Engineering Analysis with Boundary Elements 156 (2023) 240–250

Contents lists available at ScienceDirect

Engineering Analysis with Boundary Elements

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

# Polyharmonic splines interpolation on scattered data in 2D and 3D with applications

Kalani Rubasinghe [a,d], Guangming Yao [a,*], Jing Niu [b], Gantumur Tsogtgerel [c]

[a] Department of Mathematics, Clarkson University, Potsdam, NY 13699-5815, USA
[b] Department of Mathematics, Harbin Normal University, Harbin, China
[c] Department of Mathematics and Statistics, McGill University, Montréal, Québéc, H3A 2K6, Canada
[d] Department of Mathematics, State University of New York at Canton, Canton, NY 13617, USA

ARTICLE INFO

*Keywords:*
Radial basis functions
Interpolation
Scattered data
Parallel computing

ABSTRACT

Data interpolation is a fundamental problem in many applied mathematics and scientific computing fields. This paper introduces a modified implicit local radial basis function interpolation method for scattered data using polyharmonic splines (PS) with a low degree of polynomial basis. This is an improvement to the original method proposed in 2015 by Yao et al.. In the original approach, only radial basis functions (RBFs) with shape parameters, such as multiquadrics (MQ), inverse multiquadrics (IMQ), Gaussian, and Matern RBF are used. The authors claimed that the conditionally positive definite RBFs such as polyharmonic splines $r^{2n}\ln r$ and $r^{2n+1}$ "failed to produce acceptable results".

In this paper, we verified that when polyharmonic splines together with a polynomial basis is used on the interpolation scheme, high-order accuracy and excellent conditioning of the global sparse systems are gained without selecting a shape parameter. The scheme predicts functions' values at a set of discrete evaluation points, through a global sparse linear system. Compared to standard implementation, computational efficiency is achieved by using parallel computing. Applications of the proposed algorithms to 2D and 3D benchmark functions on uniformly distributed random points, the Halton quasi-points on regular or Stanford bunny shape domains, and an image interpolation problem confirmed the effectiveness of the method. We also compared the algorithms with other methods available in the literature to show the superiority of using PS augmented with a polynomial basis. High accuracy can be easily achieved by increasing the order of polyharmonic splines or the number of points in local domains, when small order of polynomials are used in the basis. MATLAB code for the 3D bunny example is shared on MATLAB Central File Exchange (Yao, 2023).

## 1. Introduction

Scattered data interpolation is a common and fundamental problem in many scientific and engineering studies [1–7]. There has already been so much work [8] in this area yet interpolation is still a difficult and computationally expensive problem. Polynomial interpolation and piece-wise polynomial splines have been generally used until 1971, Rolland Hardy introduced an interpolation method based on multiquadric (MQ) radial basis function (RBF) [9]. Since then, many different RBF interpolation schemes have been developed. A few different methods can be found in [10–12], and a comparison of radial basis functions methods can be found in [13].

Despite the simplicity, applicability to various kinds of problems, and effectiveness of RBF-based methods, the resultant system of equations is often ill-conditioned when there is a large number of data points. Apart from that, global interpolation methods based on RBFs suffer from typical drawbacks of a global method such as high memory requirement and high computational cost. In fact, global RBF methods produce a linear system of size large as the number of data points. Generally, the costs of direct solution of such systems are $\mathcal{O}(N^3)$ and $\mathcal{O}(N^2)$ memory usage. There are several methods that get around these issues such as domain decomposition [14,15], accelerated iterated approximate moving least squares [16], RBF-QR algorithms [17], a compactly supported RBFs [18], radial basis function-finite difference method [19], and many others. One disadvantage of the domain decomposition methods is that the domain discretization and joining phase of the local interpolants are not easy to implement. Recently, Hansen shared a toolbox "regtools" including regtoolsTSVD, Tikhonov, and LSQR regularization methods for analysis and solution of discrete ill-posed problems [20].

* Corresponding author.
*E-mail addresses:* gyao@clarkson.edu (G. Yao), njirwin@163.com (J. Niu).

https://doi.org/10.1016/j.enganabound.2023.08.001
Received 1 March 2023; Received in revised form 1 August 2023; Accepted 2 August 2023

The two methods proposed in [21] are fast localized RBF algorithms for large-scale 2D scattered data interpolation. In these methods, an interpolation is performed on each local influence domain and then all the influence domains are combined into a sparse matrix with the use of RBFs like Gaussian, MQ, Normalized MQ, and Matern. These methods are categorized as implicit methods in such a way that the interpolant is not explicitly defined by the numerical approximation, instead the approximation at sets of discrete evaluation points is given. We will continue to use such characterization throughout our work. On the other hand, if the interpolation function is defined by the numerical schemes, we call it an explicit method.

