Applied Mathematics and Computation 265 (2015) 91–102

Contents lists available at ScienceDirect

Applied Mathematics and Computation

journal homepage: www.elsevier.com/locate/amc

# Implicit local radial basis function interpolations based on function values

Guangming Yao [a,*], Jia Duo [b], C.S. Chen [c,d], L.H. Shen [e]

[a] Department of Mathematics, Clarkson University, Potsdam, NY 13699-5815, USA
[b] Department of Mathematics, Heilongjiang Institute of Science and Technology, Heilongjiang, China
[c] Department of Mathematics, University of Southern Mississippi, Hattiesburg, MS, USA
[d] College of Mathematics, Taiyuan University of Technology, Taiyuan, China
[e] Department of Civil Engineering, National Taiwan University, Taiwan

ARTICLE INFO

*Keywords:*
RBFs
Interpolation
Large-scale
Sparse matrix

ABSTRACT

In this paper we propose two fast localized radial basis function fitting algorithms for solving large-scale scattered data interpolation problems. For each given point in the given data set, a local influence domain containing a small number of nearest neighboring points is established and a global interpolation is performed within this restricted domain. A sparse matrix is formulated based on the global interpolation in these local influence domains. The proposed methods have achieved both low computational cost and minimal memory storage. In comparison with the compactly supported radial basis functions, the proposed fitting algorithms are highly accurate. The numerical examples have provided strong evidence that the two proposed algorithms are indeed highly efficient and accurate. In the two proposed algorithms, we have successfully solved a large-scale interpolation problem with 225,000 interpolation points in two dimensional space.



## 1. Introduction

During the past two decades, radial basis functions (RBFs) have undergone intensive research and achieved enormous success in various areas of science and engineering such as multivariate data interpolation and approximation, surface reconstruction, computer graphics, numerical solutions of partial differential equations, neural networks, machine learning, etc. The initial development of RBFs was focused on multivariate data interpolation. In 1982, Franke [1] published a review paper evaluating virtually all of the interpolation methods for scattered data sets available at that time. As a result, RBFs have attracted great attention as an effective tool for scattered data interpolation problems. One of the attractive features of RBFs is the simplicity of handling high dimensional scattered data.

Despite the many attractive features of RBFs, most of the commonly used RBFs are globally supported which means the resulting matrix for data reconstruction is dense and could be highly ill-conditioned. For large-scale problems, this implies high computational costs and possible stability issues. For instance, using the general direct solver to fit an RBF with $N$ centers requires $O(N^3)$ flops and $O(N^2)$ storage. These computational costs are not feasible when $N$ is sufficiently large. In many real life applications, the data set can easily go beyond 10,000 which is difficult for RBFs to handle in a reasonable way. In the past, the interpolation of data associated with large-scale problems was typically achieved by decomposing the problem into sub-domains

* Corresponding author. Tel.: +1 315 268 6496.
*E-mail address:* guangmingyao@gmail.com, gyao@clarkson.edu (G. Yao).

http://dx.doi.org/10.1016/j.amc.2015.04.107
0096-3003/

**Table 1**
Commonly used radial basis functions.

| | |
|---|---|
| Gaussian: | $\varphi(r) = e^{-cr^2}$, $c > 0$ |
| Inverse multiquadrics: | $\varphi(r) = (r^2 + c^2)^{-1/2}$, $c > 0$ |
| Matern function: | $\varphi(r) = (cr)^n K_n(cr)$, $c > 0$, where $K_n$ is the spherical Bessel function, $n > 0$ |
| Multiquadrics: | $\varphi(r) = (r^2 + c^2)^{1/2}$, $c > 0$ |
| Normalized multiquadrics: | $\varphi(r) = (1 + (cr)^2)^{1/2}$, $c > 0$ |
| Thin-plate spline: | $r^2 \ln(r)$ |
| Polyharmonic: | $r^{2n} \ln(r)$ |

on which local interpolation problems are solved, often with additional constraints across the domain boundaries so as to assure continuity of the function and perhaps continuity of its derivatives as well [1–3]. One disadvantage of the domain decomposition technique is the requirement of domain discretization which is often very tedious. For high dimensional problems, such an approach is non-trivial. Compactly supported radial basis functions (CS-RBFs) [4] were developed in the mid-1990's to alleviate these difficulties. The use of CS-RBF interpolation generates a sparse interpolation matrix which is desirable for interpolating large-scale problems. However, the convergence rate of CS-RBFs is relatively slow, and it is difficult to achieve high accuracy.

We can observe that there are a number of fast fitting algorithms and fast computational methods available for tackling the large-scale interpolation problems. We refer readers to references [5–8].

Our purpose in this paper is to propose two localized interpolation techniques to reconstruct surfaces based on the given interpolation points. Such localized methods are different from CS-RBFs. In our approach, global RBFs are used on each local influence domain. A global interpolation is performed on each influence domain and then a sparse matrix is formulated to link all of the influence domains together. The RBFs with strong convergence rates such as Gaussian, multiquadrics (MQ), normalized MQ, inverse MQ, and Matern functions are used as the basis functions. The sub-optimal shape parameter of these RBFs can be obtained by using the technique of the so called leave one out cross validation (LOOCV) [9,10]. For every given data point, we need to search for neighboring evaluation points to form the influence domain so that the functions' values at the evaluation points can be approximated. Our proposed approaches have the features of fast computation, small memory storage, and high accuracy for large-scale fitting problems. The highly efficient kd-tree algorithm [11] is used in the local method to search for nearest neighbors of the evaluation points among the interpolation points.

## 2. Radial basis functions interpolation

RBFs have been applied to solve many problems in science and engineering. Among these problems, scattered data interpolation is one of the earliest topics for applications of RBFs. Let $\Omega \subset \mathbf{R}^n$, and $\mathbf{x}_i \in \Omega$. We are given $(\mathbf{x}_i, f(\mathbf{x}_i)) \equiv (\mathbf{x}_i, f_i)$ as known coordinate pairs. Let $\{\mathbf{s}_j\}_{j=1}^N \subset \Omega$ denote the evaluation or test points at which we seek to interpolate $f$ by $\hat{f}$, such that $\hat{f}(\mathbf{s}_j) \approx f(\mathbf{s}_j)$, $j = 1, \ldots, N$, with $\hat{f}(\mathbf{s}_k) = f(\mathbf{s}_k)$ whenever $\mathbf{s}_k = \mathbf{x}_k$.

The interpolation of the multivariate function $f$ can be constructed by linear combinations of univariate interpolation functions with Euclidean norm $\|\cdot\|$. Popular invariant functions include the radial basis functions whose value at any point $\mathbf{x} \in \mathbf{R}^n$ depends only on the distance from the fixed point $\mathbf{c}$ and can be written as

$$\phi(\mathbf{x}) = \phi(\|\mathbf{x} - \mathbf{c}\|),$$

where $\mathbf{c}$ is the center of the radial basis function $\phi$. Some commonly used RBFs are listed in Table 1.

