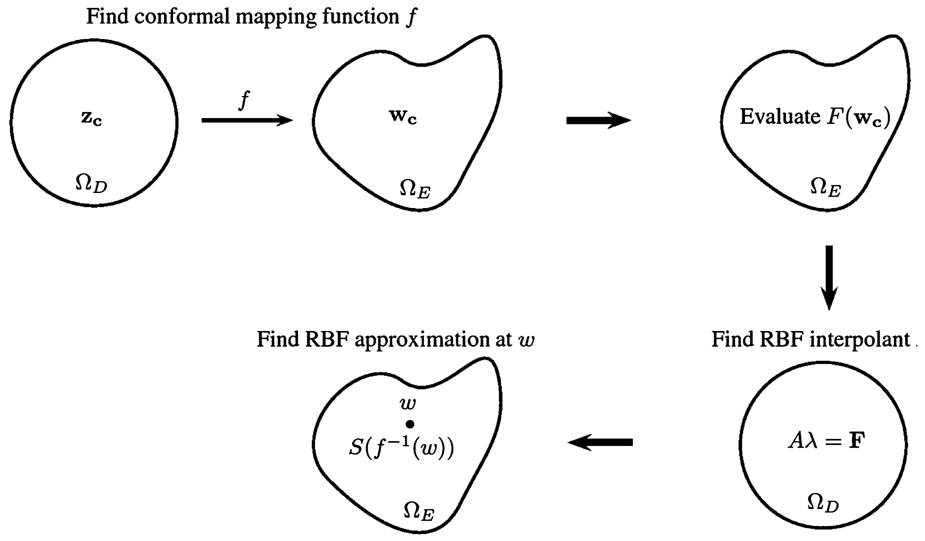


Fig. 2 The process of conformally transplant RBF interpolant S on irregular domain

Theorem 5.1 *Let P be interior of a polygon Γ having vertices $w_1, \dots, w_n$ and interior angles $\alpha_1\pi, \dots, \alpha_n\pi$ in counterclockwise order. Let f be any conformal map from the unit disk Ω_D to P. Then*

$$f(z) = A + C \int^z \prod_{k=1}^{n} \left(1 - \frac{\xi}{z_k}\right)^{\alpha_k - 1} d\xi$$

for some complex constants A and C, where $w_k = f(z_k)$ for $k = 1, \dots, n$.

For in-depth study regarding SC mapping (e.g. the proof of Theorem 5.1), one should consult [11]. In all our numerical experiments, we utilize Driscoll's freely available SC MATLAB toolbox [8].

6 Numerical Experiments

We present our results in a style of a tutorial in the spirit of promoting reproducible research. All of our computations are run using MATLAB 7 on a MacBook Pro with 4 GB of RAM. Note that the directory containing SC toolbox files should be added to the MATLAB path. The RBF coefficients can in most cases be computed in a few seconds. Once RBF coefficients are computed, evaluations of RBF approximations at desired points can be done using matrix vector product code **diskRBFinterp.m** available in the Appendix. The RBF evaluation code **diskRBFinterp.m**, though fast enough for our numerical experiments, is not efficient. In the future, we will implement the fast RBF matrix-vector product following the work in [28]. In all our numerical experiments in MATLAB, RBFs are defined such that we input square of Euclidean distance. For simplicity, we choose $n = m$ and n, m will be increased to get the profile of error convergence with respect to the exact solutions until no more accuracy can be gained due to rounding error.