The authors in [21] claimed that the conditionally positive definite RBFs such as polyharmonic splines $r^{2n}\ln r$ and $r^{2n+1}$ "failed to produce acceptable results". However, in this paper, we verified that when polyharmonic splines together with a polynomial basis is used on the implicit interpolation scheme, high-order accuracy, and excellent conditioning of the global sparse systems are gained without the need for selecting a shape parameter. The performance of the proposed algorithm is even more accurate than the most recent publication [7] in 2023. Although [7] is a global interpolation scheme, our algorithms are local.

One may argue the proposed algorithm is the same as the idea of the Radial Basis Function-Finite Difference Method (RBF-FD) [19] when applied to interpolation problems. However, there are fundamental differences between the two methods: (1) Our method is an implicit method, where only approximations at a set of discrete evaluation points are produced, but RBF-FD is an explicit interpolation in which an interpolation function is given by the numerical scheme; (2) Our method constructs the local domains by searching within the interpolation points or union of interpolation and evaluation points, but the RBF-FD constructs local domains purely within the evaluation points.

In Section 2, we briefly introduced global RBF interpolation and the positive-definiteness of the RBFs. In Section 3, we propose the localized RBF interpolation methods for scattered data interpolation in $\mathbb{R}^d$, where $d$ is a positive integer. We categorize this algorithm as a Local Implicit Interpolation using Polyharmonic Splines and Polynomials (LI2Poly2) Algorithm 1 and Algorithm 2 based on two different ways of creating local domains. Section 4 dedicates to numerical experiments carried on with 2D scattered data followed by a few 3D experiments. Numerical results are compared with the implicit local RBF method [21] and the CS-RBF method. The performance of the proposed method is demonstrated regarding accuracy, efficiency, and parameter selection. In Section 5, we draw some conclusions on the usage of polyharmonic spline basis in the proposed method and possible improvements.

## 2. Radial basis function interpolation

The problem of scattered data interpolation is, given a set of distinct data points $\mathbf{x}_i \in \mathbb{R}^d$ with associated data values $\mathbf{y}_i \in \mathbb{R}$ for $i = 1, 2, \dots, N$, find an interpolant function $\hat{f}(\mathbf{x}) : \mathbb{R}^d \to \mathbb{R}$, satisfying $\hat{f}(\mathbf{x}_i) = \mathbf{y}_i$, $i = 1, \dots, N$.

The global radial basis function interpolant $\hat{f}$ is given by

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

where $\phi(r)$ is a radial basis function with the $r = \|\mathbf{x} - \mathbf{x}_j\| \geq 0$ defined as the Euclidean distance, and $\mathbf{x}_j$ is the center of the RBF. Note that $r$ is the distance between point $\mathbf{x}$ and the centers of the basis functions.

The unknown real coefficients $\alpha_j, j = 1, 2, \dots, N$ are determined by enforcing the interpolation condition $\hat{f}(\mathbf{x}_i) = \mathbf{y}_i$ :

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

The resulting $N \times N$ linear system of equations can be represented by a matrix form

$$A\alpha = b,$$

**Table 1**
The dimension of the basis $\{p_1, p_2, \dots, p_q\}$ where $p_i$, $i = 1, \dots, q$ are polynomials of degree up to $m-1$ in $d$ dimension.

| $m-1$ | $d=1$ | $d=2$ | $d=3$ |
|---|---|---|---|
| 0 | 1 | 1 | 1 |
| 1 | 2 | 3 | 4 |
| 2 | 3 | 6 | 10 |
| 3 | 4 | 10 | 20 |
| 4 | 5 | 15 | 35 |
| 5 | 6 | 21 | 56 |
| 6 | 7 | 28 | 84 |

where $\alpha = [\alpha_1, \alpha_2, \dots, \alpha_N]^T$, $b = [\mathbf{y}_1, \mathbf{y}_2, \dots, \mathbf{y}_N]^T$, and entries of $A$ are given by $a_{i,j} = \phi(\|\mathbf{x}_i - \mathbf{x}_j\|)$, $1 \leq i, j \leq N$. The solution of the above interpolation problem exists and is unique, if and only if the interpolation matrix $A$ is nonsingular, which is true for certain choices of RBFs that are positive definite.