The interpolants $\hat{f}$ to function $f$ can be obtained by

$$\hat{f}(\mathbf{x}) = \sum_{j=1}^{N} \alpha_j \phi(\|\mathbf{x} - \mathbf{x}_j\|), \tag{1}$$

in which all of the interpolation points $\mathbf{x}_i$, $i = 1, 2, \ldots, N$ are chosen as the centers of the basis functions, and $\alpha_i$, $i = 1, 2, \ldots, N$ are unknown real coefficients, which can be obtained by interpolation:

$$f(\mathbf{x}_i) = \sum_{j=1}^{N} \alpha_j \phi(\|\mathbf{x}_i - \mathbf{x}_j\|), \quad i = 1, 2, \ldots, N. \tag{2}$$

The resulting system of linear equations can be written in a matrix/vector format

$$A\alpha = \mathbf{f},$$

where $\boldsymbol{\alpha} = [\alpha_1, \alpha_2, \cdots, \alpha_N]^T$, $\mathbf{f} = [f_1, f_2, \cdots, f_N]^T$, and

$$A = \begin{bmatrix} \phi(\|\mathbf{x}_1 - \mathbf{x}_1\|) & \cdots & \phi(\|\mathbf{x}_1 - \mathbf{x}_n\|) \\ \vdots & \cdots & \vdots \\ \phi(\|\mathbf{x}_n - \mathbf{x}_1\|) & \cdots & \phi(\|\mathbf{x}_n - \mathbf{x}_n\|) \end{bmatrix}, \tag{3}$$

where $T$ is used to denote the transpose of a vector or matrix. The invertibility of the linear system (3) is difficult to determine. If the RBF $\phi$ is positive definite, the solvability of the system of linear equations (3) is guaranteed. For example, Gaussian and Matern/Soboleve RBFs are positive definite, and inverse multiquadrics and CS-RBFs [4,12,13] are strictly positive definite [14] and thus maintain the invertibility of matrix (3).

MQ is conditionally positive definite [15]. The interpolation matrix generated by thin plate splines can be singular, even with non-trivial sets of distinct centers [16,17]. Polyharmonic splines, a general case of thin plate splines, are conditionally positive definite. A polynomial of a certain maximal degree must to be added to (1) to insure invertibility. Let $\mathscr{P}$ be a polynomial function space of order $q$, and let $p_1, p_2, \ldots, p_q$ be a basis of $\mathscr{P}$; then the interpolation function in (1) can be augmented in the following manner

$$\hat{f}(\mathbf{x}) = \sum_{j=1}^{N} \alpha_j \phi(\|\mathbf{x} - \mathbf{x}_j\|) + \sum_{l=1}^{q} \beta_l p_l(\mathbf{x}). \tag{4}$$

Since there are $q$ additional degrees of freedom in (4), the standard polynomial insolvency constraint [18],

$$p_l(\mathbf{x}) = 0, \forall l = 1, 2, \ldots, q \quad \text{and} \quad p \in \mathcal{P}_{q-1} \Longrightarrow p = 0, \tag{5}$$

must be applied. Thus the collocation technique gives the following linear system:

$$\sum_{j=1}^{N} \alpha_j \phi(\|\mathbf{x}_i - \mathbf{x}_j\|) + \sum_{l=1}^{q} \beta_l p_l(\mathbf{x}_i) = f(\mathbf{x}_i), \quad i = 1, 2, \ldots, N, \tag{6}$$

$$\sum_{j=1}^{N} \alpha_j p_l(\mathbf{x}_j) = 0, \quad l = 1, 2, \ldots, q. \tag{7}$$

The system (6)–(7) is a square linear system with $N + q$ unknown coefficients, whose solution can be obtained using standard methods providing the coefficients uniquely determine the interpolation function (4).

RBFs typically provide very high rates of convergence. The error in approximating or interpolating a set of data representing the sampling of a smooth function goes to zero very rapidly. Thus, fewer discrete points are needed to accurately represent or reconstruct the underlying functions. This makes RBFs highly efficient computational tools. Furthermore, RBFs are easy to implement numerically. However, constructing the system of linear equations by including all of the collocation points within the domain is generally unstable because the resulting dense matrices become increasingly ill-conditioned and the matrices are sensitive to the choice of free parameters during the RBF formulation. This makes it difficult, if not impossible, to use global RBF interpolation to solve large-scale problems.

In contrast, localized formulations can reduce the ill-conditioning of the coefficient matrix. Iterative methods can be used to improve efficiency when determining coefficients. Since only the nearby collocation points are needed in the formulation when using local methods, the ill-conditioning associated with large, dense matrix systems can be alleviated. Another important advantage of local RBF approaches is that the MQ or IMQ shape parameter affects the numerical results only slightly. In general, the MQ is regarded as one of the best RBFs in terms of accuracy, assuming a suitable shape parameter can be found. For global RBF approaches, choosing a suitable shape parameter is still a challenging issue. A further advantage of local approaches is that the computational efficiency does not compromise the accuracy of the methods. Instead of solving a dense system as required by global approaches, local approaches result in a sparse matrix that can be solved efficiently.

## 3. Implicit local radial basis function interpolation

Let $\{\mathbf{x}_i\}_{i=1}^{N}$ be a given large set of scattered data points (sampling points), and $\{\mathbf{z}_j\}_{j=1}^{N_t}$ be a set of evaluation points. Suppose the function values $\{f(\mathbf{x}_i)\}_{i=1}^{N}$ at the interpolation points are given; we try to use these known values to obtain the approximate values of $\{f(\mathbf{z}_j)\}_{j=1}^{N_t}$.

### 3.1. Algorithm 1

For each $\mathbf{x}_i$, we first create a local influence domain, $\Omega_i = \{\mathbf{z}_j^{[i]}\}_{j=1}^{n_s}$, which includes the nearest $n_s$ evaluation points to $\mathbf{x}_i$. In this approach, each influence domain contains only evaluation points (see Fig. 1).

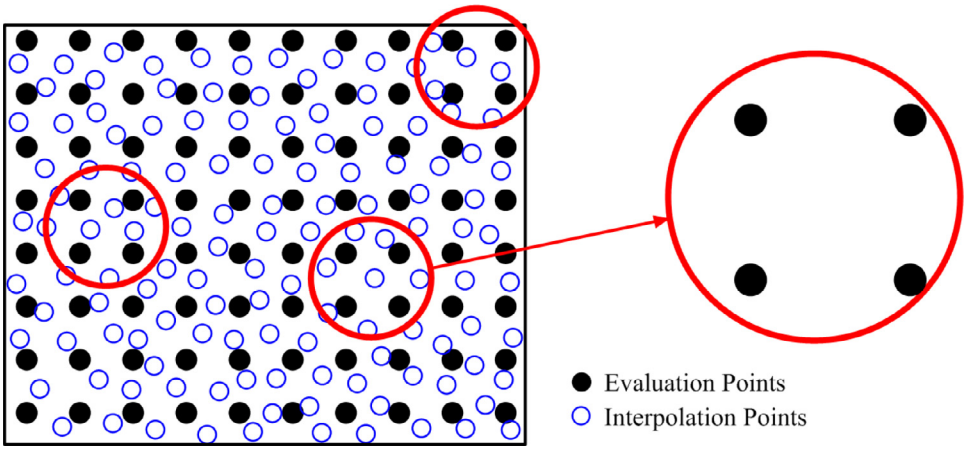


**Fig. 1.** Local influence domain containing only evaluations points.

Let $\phi$ be a radial basis function. Choose all evaluation points in the local domain $\Omega_i$ as the centers of the basis functions. From (1), using RBF interpolation on $\Omega_i$, we have the following linear system