6.1 Experiment 1

Our first experiment is to interpolate a function defined by

$$F(x,y)=e^{-81(x^2+y^2)/4} \tag{11}$$

on a polygonal domain with vertices $\{(0,1), (-1,1), (-1,-1), (1.5,-1), (1.5,0), (1,0)\}$ arranged in counterclockwise order. First, we define the function to be interpolated and create an SC map from a unit disk centered at $(0,0)$ to the polygon.

```
>> F = @(x,y) exp((-81/4)*(x.^2 + y.^2));
>> p = polygon([1i,-1+1i,-1-1i,1.5-1i,1.5,1]);
>> options = scmapopt('Tolerance',1e-14);
>> f = diskmap(p,options);
>> f = center(f,0)

  diskmap object:

      vertex          alpha          prevertex            arg/pi
-------------------------------------------------------------------
 0.00000+1.00000i   0.75000   -0.69538+0.71864i   0.744764459558
-1.00000+1.00000i   0.50000   -0.97551+0.21995i   0.929411276130
-1.00000-1.00000i   0.50000   -0.32539-0.94558i   1.394503974180
 1.50000-1.00000i   0.50000    0.96672-0.25584i   1.917646561233
 1.50000+0.00000i   0.50000    0.98953-0.14429i   1.953909049502
 1.00000+0.00000i   1.25000    1.00000+0.00000i   2.000000000000

  c = 0.87630288 - 0.4189778i
  Conformal center at 0.0000 + 0.0000i

  Apparent accuracy is 2.14e-13

>> plot(f);
```

Left image of Fig. 3 shows the shape of the domain. Once the SC map from the unit disk to the polygon is obtained, we are ready to solve RBF coefficients. As an example, we choose $m=10$ and $n=10$ ($N=100$ centers in total) to create RBF centers on concentric circles on a unit disk as in equation. IMQ-RBF (2) with shape parameter $\varepsilon=1.95$ is chosen.

```
>> m = 10; n = 10;
>> ep = 1.95;
>> phi = @(r2) 1./sqrt(ep^2*r2+1);
   %r2 is the square of distance
>> rbfinfo = diskRBF(n,m,f,F,phi,ep);
RBF coefficients solved!
Elapsed time is 0.013026 seconds.
```

The plot of the image of RBF centers on the polygon can be seen in Fig. 3 by executing:

```
>> wc = f(rbfinfo.centers);
>> figure; plot(f); hold on;
```

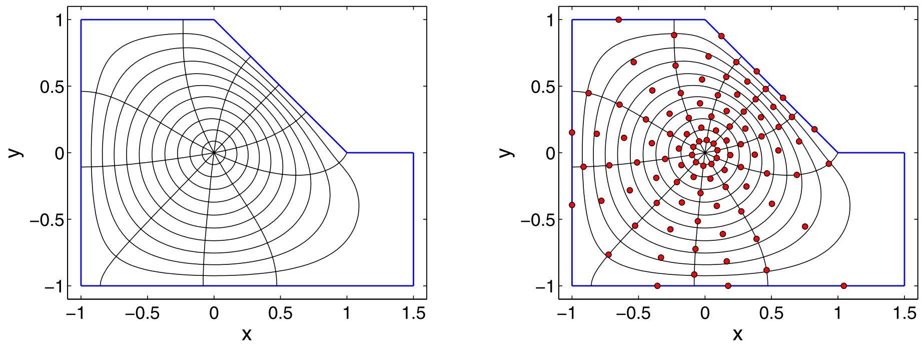


Fig. 3 *Left*: Irregular domain for Experiment 1 and the images of conformally mapped *concentric circles*. *Right*: *Dots* are the images of RBF centers which are mapped from the disk to irregular geometry. The images of *concentric circles* from the built-in command **plot(f)** do not necessarily coincide with *concentric circles* where RBF centers are located on the unit disk since different radiuses are used