**Theorem 2.1** ([12]). *A real-valued continuous function $\phi$ is positive definite on $\mathbb{R}^d$ if and only if it is even and*

$$\sum_{i=1}^{N}\sum_{j=1}^{N} \alpha_i \alpha_j \phi(\mathbf{x}_i - \mathbf{x}_j) \geq 0, \tag{3}$$

*for any $N$ distinct data points $\mathbf{x}_1, \dots, \mathbf{x}_N \in \mathbb{R}^d$ and $\alpha = (\alpha_1, \dots, \alpha_N)^T \in \mathbb{R}^N$.*

Radially symmetric RBFs are clearly even functions. If $\phi(r)$ is a positive definite function, it can be proved that the interpolation matrix $A$ is a positive definite matrix for any distinct points $\mathbf{x}_1, \dots, \mathbf{x}_N$ making it nonsingular. For example, Gaussian, inverse multiquadrics (IMQ), Matern, and compactly-supported RBFs (CS-RBFs) are positive definite functions [22,23]. Other commonly used RBFs such as multiquadrics (MQ) and polyharmonic splines (PS) on the other hand have only been shown to be conditionally positive definite.

**Definition 2.2** ([22]). A real-valued continuous even function $\phi$ is called conditionally positive definite of order $m$ on $\mathbb{R}^d$ if

$$\sum_{i=1}^{N}\sum_{j=1}^{N} \alpha_i \alpha_j \phi(\mathbf{x}_i - \mathbf{x}_j) \geq 0, \tag{4}$$

for any $N$ distinct data points $\mathbf{x}_1, \dots, \mathbf{x}_N \in \mathbb{R}^d$ and $\alpha = (\alpha_1, \dots, \alpha_N)^T \in \mathbb{R}^N$ satisfying

$$\sum_{j=1}^{N} \alpha_j p(\mathbf{x}_j) = 0 \tag{5}$$

for any real valued polynomial $p$ of degree at most $m-1$. The function $\phi$ is called strictly conditionally positive definite of order $m$ on $\mathbb{R}^d$ if the points $\mathbf{x}_1, \dots, \mathbf{x}_N \in \mathbb{R}^d$ are distinct, and $\alpha \neq \mathbf{0}$ implies strict inequality in (4).

When using radial basis functions that are conditionally positive definite, one has to add polynomial basis functions of a certain maximal degree to the interpolate function (1) as follows

$$\hat{f}(\mathbf{x}) = \sum_{j=1}^{N} \alpha_j \phi(\|\mathbf{x} - \mathbf{x}_j\|) + \sum_{k=1}^{q} \beta_k p_k(\mathbf{x}) \tag{6}$$

where $\{p_1, p_2, \dots, p_q\}$ forms a basis for $\mathcal{P}^d_{m-1}$, space of polynomials of total degree less than or equal to $m-1$ in $d$-variables, where $q = \binom{d+m-1}{d}$. See Table 1 for some examples of the values of $q$ in $d$ dimension.

The new linear system is created by enforcing the interpolation condition $\hat{f}(\mathbf{x}_i) = \mathbf{y}_i$ for $i = 1, \dots, N$. We also consider the following additional insolvency constraints for the polynomial part to guarantee a unique solution for the new linear system:

$$\sum_{j=1}^{N} \alpha_j p_k(\mathbf{x}_j) = 0, \quad k = 1, \dots, q. \tag{7}$$

Thus, the following linear system can be achieved by combining the interpolation condition and the additional insolvency constraints (7):

$$\begin{pmatrix} A & P^T \\ P & \mathbf{0} \end{pmatrix} \begin{pmatrix} \alpha \\ \beta \end{pmatrix} = \begin{pmatrix} b \\ \mathbf{0} \end{pmatrix}, \tag{8}$$

where $\alpha = [\alpha_1, \alpha_2, \dots, \alpha_N]^T$, $\beta = [\beta_1, \beta_2, \dots, \beta_q]^T$, $b = [\mathbf{y}_1, \mathbf{y}_2, \dots, \mathbf{y}_N]^T$, and entries of $A$ are given by $a_{i,j} = \phi(\|\mathbf{x}_i - \mathbf{x}_j\|)$, $1 \le i, j \le N$ and entries of $P$ are given by $p_{i,j} = p_j(\mathbf{x}_i)$, $1 \le i \le N, 1 \le j \le q$. Once the unknown coefficients are obtained by solving the resulting $(N + q) \times (N + q)$ linear system, the interpolations can be made. Global interpolation methods based on RBF are easy to implement but, when the number of collocation points within the domain is high, the resulting matrix suffers from ill-conditioning and problems associated with computational cost including storage and time. On the other hand, adding polynomials in a global context can lead to other drawbacks associated with polynomial interpolation. With the limitations of global methods, formulations of localized methods offer an alternative for large-scale realistic data. In the next section, we present two local methods which have similar localization procedures as the generalized finite difference method.