$$\begin{bmatrix} \hat{f}(\mathbf{z}_1^{[i]}) \\ \hat{f}(\mathbf{z}_2^{[i]}) \\ \vdots \\ \hat{f}(\mathbf{z}_{n_s}^{[i]}) \end{bmatrix} = \begin{bmatrix} \phi(\|\mathbf{z}_1^{[i]} - \mathbf{z}_1^{[i]}\|) \ \dots \ \phi(\|\mathbf{z}_1^{[i]} - \mathbf{z}_{n_s}^{[i]}\|) \\ \phi(\|\mathbf{z}_2^{[i]} - \mathbf{z}_1^{[i]}\|) \ \dots \ \phi(\|\mathbf{z}_2^{[i]} - \mathbf{z}_{n_s}^{[i]}\|) \\ \vdots \quad\quad \vdots \quad\quad \vdots \\ \phi(\|\mathbf{z}_n^{[i]} - \mathbf{z}_1^{[i]}\|) \ \dots \ \phi(\|\mathbf{z}_n^{[i]} - \mathbf{z}_{n_s}^{[i]}\|) \end{bmatrix} \begin{bmatrix} \alpha_1^{[i]} \\ \alpha_2^{[i]} \\ \vdots \\ \alpha_{n_s}^{[i]} \end{bmatrix}. \tag{8}$$

The above linear system of equations can be written in the matrix form

$$\hat{\mathbf{f}}_{n_s} = \mathbf{\Phi}_{n_s n_s} \boldsymbol{\alpha}^{[i]} \tag{9}$$

where $\hat{\mathbf{f}}_{n_s} = [\hat{f}(\mathbf{z}_1^{[i]}), \hat{f}(\mathbf{z}_2^{[i]}), \dots, \hat{f}(\mathbf{z}_{n_s}^{[i]})]^T$, $\boldsymbol{\alpha}^{[i]} = [\alpha_1^{[i]}, \alpha_2^{[i]}, \dots, \alpha_{n_s}^{[i]}]^T$, and

$$\mathbf{\Phi}_{n_s n_s} = [\phi(\|\mathbf{z}_j^{[i]} - \mathbf{z}_k^{[i]}\|)]_{1 \le j,k \le n_s}.$$

It can be proved that $\Phi_{n_s n_s}$ is non-singular for positive definite RBFs if all evaluation points inside $\Omega_i$ are distinct. Thus,

$$\boldsymbol{\alpha}^{[i]} = \mathbf{\Phi}_{n_s n_s}^{-1} \hat{\mathbf{f}}_{n_s}. \tag{10}$$

Therefore, for $i = 1, 2, \dots, N$,

$$\begin{aligned} f(\mathbf{x}_i) &= \sum_{j=1}^{n_s} \alpha_j^{[i]} \phi(\|\mathbf{x}_i - \mathbf{z}_j^{[i]}\|) \\ &= [\phi(\|\mathbf{x}_i - \mathbf{z}_1^{[i]}\|), \dots, \phi(\|\mathbf{x}_i - \mathbf{z}_{n_s}^{[i]}\|)] \boldsymbol{\alpha}^{[i]} \\ &= [\phi(\|\mathbf{x}_i - \mathbf{z}_1^{[i]}\|), \dots, \phi(\|\mathbf{x}_i - \mathbf{z}_{n_s}^{[i]}\|)] \Phi_{n_s n_s}^{-1} \hat{\mathbf{f}}_{n_s} \\ &= \mathbf{\Lambda}_{n_s}(\mathbf{x}_i) \hat{\mathbf{f}}_{n_s}, \end{aligned} \tag{11}$$

where

$$\mathbf{\Lambda}_{n_s} = [\phi(\|\mathbf{x}_i - \mathbf{z}_1^{[i]}\|), \dots, \phi(\|\mathbf{x}_i - \mathbf{z}_{n_s}^{[i]}\|)] \mathbf{\Phi}_{n_s n_s}^{-1}. \tag{12}$$

We can easily extend $\hat{\mathbf{f}}_{n_s}$ to a global vector $\hat{\mathbf{f}}_{N_t} = [\hat{f}(\mathbf{z}_1), \dots, \hat{f}(\mathbf{z}_{N_t})]^T$ by adding zeros into $\mathbf{\Lambda}_{n_s}$; i.e.,

$$f(\mathbf{x}_i) = \mathbf{\Lambda}_{N_t}(\mathbf{x}_i) \hat{\mathbf{f}}_{N_t}, \quad 1 \le i \le N. \tag{13}$$

This is an $N \times N_t$ sparse system with $N_t$ unknown approximate function values at $N_t$ evaluation points, $\hat{f}(\mathbf{z}_j), j = 1, 2, \dots, N_t$. If $N < N_t$, (13) is an under-determined system which gives inaccurate interpolation. If $N_t \le N$, then the system is an over-determined or a squared system which can be solved by the least squares method.

Once the approximate function values at $N_t$ evaluation points are obtained, we can proceed to approximate the derivatives $f$ at the original interpolation points. From (14), we have, for $i = 1, 2, \dots, N$,

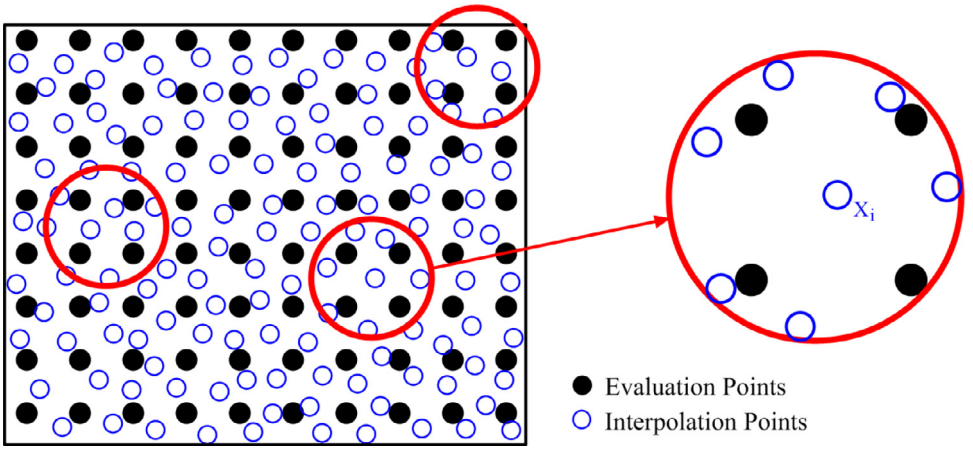


**Fig. 2.** Interpolation and evaluations points in the local influence domain.

$$\begin{aligned} f_x(\mathbf{x}_i) &= \sum_{j=1}^{n_s} \alpha_j^{[i]} \phi_x(\|\mathbf{x}_i - \mathbf{z}_j^{[i]}\|) \\ &= [\phi_x(\|\mathbf{x}_i - \mathbf{z}_1^{[i]}\|), \ldots, \phi_x(\|\mathbf{x}_i - \mathbf{z}_{n_s}^{[i]}\|)]\boldsymbol{\alpha}^{[i]} \\ &= [\phi_x(\|\mathbf{x}_i - \mathbf{z}_1^{[i]}\|), \ldots, \phi_x(\|\mathbf{x}_i - \mathbf{z}_{n_s}^{[i]}\|)]\Phi_{n_s n_s}^{-1} \hat{\mathbf{f}}_{n_s} \\ &= \boldsymbol{\Theta}_{n_s}(\mathbf{x}_i)\hat{\mathbf{f}}_{n_s} \end{aligned} \tag{14}$$