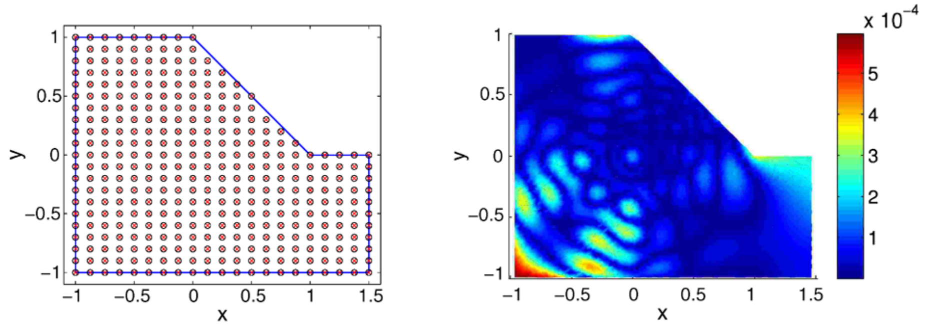


Fig. 4 *Left*: Locations where function approximations are sought. *Right*: Error distributions compared to exact function values inside the domain. As expected, error is typically higher around corners and/or near boundaries

```
>> plot(wc,'ok','MarkerFaceColor','r','MarkerSize',5)
```

The next step is to pick points on the polygon whose values need to be approximated. As an example, we choose locations of points as in Fig. 4. The approximated values will be compared with the exact solutions.

```
>> M=21;
>> xx = linspace(-1,1.5,M);
>> yy = linspace(-1,1,M);
>> [x,y]=meshgrid(xx,yy); x = x(:); y = y(:);
>> in = inpolygon(x,y,real([p.vertex;p.vertex(1)]),...
        imag([p.vertex;p.vertex(1)]));
>> x = x(in); y=y(in);
>> figure; plot(p); hold on; plot(x,y,'xr')
>> ff = diskRBFinterp(x,y,rbfinfo);
>> maxerror = max(abs(ff-F(x,y)))

maxerror = 6.039988126662153e-04
```

Table 1 IMQ ($\varepsilon = 1.95$): Error convergence values for interpolating function (11). Plot of convergence trend can be seen in the *figure on the right*

m	n	$\lvert\cdot\rvert_\infty$	t_λ (s)
5	5	2.871228776707546e−01	0.006550
10	10	6.039988126554461e−04	0.003473
20	20	1.433523070098297e−06	0.014644
30	30	4.835310780706259e−08	0.036880
40	40	2.998488463928120e−09	0.074084
50	50	7.806039212402224e−10	0.142204
60	60	8.479065291714190e−10	0.233762

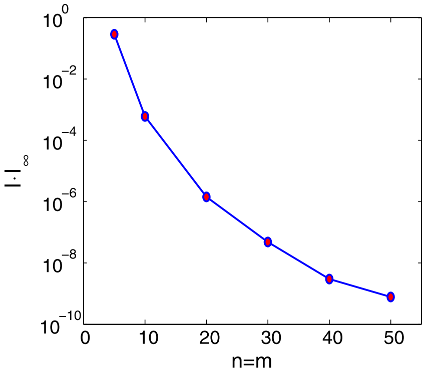
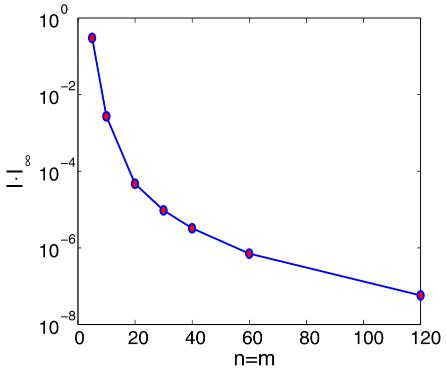


Table 2 r^5: Error convergence values for interpolating function (11). Plot of convergence trend can be seen in the *figure on the right*

m	n	$\lvert\cdot\rvert_\infty$	t_λ (s)
5	5	3.018826081384782e−01	0.001434
10	10	2.707714803362737e−03	0.003011
20	20	4.720376693800237e−05	0.015444
30	30	9.583004695661574e−06	0.038313
40	40	3.267962317004291e−06	0.076959
60	60	7.121329571527065e−07	0.247550
120	120	5.782109557177031e−08	1.790299
200	200	1.016468711897323e−08	8.402037



Results using more evaluation points (say, 1345 points) in the domain with varying $n = m$ for IMQ case and r^5 can be seen in Tables 1 and 2 respectively. Note that t_λ is the execution time for solving RBF coefficients measured using the MATLAB's stop-watch command: **tic toc**.