## 3. Localized RBF interpolation based on polyharmonic splines

The two methods proposed in [21] are fast localized RBF algorithms for large-scale 2D scattered data interpolation. In these methods, an interpolation is performed on each local influence domain and then all the influence domains are combined into a sparse matrix with the use of RBFs like Gaussian, MQ, Normalized MQ, and Matern. All of these RBFs come with a free shape parameter that needs to be chosen carefully in the interpolation process. Yet, there is no practical and theoretical procedure for choosing the optimal shape parameter in applications except for some efforts made [24–27]. It is true that smooth RBFs like MQ give more accurate results at smaller values of the shape parameter $c$. But there is a trade-off between accuracy and conditioning. As $c$ decreases function becomes flatter and accuracy increases until numerical ill-conditioning steps in. There are ways to improve the performance of MQ RBF methods, such as employing fictitious nodes [28,29], pre-conditioning [15], etc [30–32].

We have improved two local methods in [21] by incorporating shape parameter-free polyharmonic splines (PS RBF) together with polynomial basis functions. This is an extension to the current methods, achieving high accuracy without the need of selecting a shape parameter. It is known that when high-order polynomials are used in global methods can lead to Runge's phenomenon. Such effects are alleviated simply in the local context as one is only interested in interpolation within a small neighborhood.

For convenience, let us consider the interpolation problem in two-dimensional spaces. Suppose we have a set of distinct scattered data points $\{\mathbf{x}_i\}_{i=1}^N \subset \mathbb{R}^2$ and their function values $\{f(\mathbf{x}_i)\}_{i=1}^N \subset \mathbb{R}$. Let $\{\mathbf{z}_j\}_{j=1}^{N_t} \subset \mathbb{R}^2$ be a set of evaluation points. We try to find the interpolant $\hat{f}$ such that $\hat{f}(\mathbf{z}_j) \approx f(\mathbf{z}_j)$, $j = 1, \dots, N_t$ and $\hat{f}(\mathbf{x}_i) = f(\mathbf{x}_i)$, $i = 1, \dots, N$.

The polyharmonic splines in $\mathbb{R}^d$ are defined as follows:

$$\phi_{d,k}(r) = \begin{cases} r^{2k-d} \ln(r), & \text{for } d \text{ even} \\ r^{2k-d}, & \text{for } d \text{ odd} \end{cases}$$

where $2k > d$, and with $k \in \mathbb{N}$. For example, in $\mathbb{R}^2$, $\phi_{2,3}(r) = r^4 \log(r)$, then $d = 2, k = 3$, thus polynomials up to degree 2 or higher need to be added to ensure the unique solvability.

### 3.1. LI2Poly2-Algorithm 1

For each $\mathbf{x}_i$, we choose $n$ nearest evaluation points to $\mathbf{x}_i$ to create a local influence domain $\Omega_i = \{\mathbf{z}_j^{[i]}\}_{j=1}^n$, in which $j = 1 \dots n$ denotes for local indexing for each node in the $\Omega_i$. Note that $n \ll N$. In this construction, each interpolation point has a local influence domain that contains only $n$ evaluation points. Fig. 1 shows an example of local domain construction with $n = 5$.

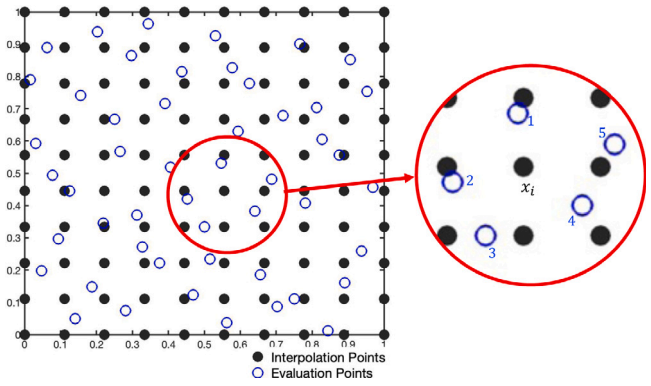



**Fig. 1.** Local influence domain of $\mathbf{x}_i$ with five nearest evaluation points in Algorithm 1.

Now let us focus on the RBF interpolation on the local domain $\Omega_i$, and let $\mathbf{z} = \mathbf{z}_j^{[i]} \in \Omega_i$ for some $j \le n$. Thus, $\hat{f}(\mathbf{z})$ can be written as previously shown in (6) which is the following,