where

$$\boldsymbol{\Theta}_{n_s} = \left[\phi_x(\|\mathbf{x}_i - \mathbf{z}_1^{[i]}\|), \ldots, \phi_x(\|\mathbf{x}_i - \mathbf{z}_{n_s}^{[i]}\|)\right] \boldsymbol{\Phi}_{n_s n_s}^{-1}. \tag{15}$$

Similar to (13), we have

$$f_x(\mathbf{x}_i) = \boldsymbol{\Theta}_{N_t}(\mathbf{x}_i)\hat{\mathbf{f}}_{N_t}, \quad 1 \le i \le N. \tag{16}$$

Since $\hat{\mathbf{f}}_{N_t}$ is known from solving (13), $\{f_x(\mathbf{x}_i)\}_{i=1}^N$ can be approximated from (16). $\{f_y(\mathbf{x}_i)\}_{i=1}^N$ can be obtained in a similar way.

The limitation of such an approach is that the number of evaluation points has to be sufficiently close to the number of interpolation points since each interpolation point needs the evaluation points as support in the influence domain.

### 3.2. Algorithm 2

In this subsection, we propose to modify Algorithm 1 from the last subsection by employing all of the interpolation and evaluation points in each influence domain (see Fig. 2). In this way, even if only a small number of evaluation points is available, we can still conduct the approximation on these evaluation points. However, we have to overcome the difficulty that the resultant matrix is under-determined.

Let $\{\mathbf{x}_i\}_{i=1}^{n_1}$ and $\{\mathbf{z}_j^{[i]}\}_{j=1}^{n_2}$ be the interpolation and evaluation points inside the influence domain. Let $\{\mathbf{y}_k^{[i]}\}_{k=1}^{n_s} = \{\mathbf{x}_i\}_{i=1}^{n_1} \cup \{\mathbf{z}_j^{[i]}\}_{j=1}^{n_2}$ where $n_s = n_1 + n_2$. Replacing $\mathbf{z}_j^{[i]}$ by $\mathbf{y}_j^{[i]}$ from (8) to (12), a similar interpolation procedure can be obtained. As a result, similar to (13), we have

$$f(\mathbf{x}_i) = \boldsymbol{\Lambda}_{N_t+N}(\mathbf{x}_i)\begin{bmatrix} \hat{\mathbf{f}}_{N_t} \\ \mathbf{f}_N \end{bmatrix}, \quad 1 \le i \le N, \tag{17}$$

where $\mathbf{f}_N = [f(\mathbf{x}_1), \ldots, f(\mathbf{x}_N)]^T$. Notice that (17) is an $N \times (N_t + N)$ system of equations which is an under-determined system. Alternatively, we reformulate (17) in the following way so that the system is solvable

$$\begin{bmatrix} \boldsymbol{\Xi}_{N\times N_t} & \boldsymbol{\Xi}_{N\times N} \\ \mathbf{O}_{N\times N_t} & \mathbf{I}_{N\times N} \end{bmatrix} \begin{bmatrix} \hat{\mathbf{f}}_{N_t} \\ \mathbf{f}_N \end{bmatrix} = \begin{bmatrix} \mathbf{f}_N \\ \mathbf{f}_N \end{bmatrix} \tag{18}$$

where

$$[\boldsymbol{\Xi}_{N\times N_t} \quad \boldsymbol{\Xi}_{N\times N}] = \boldsymbol{\Lambda}_{N_t+N}.$$

To illustrate, the profile of the sparse matrix in (18) for $N = 100$ and $N_t = 80$ is shown in Fig. 3 in which the interpolation points are uniformly distributed grid points and the evaluation points are quasi-random points. The novelty of this approach is the inclusion of the identity matrix $\mathbf{I}_{N\times N}$ and the original data values $\mathbf{f}_N$ into the formulation of (18). The approximate function

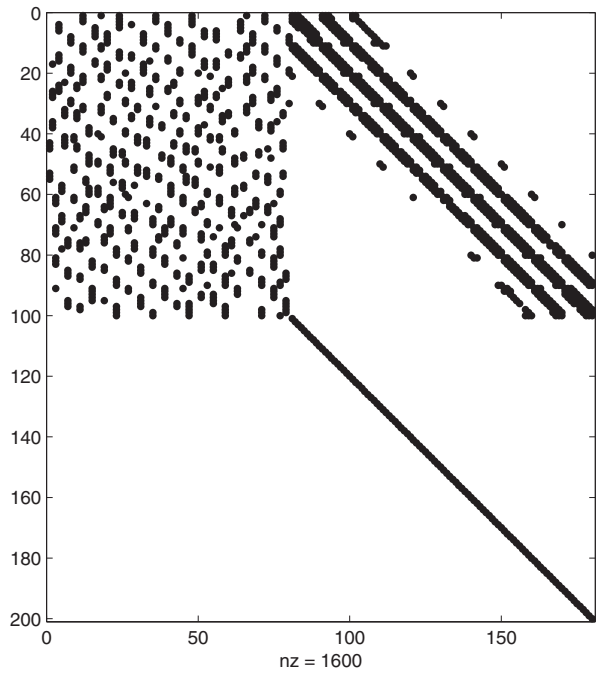


**Fig. 3.** Profile of the sparse matrix in (18) for $N = 100$ and $N_t = 80$ .

values $\hat{\mathbf{f}}_{N_t}$ at the evaluation points can be obtained by solving (18) using the least squares method. The partial derivative $f_x$ can be evaluated in a similar way as (16).