6.2 Experiment 2: Shifting Strategy

Suppose we want to interpolate

$$F(x, y) = e^{-81((x-1)^2+(y+0.5)^2)/4}, \tag{12}$$

and reuse the SC map in Experiment 1. Function (12) is the same function as in Experiment 1 except the region of high activity is shifted to $(1, -0.5)$. We proceed by redefining the function $F(x, y)$ with our new function.

```
>> F = @(x,y) exp((-81/4)*((x-1).^2 + (y+0.5).^2));
>> rbfinfo = diskRBF(n,m,f,F,phi,ep);
>> ff = diskRBFinterp(x,y,rbfinfo);
```

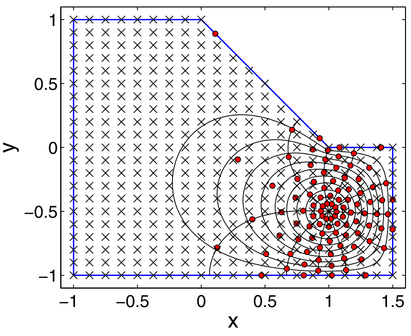


Fig. 5 RBF centers (*dots*) are shifted to be around the region of high activity. $\times$ are locations where function approximations are sought

```
>> maxerror = max(abs(ff-F(x,y)))
RBF coefficients solved!
Elapsed time is 0.003209 seconds.

maxerror = 7.635716900567395e-01
```

It turns out that the maximum error is rather poor. The reason why this gives a bad result is that our unit disk is centered at $(0,0)$. In order to regain the accuracy back, we shift the center of the unit disk to $(1,-0.5)$ where high activity occurs. The picture of the distribution of centers after shifting is shown in Fig. 5. The process of recomputing the conformal map by shifting the center and then regenerate the RBF interpolant are shown in the following commands:

```
>> f = center(f,1-0.5*1i);
>> rbfinfo = diskRBF(n,m,f,F,phi,ep);
>> ff = diskRBFinterp(x,y,rbfinfo);
>> maxerror = max(abs(ff-F(x,y)))
RBF coefficients solved!
Elapsed time is 0.003022 seconds.

maxerror = 9.746507145604117e-04
```

Note that the level of accuracy is back to similar result as in Experiment 1.

6.3 Experiment 3: Interpolation on a Narrow Rectangle

In this example, we want to interpolate the function

$$F(x,y)=\tanh(10y) \quad \text{on } [-0.3,0.3]\times[-1,1]. \tag{13}$$

Figure of the domain as well as RBF centers for $n=m=10$ can be seen in Fig. 6. The trend of error convergence can be seen in Table 3. In addition to the narrow region, there is also a region with steep gradient in the y direction that needs to be resolved. This example can usually be a challenge for RBF methods since the condition number $\kappa(A)$ can quickly grow as distances among centers become smaller [30].

A common way to delay the growth of $\kappa(A)$ is to use variable shape parameters chosen based on nearest neighbors' distances [10, 18]. In that case, larger shape parameters are

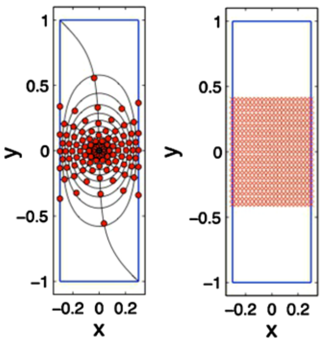


Fig. 6 *Left*: Narrow rectangle domain. RBF centers are distributed in the region of steep gradient. *Right*: Locations where function approximations are sought