$$\hat{f}(\mathbf{z}) = \sum_{j=1}^{n} \alpha_j \phi(\|\mathbf{z} - \mathbf{z}_j^{[i]}\|) + \sum_{l=1}^{q} \alpha_{n+l}\, p_l(\mathbf{z}).$$

where $\phi$ is chosen to be polyharmonic splines RBF and $\{p_1, p_2, \dots, p_q\}$ forms a basis for polynomials up to degree less than or equal to $m - 1$ in $\mathbb{R}^d$, $q = \binom{d+m-1}{d}$. Notice that $m = k - \lceil d/2 \rceil + 1$ is the order of the conditionally positive definite polyharmonic splines $\phi_{d,k}$ [33]. In addition, the size of the local domain should be greater than the number of polynomial basis functions $q = \binom{d+m-1}{d}$ for this to work. In this example, the local domain size needs to be larger than $q = \binom{2+2}{2} = 6$ in $\mathbb{R}^2$.

Collocation on the local domain of influence leads to the following system:

$$\sum_{j=1}^{n} \alpha_j \phi(\|\mathbf{z}_k^{[i]} - \mathbf{z}_j^{[i]}\|) + \sum_{l=1}^{q} \alpha_{n+l}\, p_l(\mathbf{z}_k^{[i]}) = \hat{f}(\mathbf{z}_k^{[i]}), \quad k = 1, 2, \dots n, \tag{9}$$

$$\sum_{j=1}^{n} \alpha_j\, p_l(\mathbf{z}_k^{[i]}) = 0, \quad l = 1, 2, \dots q. \tag{10}$$

Let the coefficient matrices on the first and second terms of the left-hand side of (9) be $\boldsymbol{\Phi}$ and $\boldsymbol{P}$ respectively. That is

$$\boldsymbol{\Phi} = \begin{pmatrix} \phi(\|\mathbf{z}_1^{[i]} - \mathbf{z}_1^{[i]}\|) & \dots & \phi(\|\mathbf{z}_1^{[i]} - \mathbf{z}_n^{[i]}\|) \\ \phi(\|\mathbf{z}_2^{[i]} - \mathbf{z}_1^{[i]}\|) & \dots & \phi(\|\mathbf{z}_2^{[i]} - \mathbf{z}_n^{[i]}\|) \\ \vdots & \vdots & \vdots \\ \phi(\|\mathbf{z}_n^{[i]} - \mathbf{z}_1^{[i]}\|) & \dots & \phi(\|\mathbf{z}_n^{[i]} - \mathbf{z}_n^{[i]}\|) \end{pmatrix}, \text{ and}$$

$$\mathbf{P} = \begin{pmatrix} 1 & x_1 & y_1 & \dots & y_1^{m-1} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & x_n & y_n & \dots & y_n^{m-1} \end{pmatrix}. \tag{11}$$

Then we can introduce a block matrix from for the system (9)–(10) as follows:

$$\begin{pmatrix} \boldsymbol{\Phi} & \boldsymbol{P} \\ \boldsymbol{P}^T & \mathbf{0} \end{pmatrix} \boldsymbol{\alpha}^{[i]} = \begin{pmatrix} \hat{\mathbf{f}}_n \\ \mathbf{0} \end{pmatrix}, \tag{12}$$

where $\boldsymbol{\alpha}^{[i]} = [\alpha_1, \alpha_2, \dots, \alpha_{n+q}]^T$, $\hat{\mathbf{f}}_n = [\hat{f}(\mathbf{z}_1^{[i]}), \hat{f}(\mathbf{z}_2^{[i]}), \dots, \hat{f}(\mathbf{z}_n^{[i]})]^T$. Let the coefficient matrix in (12) be $\boldsymbol{\Psi}$. Then the unknown coefficients in (9)–(10) can be expressed as

$$\boldsymbol{\alpha}^{[i]} = \boldsymbol{\Psi}^{-1} \begin{pmatrix} \hat{\mathbf{f}}_n \\ \mathbf{0} \end{pmatrix}. \tag{13}$$

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

$$f(\mathbf{x_i}) = \sum_{j=1}^{n} \alpha_j \phi(\|\mathbf{x}_i - \mathbf{z}_j^{[i]}\|) + \sum_{l=1}^{q} \alpha_{n+l}\, p_l(\mathbf{x}_i),$$