Note that (18) can be further simplified to the following form:

$$\Xi_{N\times N_t}\hat{\mathbf{f}}_{N_t} = \mathbf{f}_N - \Xi_{N\times N}\mathbf{f}_N. \tag{19}$$

## 4. Numerical results

To demonstrate the effectiveness of our proposed algorithm, we first perform the test on six benchmark functions [1]. We also compare our results with compactly supported radial basis functions (CS-RBFs). In this section, we choose the algorithm of leave-one-out cross validation (LOOCV) [9,10] for selecting a sub-optimal value of the shape parameter. The following MATLAB code of LOOCV is given in [9]. We refer readers to reference [9] for further details.

```
1  function ceps = costEps(c,rbf,DM,rhs)
2  A = rbf(DM,c);
3  invA = pinv(A);
4  errorvector = (invA*rhs)./diag(invA);
5  ceps = norm(errorvector);
```

Note that 'rhs' in the code is the right hand side of the matrix system $Ax = b$. Here we choose 'rhs' as the test function $f(x, y)$. 'rbf' for the Gaussian function is defined as follows:

```
rbf = @ (c, r) exp(-c*r.^2);
```

DM is the distance matrix of matrix $A$ in the influence domain; i.e.,

$$DM = \begin{pmatrix} 0 & \|x_1^{[i]} - x_2^{[i]}\| & \cdots & \|x_1^{[i]} - x_n^{[i]}\| \\ \|x_2^{[i]} - x_1^{[i]}\| & 0 & \cdots & \|x_2^{[i]} - x_n^{[i]}\| \\ \cdots & \cdots & \cdots & \cdots \\ \|x_n^{[i]} - x_1^{[i]}\| & \|x_n^{[i]} - x_2^{[i]}\| & \cdots & 0 \end{pmatrix}$$

**Table 2**
Absolute maximum errors and sub-optimal shape parameters using Algorithm 1 and $n_s = 30$ for various $N$ and $N_t$.

| $(N, N_t)$ | $(50^2, 2000)$ | | $(100^2, 9000)$ | | $(150^2, 20,000)$ | | $(200^2, 35,000)$ | |
|---|---|---|---|---|---|---|---|---|
| | $c$ | $\epsilon_{\max}$ | $c$ | $\epsilon_{\max}$ | $c$ | $\epsilon_{\max}$ | $c$ | $\epsilon_{\max}$ |
| F1 | 4.994 | 2.12E−04 | 4.944 | 2.35E−05 | 1.922 | 2.47E−05 | 4.994 | 3.97E−06 |
| F2 | 4.994 | 1.87E−04 | 2.414 | 2.51E−05 | 1.909 | 1.65E−05 | 4.993 | 5.05E−06 |
| F3 | 4.831 | 3.58E−05 | 3.909 | 1.06E−06 | 3.037 | 8.66E−07 | 1.909 | 4.55E−07 |
| F4 | 1.248 | 3.61E−06 | 3.201 | 3.20E−07 | 1.830 | 1.10E−07 | 0.967 | 1.03E−07 |
| F5 | 0.376 | 4.07E−05 | 3.578 | 4.59E−06 | 4.569 | 1.13E−06 | 1.018 | 1.40E−06 |
| F6 | 0.375 | 1.26E−05 | 2.160 | 2.46E−06 | 0.526 | 1.07E−06 | 0.959 | 4.15E−07 |

Using MATLAB, the calling sequence for the cost function costEps is given by

```
c = fminbnd(@(c) costEps(c,rbf,DM,rhs),minc,maxc);
```

where '`minc`' and '`maxc`' define the interval to search for the sub-optimal shape parameter $c$. For the numerical tests in this section, we choose minc = 0 and maxc = 5. Since the search of the sub-optimal shape parameter using LOOCV could be time consuming, it is not cost effective to search the sub-optimal shape parameter for each influence domain. In our test, we find it is sufficient to search the shape parameter for one of the influence domains and then use it for the rest of the computation. Such a strategy may not be the best, but we find it quite satisfactory as shown in the numerical results in this section.

In the numerical implementation, we select a set of RBF centers $\{(x_i, y_i)\}_{i=1}^{N}$ and another set of evaluation (or test) points $\{(\xi_i, \eta_i)\}_{i=1}^{N_t}$ with the constraint that $N_t < N$. Furthermore, we select the evenly distributed RBF centers and apply Halton quasi-random points for the selection of evaluation points. For convenience of locating evaluation points in the influence domain, we merge these two sets of points together by placing the evaluation points on top of the center points. Let $n_s$ denote the number of points in each influence domain. To efficiently search evaluation points inside each influence domain, we employ the kd-tree algorithm [11].

All the numerical results were obtained on a Dell Latitude E6410 8 GB RAM under Windows 7 using MATLAB.

**Example 1.** In this example, we consider Franke's six benchmark test functions [1] on the unit square. These test functions are given as follows

1. Surface F1:

$$F1(x,y) = \frac{3}{4}\exp\left(\frac{-1}{4}((9x-2)^2+(9y-2)^2)\right) + \frac{3}{4}\exp\left(\frac{-1}{49}(9x+1)^2 - \frac{1}{10}(9y+1)^2\right) + \frac{1}{2}\exp\left(\frac{-1}{4}(9x-7)^2 - \frac{1}{4}(9y-3)^2\right) - \frac{1}{5}\exp(-(9x-4)^2-(9y-7)^2). \tag{20}$$

2. Surface F2:

$$F2(x,y) = \frac{1}{9}(\tanh(9y-9x)+1). \tag{21}$$

3. Surface F3:

$$F3(x,y) = \frac{1.25+\cos(5.4y)}{6[1+(3x-1)^2]}. \tag{22}$$

4. Surface F4:

$$F4(x,y) = \frac{1}{3}\exp\left[-\frac{81}{16}\left(\left(x-\frac{1}{2}\right)^2+\left(y-\frac{1}{2}\right)^2\right)\right]. \tag{23}$$

5. Surface F5:

$$F5(x,y) = \frac{1}{3}\exp\left[-\frac{81}{4}\left(\left(x-\frac{1}{2}\right)^2+\left(y-\frac{1}{2}\right)^2\right)\right]. \tag{24}$$

6. Surface F6:

$$F6(x,y) = \frac{1}{9}\left[64 - 81\left(\left(x-\frac{1}{2}\right)^2+\left(y-\frac{1}{2}\right)^2\right)\right] - \frac{1}{2}. \tag{25}$$

The profiles of F1 – F6 are shown in Figs. 4–6. In the first part of this example, we employ Algorithm 1 to carry out the computation. In Table 2, we show the maximum errors and sub-optimal shape parameters for various $N$ and $N_t$ using $n_s = 30$. Note that the initial selection of $n_s$ points in the influence domain contains both RBF centers and evaluation points. After we