Table 3 IMQ ($\varepsilon = 3.25$): Error convergence values for interpolating function (13) on narrow domain as in Fig. 6

m	n	$\|\cdot\|_\infty$	t_λ (s)
5	5	1.802765996014689e−01	0.001410
10	10	3.699315545079440e−02	0.003316
20	20	3.626362118354209e−03	0.014435
30	30	4.462425364284428e−04	0.036919
40	40	5.832149101614448e−05	0.080982
50	50	8.373242851966722e−06	0.152643
60	60	1.294515106553540e−06	0.256228
70	70	2.309790143595336e−07	0.376411
80	80	3.895641764728452e−08	0.548112
90	90	9.455389193835373e−08	0.756715

chosen for centers around high activity regions. With larger parameters, basis functions will show their "local" shapes. Away from high activity regions, smaller shape parameters are chosen such that basis functions will show their "global" property.

However, a constant shape parameter must be used with our method as the use of a variable shape parameter destroys the circulant structures of the interpolation matrix. Another alternative way is to normalize the distance norm based on the size of the strip using SVD [6]. We will take a look at this method in the future.

6.4 Experiment 4: Interpolation on a Curved Domain

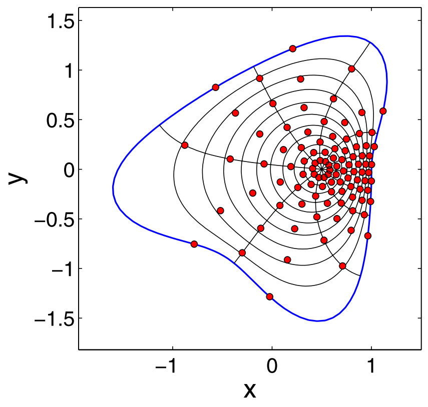
Our last experiment is to interpolate the function

$$F(x, y) = 1 - \tanh(5((x - 0.5)^2 + y^2)) \tag{14}$$

on the domain with parametric equation in polar coordinates given by

$$\frac{1}{r^2} = \left|\cos\left(\frac{3\theta}{4}\right)\right|^8 + \left|\sin\left(\frac{3\theta}{4}\right)\right|^3, \tag{15}$$

as seen in Fig. 7. We discretize the curved boundary into 126 equally-spaced points and connect them in counter clockwise way into a polygon. Once the vertices are known, the conformal map can be computed and the process follows the previous numerical experiments.



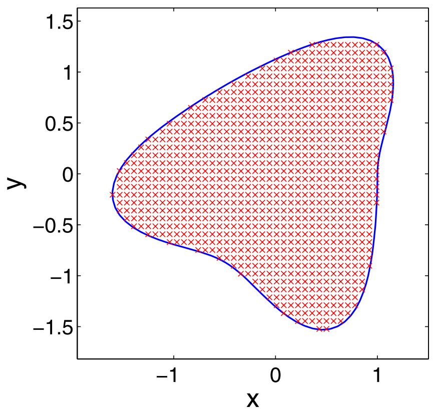
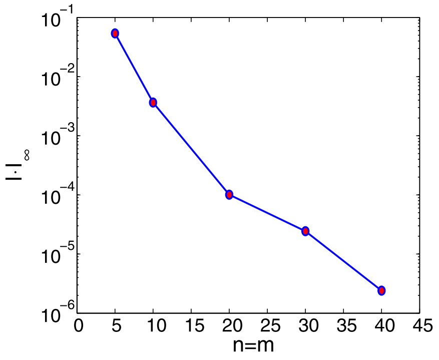


Fig. 7 (Color online) *Left*: Shape of the domain (15) and RBF centers in *red dots*. *Right*: Locations where approximation values are sought