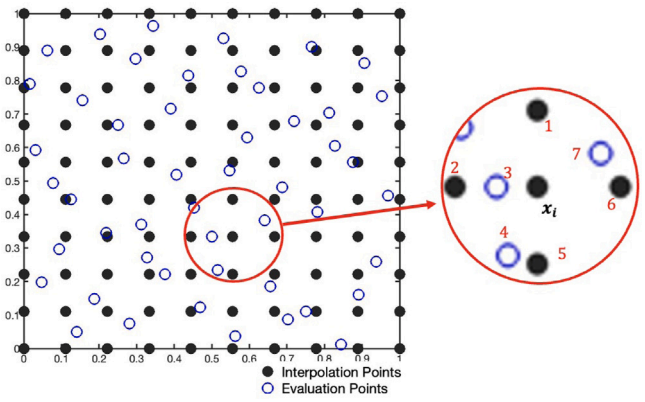



**Fig. 2.** Local influence domain consists of seven nearest points to $x_i$ in Algorithm 2.

$$
\begin{aligned}
&= \boldsymbol{\Phi}(\mathbf{x}_i)\boldsymbol{\alpha}^{[i]} \\
&= \boldsymbol{\Phi}(\mathbf{x}_i)\boldsymbol{\Psi}^{-1}\begin{pmatrix} \hat{\mathbf{f}}_n \\ \mathbf{0} \end{pmatrix} \\
&= \boldsymbol{\Lambda}_{n+q}(\mathbf{x}_i)\begin{pmatrix} \hat{\mathbf{f}}_n \\ \mathbf{0} \end{pmatrix} \\
&= \boldsymbol{\Lambda}_n(\mathbf{x}_i)\hat{\mathbf{f}}_n, \qquad (14)
\end{aligned}
$$

where

$$
\begin{aligned}
\boldsymbol{\Phi}(\mathbf{x}_i) = \Big[&\phi(\|\mathbf{x}_i - \mathbf{z}_1^{[i]}\|), \dots, \phi(\|\mathbf{x}_i - \mathbf{z}_n^{[i]}\|), 1, x, y, x^2, xy, y^2, \dots, \\
&x^m, x^{m-1}y, \dots, xy^{m-1}, y^m\Big],
\end{aligned}
$$

and $\boldsymbol{\Lambda}_{n+q}(\mathbf{x}_i) = \boldsymbol{\Phi}(\mathbf{x}_i)\boldsymbol{\Psi}^{-1}$. Note that $\boldsymbol{\Lambda}_n(\mathbf{x}_i)$ is obtained by omitting last $q$ elements of the vector $\boldsymbol{\Lambda}_{n+q}(\mathbf{x}_i)$.

The system (14) can be reformulated to an $N \times N_t$ sparse system easily by extending local $\hat{\mathbf{f}}_n$ to a global $\hat{\mathbf{f}}_{N_t} = [\hat{f}(\mathbf{z}_1), \hat{f}(\mathbf{z}_2), \dots, \hat{f}(\mathbf{z}_{N_t})]^T$. This can be done by inserting zeros to $\boldsymbol{\Lambda}_n(\mathbf{x}_i)$ based on the mapping between $\hat{\mathbf{f}}_n$ and $\hat{\mathbf{f}}_{N_t}$. It follows that

$$
\mathbf{f}(\mathbf{x}_i) = \boldsymbol{\Lambda}_{N_t}(\mathbf{x}_i)\hat{\mathbf{f}}_{N_t} \qquad (15)
$$

where $\boldsymbol{\Lambda}_{N_t}(\mathbf{x}_i)$ is a vector that is obtained by adding zeros into $\boldsymbol{\Lambda}_n(\mathbf{x}_i)$ appropriate at places.

For example, assume $N_t = 50, n = 3$, and $\Omega_i = \{\mathbf{z}_1^{[i]}, \mathbf{z}_2^{[i]}, \mathbf{z}_3^{[i]}\} = \{\mathbf{z}_{12}, \mathbf{z}_{20}, \mathbf{z}_{23}\}$. Then we need to insert 47 zeros into $\boldsymbol{\Lambda}_n(\mathbf{x}_i)$, while only keeping non zero values at 12th, 20th and 23rd positions. i.e, $\boldsymbol{\Lambda}_{N_t}(\mathbf{x}_i) = [0, 0, \dots, \underbrace{\#}_{12^{th}}, 0, 0, \dots, 0, \underbrace{\#}_{20^{th}}, 0, 0, \underbrace{\#}_{23^{rd}}, 0, 0, \dots, \underbrace{0}_{50^{th}}]$. By solving (15), the unknown approximation function values at $N_t$ evaluation points, $\hat{f}(\mathbf{z}_j)$, $j = 1, 2, \dots, N_t$ can be found. This can be done using any direct or iterative methods for solving systems of linear equations given that $N_t \le N$. The proposed algorithm works for any dimension $d$ like many other RBF-based methods.