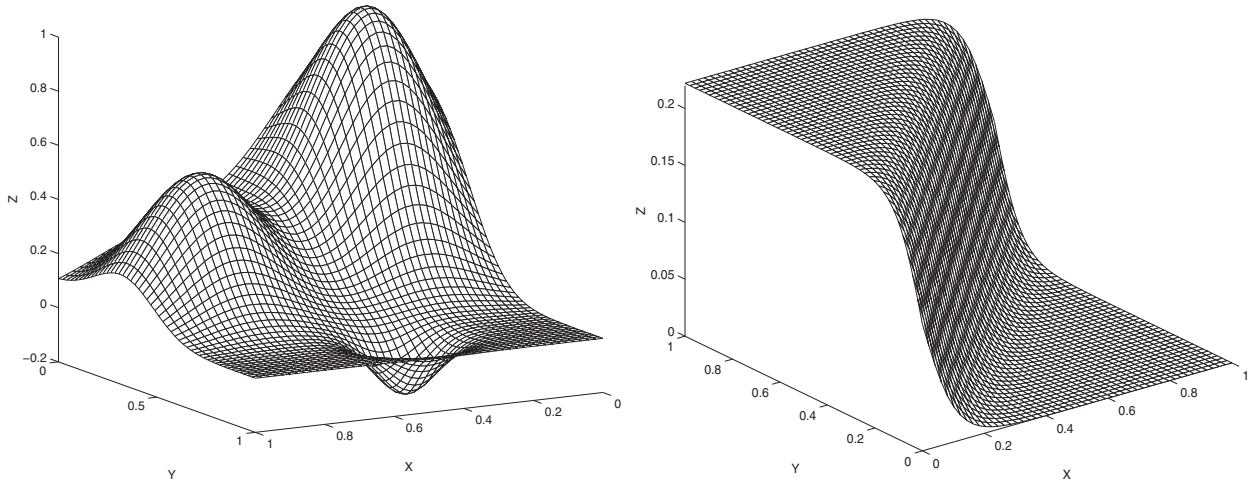


**Fig. 4.** Test functions F1 and F2.

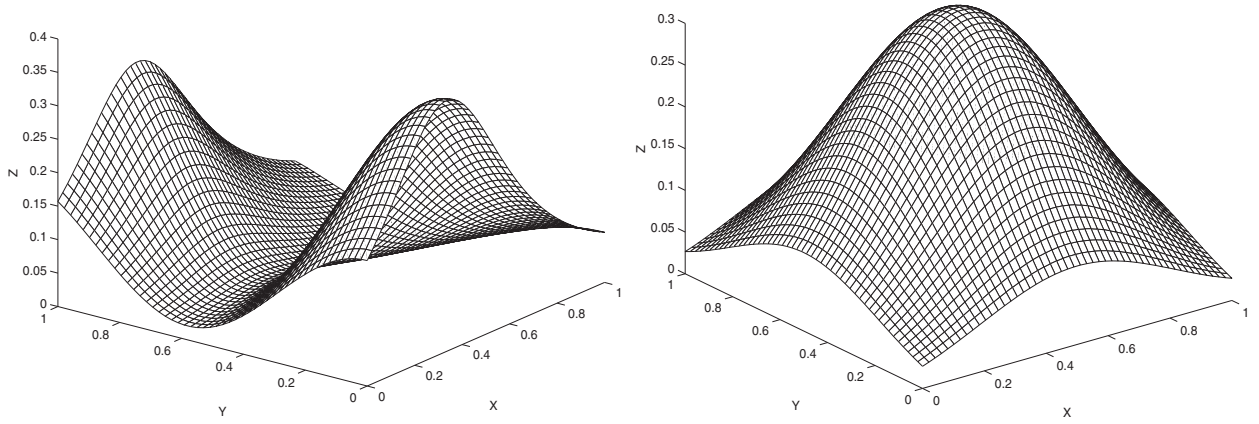


**Fig. 5.** Test functions F3 and F4.

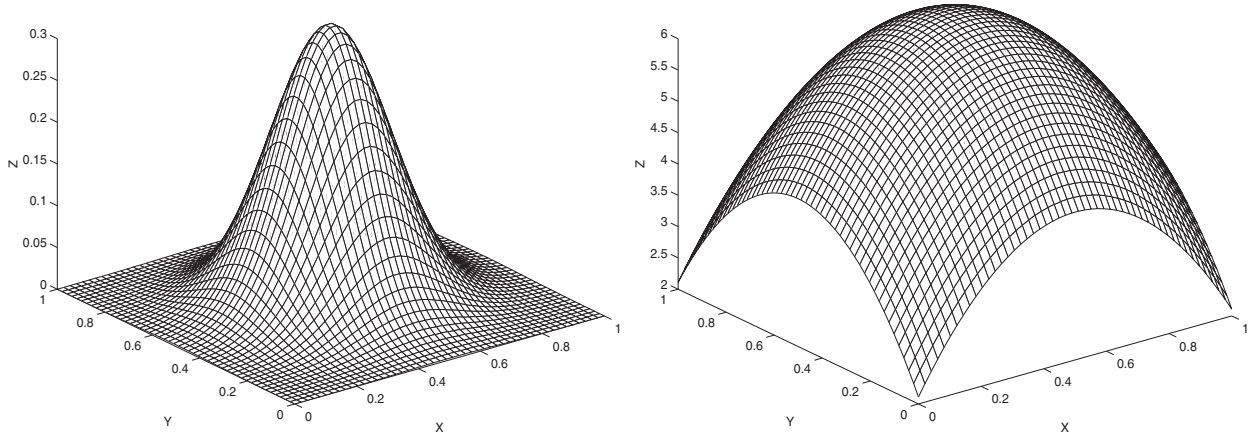


**Fig. 6.** Test functions F5 and F6.

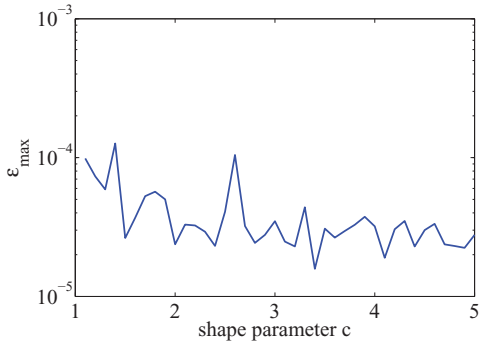


**Fig. 7.** Maximum error versus shape parameter for F1.

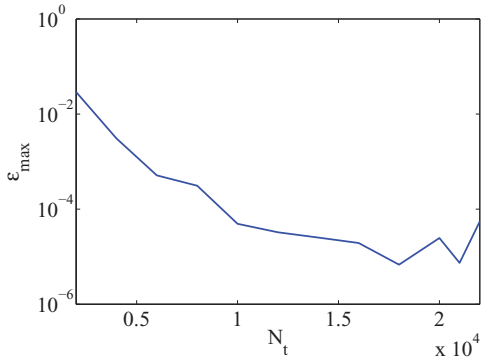


**Fig. 8.** The effect of the number of evaluation points using $N = 22, 500$, $ns = 30$ for F1.

**Table 3**
The maximum errors $\epsilon_{\max}$ for various $n_s$.

| $(N, N_t) = (50^2, 2000)$ | | | $(N, N_t) = (150^2, 18,000)$ | | |
|---|---|---|---|---|---|
| $n_s$ | $n$ | $\epsilon_{\max}$ | $n_s$ | $n$ | $\epsilon_{\max}$ |
| 20 | 7 | 5.27E−04 | 20 | 7 | 5.27E−05 |
| 30 | 11 | 2.12E−04 | 30 | 11 | 6.80E−06 |
| 40 | 15 | 4.98E−05 | 40 | 13 | 2.20E−06 |
| 50 | 20 | 1.56E−05 | 50 | 19 | 3.70E−06 |
| 60 | 24 | 2.11E−05 | 60 | 23 | 5.02E−06 |

remove RBF centers, only evaluation points remain inside the influence domain (see Fig. 1). Hence, the number of evaluation points in each influence domain could be different.