Table 4 MQ ($\varepsilon = 3.25$): Error convergence values for interpolating function (14). Plot of convergence trend can be seen in the *figure on the right*

m	n	$\lvert\cdot\rvert_\infty$	t_λ (s)
5	5	5.394400318859172e−02	0.000951
10	10	3.654902479635402e−03	0.003009
20	20	1.009164752920899e−04	0.013974
30	30	2.435753546947694e−05	0.035894
40	40	2.407381991087476e−06	0.076573
50	50	6.860593755098066e−06	0.141736



The multiquadric (MQ) RBF

$$\phi^j(x, y) = \sqrt{1 + \varepsilon^2 r_j^2(x, y)}, \tag{16}$$

is used with $\varepsilon = 3.25$ and error convergence trend is shown in Table 4. We stop gaining more accuracy after $n = m = 40$. We believe that the ill-conditioning of block diagonal matrices (after amd ordering) in D in (8) is the source of this issue. It turns out that the condition numbers in those block matrices can be as high as 10^{14}. Finding a way to squeeze out the source of ill-conditioning in D will be left for future study.

7 Remark

We invite readers to try the tutorial for experimenting with different basis functions and shape parameters. Moreover, readers may also find it interesting to compare the calculation time for solving RBF coefficients with conventional method using the backslash command \

in MATLAB. The case with $n = m = 200$ ($N = 40{,}000$ centers) might be a good case to try.

With this method, it may seem that we defeat the purpose of RBF methods where points can be freely scattered on the irregular domain. However, experimenting with distribution of points on irregular geometry and tweaking shape parameters all the time are anything other than fun. This makes creating "black box" program based on RBF difficult.

In our future work, we will investigate the accuracy of RBF derivatives under the conformal map. Fast Gauss Transform method will also be implemented for rapid interpolant evaluations. Comparison with RBF-QR is currently under investigation. Lastly, we will investigate the idea of the RBFs methods based on partition unity of disks or methods of overlapping disks for the numerical solutions of PDEs.

Acknowledgements We want to thank Elisabeth Larsson for interesting discussions and Saeja Kim for careful reading of the manuscript. We acknowledge helpful comments made by two anonymous referees.

Appendix

MATLAB files: **diskRBF.m** and **diskRBFinterp.m**.

```
function ff = diskRBFinterp(x,y,rbfinfo)
warning off;
x = x(:); y = y(:); N = length(x); ff = zeros(N,1);
m = rbfinfo.m; n = rbfinfo.n; xc = real(rbfinfo.centers);
yc = imag(rbfinfo.centers); coeff = rbfinfo.coeff;
     ep = rbfinfo.ep;
phi = rbfinfo.phi; f = rbfinfo.f;

finv = inv(f); % Compute inverse sc map from the polygon
               % to unit disk.
z = finv(x+1i*y); x = real(z); y = imag(z);
for k=1:m
    rowidx = (k-1)*n+1:k*n;
    [xx,ww] = meshgrid(xc(rowidx),x);
    r2 = (ww - xx).^2;
    [xx,ww] = meshgrid(yc(rowidx),y);
    r2 = r2 + (ww - xx).^2;
    ff = ff + phi(r2)*coeff(rowidx);
end

function rbfinfo = diskRBF(n,m,f,F,phi,ep)

% Create RBF centers on concentric circles
rmax = 1; R = repmat(rmax*(1:m)'/m,1,n);
T = repmat((1:n)-1,m,1) + repmat(0.5*mod((2:m+1)',2),1,n);
xc = R.*cos(2*pi*T/n); yc = R.*sin(2*pi*T/n);
xc = xc'; yc = yc';

wc = f(xc+1i*yc); % Image of centers on the polygon

fc = F(real(wc),imag(wc));
```

```
[s1,s2,s3] = deal(zeros(m^2*n,1));
iter = 0; tic
for k=1:m
  rowidx=(k-1)*n+1:k*n;
    for j=k:m
       colidx=(j-1)*n+1:j*n;
       dx = xc(1,k) - xc(:,j); dy = yc(1,k) - yc(:,j);
       r2 = dx.^2 + dy.^2; fphi = phi(r2);

       s1(iter+1:iter+n) = rowidx; s2(iter+1:iter+n) = colidx;
       s3(iter+1:iter+n) = n*ifft(fphi); iter = iter + n;

       if j~=k
          s1(iter+1:iter+n) = colidx; s2(iter+1:iter+n)
                            = rowidx;
          s3(iter+1:iter+n) = n*ifft(fphi([1 end:-1:2]));
                              iter = iter + n;
       end
    end
end
D = sparse(s1,s2,s3); clear s1 s2 s3;
xc = xc(:); yc = yc(:);

p = amd(D); % Reorder D to a block diagonal matrix
fchat = fft(fc)/sqrt(n); fchat = fchat(:);
coeffhat = zeros(m*n,1);
for k=1:m
    idx=(k-1)*n+1:k*n;
    coeffhat(p(idx))=D(p(idx),p(idx))\fchat(p(idx));
end
coeffhat = reshape(coeffhat,n,m);
coeff = sqrt(n)*ifft(coeffhat);
coeff = real(coeff(:));
disp('RBF coefficients solved!')
toc

% Store information to be reused
rbfinfo.coeff = coeff; rbfinfo.centers = xc+1i*yc;
rbfinfo.n = n; rbfinfo.m = m; rbfinfo.f = f; rbfinfo.F = F;
rbfinfo.phi = phi; rbfinfo.ep = ep;
```