### 3.2. LI2Poly2-Algorithm 2

Algorithm 2 is developed with the idea of taking all the interpolation points and evaluation points in each local neighborhood into consideration for the interpolation at a single point. In this construction, each interpolation point will have a local influence domain that contains both evaluation points and interpolation points, total into $n$. Fig. 2 shows an example of local domain construction with $n = 7$.

For each $\mathbf{x}_i$, we choose the nearest $n_1$ evaluation points and $n_2$ interpolation points which adds up to $n$ to create a local influence domain $\Omega_i = \{\mathbf{z}_j^{[i]}\}_{j=1}^{n_1} \bigcup \{\mathbf{x}_j^{[i]}\}_{j=1}^{n_2}$. The local interpolation procedure presented in Algorithm 1 resulted in (15) should be changed to account for the changes in Algorithm 2. The resulting system of linear equations is size $N \times (N_t + N)$ and is underdetermined as follows,

$$
\mathbf{f}(\mathbf{x}_i) = \boldsymbol{\Lambda}_{N_t+N}(\mathbf{x}_i)\begin{pmatrix} \hat{\mathbf{f}}_{N_t} \\ \mathbf{f}_\mathbf{N} \end{pmatrix}, \; i = 1, \dots, N \qquad (16)
$$

where $\mathbf{f}_\mathbf{N} = [f(\mathbf{x}_1), \dots, f(\mathbf{x}_N)]^T$. In order to solve this system, the system has been reformulated to a $2N \times (N_t + N)$ system as follows

$$
\begin{pmatrix} \boldsymbol{\Lambda}_{N_t+N} \\ \mathbf{0}_{N\times N_t} \quad \mathbf{I}_{N\times N} \end{pmatrix}\begin{pmatrix} \hat{\mathbf{f}}_{N_t} \\ \mathbf{f}_\mathbf{N} \end{pmatrix} = \begin{pmatrix} \mathbf{f}_\mathbf{N} \\ \mathbf{f}_\mathbf{N} \end{pmatrix}. \qquad (17)
$$

The approximated function values $\hat{\mathbf{f}}_{N_t}$ can be obtained by solving (17) using any system-solving technique.

The efficiency of the method can be illustrated by considering its asymptotic computational complexity. For both algorithms, we need to

Step 1. For each interpolation point, we need to calculate the coefficient matrix in (14) by

- first, build kd-tree among the all $N_t$ evaluation points for Algorithm 1 or for all $N_t$ evaluation points and $N$ interpolation points for Algorithm 2;
- second, find $n$ nearest neighbors of all $N$ interpolation points using the kd-tree created above;
- third, solve small local systems of size $(n+q)\times(n+q)$, where there are $N$ such systems;

Step 2. Solve a sparse system of size $N \times N$, in which $n$ unknowns in each equation.

Fig. 3 shows detailed computational complexity associated with both algorithms. Note that there are $N_t$ evaluation points, $N$ interpolation points, and $q$ is the number of polynomial basis. Thus, there are $N$ small systems of size $(n + q) \times (n + q)$ in Algorithm 1. The cost for solving these $N$ small systems is $\mathcal{O}(N(n + q)^3)$. However, $n, q \ll N$ in practice. Hence, the computational complexity in time for Algorithm 1 is $\mathcal{O}((N_t + nN)log(N_t)) + \mathcal{O}(N(n + q)^3) + \mathcal{O}(Nn^2)$ and the computational complexity of Algorithm 2 is $\mathcal{O}(((n + 1)N + N_t)log(N + N_t)) + \mathcal{O}(N(n + q)^3) + \mathcal{O}((N + N_t)n^2)$.

In the next section, we demonstrate the accuracy and efficiency of the proposed method using two-dimensional data intensively and some results from the three-dimensional data as well. The stability of the method is also inspected by studying the condition number of the interpolation matrices and global sparse matrices.

## 4. Numerical results

To illustrate the effectiveness of the method on scattered data, we consider examples of two and three dimensions on both regular and irregular domains. Recall the following key parameters/notations:

$N$ : the number of interpolation points
$N_t$: number of evaluation (test) points
$n$ : the number of points in the local domain of influence
$m$ : the degree of highest-order polynomials
$k$ : the order of PS.