The test functions F1 and F2 are more challenging for approximation. In the following tests, we will focus on the test of the F1 function. With the increasing number of RBF centers and evaluation points, we can achieve high accuracy. In Fig. 7, the maximum errors are obtained using $N = 100^2$, $N_t = 9000$, and $n_s = 30$ for various shape parameters. For efficiency, the same shape parameter is being used for every influence domain. We observe that the results are fairly stable with respect to the shape parameter and consistent with the results obtained in Table 2 where the sub-optimal shape parameter is $c = 4.944$.

It is important to study the relation between the number of RBF centers and the evaluation points. From Fig. 8, we observe that the larger the number of evaluation points, the better the accuracy. In general, to obtain good accuracy, $N_t$ should be close but not too close to $N$.

Another issue is how to choose the number $n_s$ for each influence domain. Let $n$ denote the number of evaluation points in each influence domain. In Table 3, we observe that for smaller $(N, N_t)$, we can choose larger $n_s$ and vice versa for larger $(N, N_t)$. In general, it is sufficient to choose $n_s = 30$ or 40.

Next, we compare our proposed algorithm with Wendland's CS-BRBs [4] where the basis function $\phi(r) = (1-r)_+^4(4r+1)$ is being used. We use the MATLAB CS-RBFs code in [19] to produce the result in Table 4. For CS-RBFs, the accuracy will be affected by the size of local support and the number of RBF centers. Thus the number of evaluation points has no impact on the final results. However, our proposed method depends both on the number of RBF centers and evaluation points. We try to make the comparison as fair as possible by considering the number of nonzero elements (nz) in the formulated sparse matrix. In contrast to the CS-RBFs, as we can see in Table 4, Algorithm 1 has a clear advantage both in accuracy and efficiency.

**Table 4**
Comparison of the proposed method and CS-RBF for the F1 function.

| $N$ | $N_t$ | ILRBF | | | CS-RBF | | |
|---|---|---|---|---|---|---|---|
| | | $\epsilon_{\max}$ | nz | CPU | $\epsilon_{\max}$ | nz | CPU |
| 16,641 | 15,500 | 7.85E−05 | 244,858 | 11.94 | 3.91E−02 | 473,337 | 34.37 |

**Table 5**
Globally defined radial basis functions.

| |
|---|
| Gaussian: $\varphi(r) = e^{-cr^2}$ for $c > 0$ |
| Inverse multiquadrics: $\varphi(r) = (r^2 + c^2)^{-1/2}$, $c > 0$<br>Matern function: $\varphi(r) = (cr)^n K_n(cr)$, where $K_n$ is the spherical Bessel function, $n > 0$<br>Multiquadrics: $\varphi(r) = (r^2 + c^2)^{1/2}$<br>Normalized multiquadrics: $\varphi(r) = (1 + (cr)^2)^{1/2}$ |

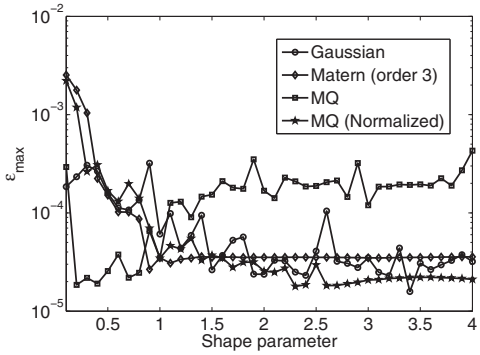


**Fig. 9.** Maximum error versus shape parameter for various RBFs.

**Table 6**
The maximum error and CPU time using $N = 122,500$, $N_t = 117,500$.

| $n_s$ | Shape parameter | $\epsilon_{\max}$ | CPU |
|---|---|---|---|
| 30 | 4.826 | 6.02E−06 | 132.8 |

**Table 7**
The absolute maximum errors and sub-optimal shape parameters using Algorithm 2 and $n_s = 30$ for various $N$ and $N_t$.

| $(N, N_t)$ | $(50^2, 1800)$ | | $(100^2, 8000)$ | | $(150^2, 18,000)$ | | $(200^2, 32,000)$ | |
|---|---|---|---|---|---|---|---|---|
| | $c$ | $\epsilon_{\max}$ | $c$ | $\epsilon_{\max}$ | $c$ | $\epsilon_{\max}$ | $c$ | $\epsilon_{\max}$ |
| F1 | 3.690 | 3.37E−06 | 3.654 | 8.59E−06 | 3.365 | 2.96E−07 | 3.024 | 4.19E−07 |
| F2 | 4.994 | 4.23E−05 | 4.270 | 1.06E−05 | 3.357 | 6.92E−07 | 4.402 | 1.60E−07 |
| F3 | 2.978 | 6.49E−07 | 3.656 | 2.05E−07 | 3.363 | 1.36E−08 | 3.024 | 6.27E−08 |
| F4 | 2.978 | 7.03E−07 | 3.655 | 1.56E−06 | 3.361 | 3.13E−08 | 2.820 | 7.80E−08 |
| F5 | 2.709 | 5.34E−07 | 3.657 | 1.48E−06 | 3.370 | 4.44E−08 | 2.917 | 4.93E−08 |
| F6 | 2.705 | 4.63E−06 | 3.214 | 5.87E−06 | 3.471 | 4.21E−07 | 3.024 | 1.92E−06 |

We would like to indicate that choosing the RBF functions is vital in our proposed localized method. The proposed method works only for those basis functions with a high convergence rate as shown in Table 5. Polyharmonic splines ($r^{2n}\ln r$) and polynomial RBFs ($r^{2n+1}$) fail to produce acceptable results. In Fig. 9, we show the maximum errors with respect to the shape parameters for various RBFs. Note that normalized MQ outperforms MQ. Overall, Gaussian, normalized MQ, and Matern RBFs have similar accuracy and stability.

To demonstrate that our proposed method is capable to handle large-scale problems, we test Algorithm 1 using $N = 122,500$ and $N_t = 117,500$. The results of the maximum error and CPU time are shown in Table 6. The CPU time is reasonable for handling such a large number of interpolation points.

In the last part of this example, we employ Algorithm 2 to perform the approximation. Here we switch from the Gaussian basis function to normalized MQ ($\sqrt{1 + (rc)^2}$). Note that the accuracy obtained in Table 7 is better than Table 2. However, even if