References

1. Amestoy, P.R., Enseeiht-Irit, Davis, T.A., Duff, I.S.: Algorithm 837: Amd, an approximate minimum degree ordering algorithm. ACM Trans. Math. Softw. **30**(3), 381–388 (2004)
2. Banjai, L.: Eigenfrequencies of fractal drums. J. Comput. Appl. Math. **198**(1), 1–18 (2007)
3. Beatson, R.K., Cherrie, J.B., Mouat, C.T.: Fast fitting of radial basis functions: methods based on preconditioned GMRES iteration. Adv. Comput. Math. **11**(2–3), 253–270 (1999)
4. Buhmann, M.D.: Radial Basis Functions: Theory and Implementations. Cambridge Monographs on Applied and Computational Mathematics, vol. 12. Cambridge University Press, Cambridge (2003)
5. Buhmann, M.D., Dyn, N.: Spectral convergence of multiquadric interpolation. Proc. Edinb. Math. Soc. (2) **36**(2), 319–333 (1993)

6. Casciola, G., Montefusco, L.B., Morigi, S.: The regularizing properties of anisotropic radial basis functions. Appl. Math. Comput. **190**(2), 1050–1062 (2007)
7. Cheng, A.H.-D., Golberg, M.A., Kansa, E.J., Zammito, G.: Exponential convergence and h-c multiquadric collocation method for partial differential equations. Numer. Methods Partial Differ. Equ. **19**(5), 571–594 (2003)
8. Driscoll, T.A.: Algorithm 843: improvements to the Schwarz-Christoffel toolbox for MATLAB. ACM Trans. Math. Softw. **31**(2), 239–251 (2005)
9. Driscoll, T.A., Fornberg, B.: Interpolation in the limit of increasingly flat radial basis functions. Comput. Math. Appl. **43**, 413–422 (2002)
10. Driscoll, T.A., Heryudono, A.R.H.: Adaptive residual subsampling methods for radial basis function interpolation and collocation problems. Comput. Math. Appl. **53**, 927–939 (2007)
11. A Driscoll, T., Trefethen, L.N.: Schwarz-Christoffel Mapping. Cambridge Monographs on Applied and Computational Mathematics, vol. 8. Cambridge University Press, Cambridge (2002)
12. Fasshauer, G.E.: Meshfree Approximation Methods with MATLAB. Interdisciplinary Mathematical Sciences, vol. 6. World Scientific Publishing Co. Pte. Ltd., Hackensack (2007). With 1 CD-ROM (Windows, Macintosh and UNIX)
13. Faul, A.C., Goodsell, G., Powell, M.J.D.: A Krylov subspace algorithm for multiquadric interpolation in many dimensions. IMA J. Numer. Anal. **25**(1), 1–24 (2005)
14. Fornberg, B.: A Practical Guide to Pseudospectral Methods. Cambridge Monographs on Applied and Computational Mathematics, vol. 1. Cambridge University Press, Cambridge (1996)
15. Fornberg, B., Larsson, E., Flyer, N.: Stable computations with Gaussian radial basis functions in 2-D. Uppsala University Technical Report, 2009
16. Fornberg, B., Piret, C.: A stable algorithm for flat radial basis functions on a sphere. SIAM J. Sci. Comput. **30**(1), 60–80 (2007/2008)