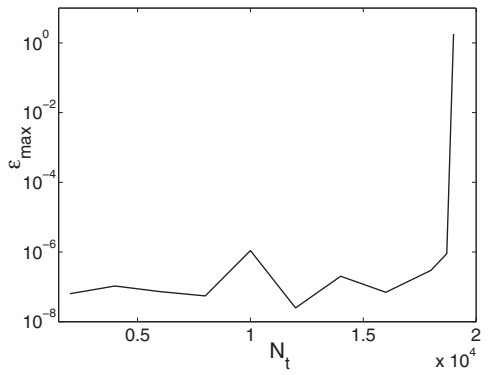


**Fig. 10.** The absolute maximum error verses $N_t$ using $N = 225,000$ and $N_s = 30$ for the F1 function.

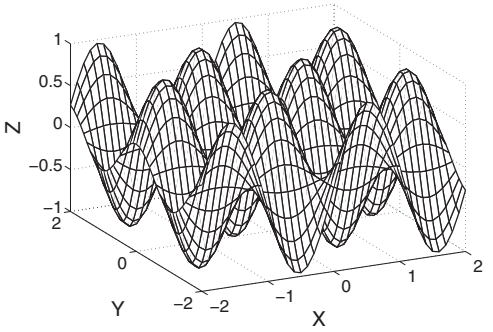


**Fig. 11.** The profile of test function F7.

**Table 8**
The absolute maximum error of function F7 and its derivative with respect to $x$ using $n_s = 50$.

| $(N, N_t)$ | $\|f - \hat{f}\|_\infty$ | $\|f_x - \hat{f}_x\|_\infty$ | RMSE($f_x$) | Shape parameter |
|---|---|---|---|---|
| $(50^2, 2000)$ | 8.28E−05 | 9.05E−03 | 4.26E−04 | 1.24 |
| $(100^2, 9000)$ | 1.55E−06 | 3.16E−04 | 1.15E−05 | 1.84 |
| $(150^2, 20,000)$ | 8.05E−07 | 1.29E−04 | 3.60E−06 | 2.40 |
| $(200^2, 35,000)$ | 2.40E−07 | 2.02E−04 | 3.37E−06 | 1.53 |

we use $n_s = 30$ for both algorithms, Algorithm 1 uses less than half of the neighboring points in the influence domains. Algorithm 1 is actually more efficient than Algorithm 2. But when the number of evaluation points is small, Algorithm 2 is more effective. Fig. 10 shows the performance of Algorithm 2 for F1 function using $N = 225,000$ and $n_s = 30$ with respect to the number of evaluation points $N_t$. The accuracy remains relatively accurate and stable for $N_t < 19,000$. This is a sharp contrast to Fig. 8 in terms of the number of evaluation points.

**Example 2.** In this example, we test Algorithm 1 for the approximation of the following function and its derivatives in the domain $[-2, 2]^2$:

Surface F7:

$$f(x, y) = \sin(3x)\cos(3y).$$

The profile of surface F7 is shown in Fig. 11. Note that the domain of this function is four times as large as surface F1 – F6 in the last example. We employ Algorithm 1 and Gaussian as the basis function in this example. We would like to indicate that the approximation of derivatives is more challenging than the approximation of functions. In Table 8, we show the absolute maximum errors of function F7 and the absolute maximum errors and root mean square errors (RMSE) of its derivative with respect to $x$ for various $N$ and $N_t$ using $n_s = 50$. The error of the derivative with respect to $y$ is similar to $x$, and we thus omit showing it. As we have noticed in Table 8, the approximation of $f_x$ is two to three orders less accurate than its counterpart $f$. The difficulty of approximating the derivatives using localized approaches is more pronounced than the global RBF approach.

## 5. Conclusions

In this paper, we proposed two algorithms using the concept of localization for the reconstruction of functions and their derivatives from a given set of scattered data using RBFs. Coupling very selective RBFs with the idea of creating an influence domain for each interpolation point, we are able to approximate the given function globally in the influence domain using a small number of neighboring points and then extend all of these local-global approximations to a global approximation through the formulation of a sparse matrix. As a result, the two proposed algorithms are highly efficient, stable, and capable of handling a large amount of scattered data. In this paper, with the limitation of using MATLAB, we have been able to use up to 225,000 interpolation points. Using other platforms such as Fortran or C++, we should be able to handle millions of data interpolation points. The proposed algorithms provide new approaches for solving large-scale interpolation problems.

The two proposed algorithms are suitable for handling different types of data. Algorithm 1 is more efficient than Algorithm 2. However, when the number of evaluation points is far less than the given interpolation data, Algorithm 2 is more effective.

The difficulties of global RBF interpolation, such as ill-conditioning of the resultant matrix and the determination of the shape parameters of RBFs, have been largely alleviated using the localized approximation schemes. In comparison with the CS-RBFs, which are localized RBF schemes, our proposed algorithms are far more efficient and accurate.

One limitation of the proposed schemes is that the number of evaluation points has to be less than the known interpolation points. Similar to other RBF interpolation schemes, the proposed approaches can be easily extended to higher dimensional problems. The implementation of the proposed algorithms to problems beyond 2D is currently under investigation.

## Acknowledgment

The fourth author acknowledges the financial support of the National Research Council in Taiwan under the project NSC 102-2917-I-564-042 .

## References

[1] R. Franke, Scattered data interpolation: tests of some methods, Math. Comput. 38 (1982) 181–200.
[2] H. Akima, A new method of interpolation and smooth curve fitting based on local procedures, J. ACM 17 (4) (1970) 589–602.
[3] J.C. Carr, Reconstruction and representation of 3D objects with radial basis functions, ACM SIGGRAPH (2011) 67–76.
[4] H. Wendland, Piecewise polynomial, positive definite and compactly supported radial functions of minimal degree, Adv. Comput. Math. 4 (1) (1995) 389–396.
[5] R. Beatson, J. Cherrie, C. Mouat, Fast fitting of radial basis functions: Methods based on preconditioned GMRES iteration, Adv. Comput. Math. 11 (1999) 253–270.
[6] R. Beatson, J. Cherrie, D. Ragozin, Fast evaluation of radial basis functions: methods for four-dimensional polyharmonic splines, SIAM J. Math. Anal. 32 (2001) 1272–1310.
[7] R. Beatson, L. Greengard, A Short Course on Fast Multipole Methods, Oxford University Press, 1997, pp. 1–37.
[8] R. Beatson, W. Light, Fast solution of the radial basis function interpolation equations: domain decomposition methods, SIAM J. Sci. Comput. 22 (2000) 1717–1740.
[9] G.E. Fasshuauer, J.G. Zhang, On choosing 'optimal' shape parameter for RBF approximation, Numer. Algorithms 45 (2007) 345–368.
[10] S. Rippa, An algorithm for selecting a good value for the parameter c in radial basis function interpolation, Adv. Comput. Math. 11 (2-3) (1999) 193–210.
[11] J. Bentley, Multidimensional binary search trees used for associative searching, CACM 18 (1975) 509–517.
[12] Z. Wu, Multivariate compactly supported positive definite radial functions, AICM 4 (1995) 283–292.
[13] M. Buhmann, Radial functions on compact support, Proc. Edinb. Math. Soc. 41 (1998) 33–46.
[14] M. Buhmann, Multivariate cardinal interpolation with radial-basis functions, Constr. Approx. 6 (1990) 225–255.
[15] R. Hardy, Multiquadric equations of topography and other irregular surfaces, J. Geophys. Res. 176 (1971) 1905–1915.
[16] J. Duchon, Fonctions-spline du type plaque mince en dimension 2, Université de Grenoble technical report 231, 1975.
[17] J. Duchon, Fonctions-spline à energie invatiante par rotation, Université de Grenoble technical report 27, 1976.
[18] C.S. Chen, Y.C. Hon, R.A. Schaback, Scientific computing with radial basis functions, Technical report, Department of Mathematics, University of Southern Mississippi, Hattiesburg, MS, USA, preprint, 2005.
[19] G. Fasshauer, Meshfree Approximation Method with MATLAB, World Scientific Publishers, Singapore, 2007.