17. Fornberg, B., Wright, G.: Stable computation of multiquadric interpolants for all values of the shape parameter. Comput. Math. Appl. **48**(5–6), 853–867 (2004)
18. Fornberg, B., Zuev, J.: The Runge phenomenon and spatially variable shape parameters in RBF interpolation. Comput. Math. Appl. **54**(3), 379–398 (2007)
19. Henrici, P.: Applied and Computational Complex Analysis. Wiley Classics Library, vol. 1. John Wiley & Sons Inc., New York (1988). Power series—integration—conformal mapping—location of zeros. Reprint of the 1974 original, A Wiley-Interscience Publication
20. Hesthaven, J., Gottlieb, S., Gottlieb, D.: Spectral Methods for Time-Dependent Problems. Cambridge University Press, Cambridge (2007)
21. Jung, J.-H., Durante, V.: An iteratively adaptive multiquardic radial basis function method for detection of local jump discontinuities. Appl. Numer. Math. **59**, 1449–1466 (2009)
22. Jung, J.-H., Gottlieb, S., Kim, S.O.: Two-dimensional edge detection based on the adaptive iterative MQ-RBF method. Appl. Numer. Math. (submitted)
23. Kansa, E.J.: 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**(8/9), 147–161 (1990)
24. Karageorghis, A., Chen, C.S., Smyrlis, Y.-S.: A matrix decomposition RBF algorithm: approximation of functions and their derivatives. Appl. Numer. Math. **57**(3), 304–319 (2007)
25. Larsson, E., Fornberg, B.: A numerical study of some radial basis function based solution methods for elliptic PDEs. Comput. Math. Appl. **46**(5–6), 891–902 (2003)
26. Platte, R.B.: How fast do radial basis function interpolants of analytic functions converge? IMA J. Numer. Anal. (submitted)
27. Platte, R.B., Driscoll, T.A.: Polynomials and potential theory for Gaussian radial basis function interpolation. SIAM J. Numer. Anal. **43**(2), 750–766 (2005) (electronic)
28. Roussos, G., Baxter, B.J.C.: Rapid evaluation of radial basis functions. J. Comput. Appl. Math. **180**(1), 51–70 (2005)
29. Sarra, S.A.: Adaptive radial basis function methods for time dependent partial differential equations. Appl. Numer. Math. **54**(1), 79–94 (2005)
30. Schaback, R.: Error estimates and condition numbers for radial basis function interpolation. Adv. Comput. Math. **3**, 251–264 (1995)
31. Schaback, R.: Limit problems for interpolation by analytic radial basis functions. J. Comput. Appl. Math. **212**, 127–149 (2008)
32. Trefethen, L.N.: Spectral Methods in MATLAB. Software, Environments, and Tools, vol. 10. SIAM, Philadelphia (2000)
33. Wendland, H.: Scattered Data Approximation. Cambridge Monographs on Applied and Computational Mathematics, vol. 17. Cambridge University Press, Cambridge (2005)
34. Yoon, J.: Spectral approximation orders of radial basis function interpolation on the Sobolev space. SIAM J. Math. Anal. **33**(4), 946–958 (2001) (electronic)