Numerical results are compared with the exact function values and the accuracy of the method is measured in terms of root mean squared error ($\epsilon_{rms}$) and the maximum absolute error ($\epsilon_{max}$), whose are given by

$$
\epsilon_{rms} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(\hat{f}_i - f_i)^2}, \qquad \epsilon_{max} = \max_{i=1}^{N}|\hat{f}_i - f_i| \qquad (18)
$$

where $\hat{f}_i = \hat{f}(\mathbf{x}_i)$ is the approximated value of $f_i = f(\mathbf{x}_i)$. The condition number of the interpolation matrix $A$ is defined as

$$
\text{cond}(A) = \|A\|_2\|A^{-1}\|_2 = \frac{\sigma_{max}}{\sigma_{min}} \qquad (19)
$$

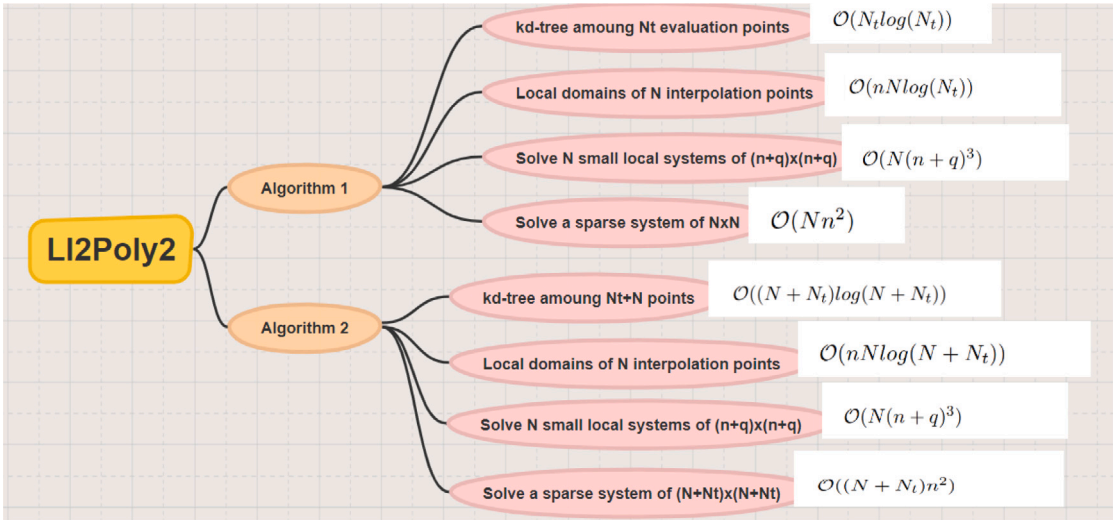
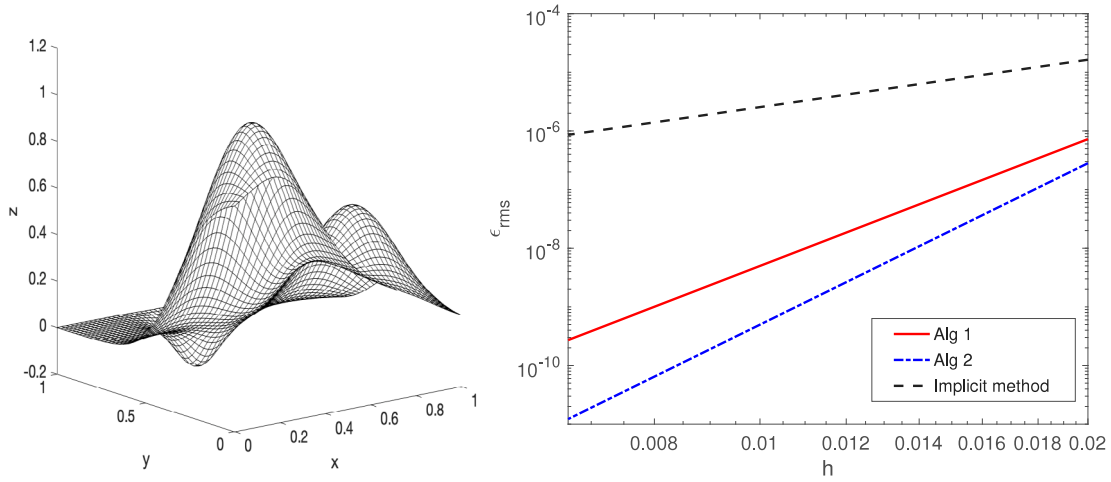



**Fig. 3.** Computational complexity in time with regards LI2Poly2 Algorithm 1 and Algorithm 2.




(a) Profile of test function $F_1$; (b) Rate of convergence w.r.t spatial variables.

**Fig. 4.** Franke's benchmark test function $F_1$ [34] on the left, and the root mean squared errors, $\epsilon_{rms}$, versus average distance between nodes, $h$, for $F_1$.

where $\sigma_{max}$ and $\sigma_{min}$ are the largest and smallest singular values of A. In our numerical experiments, we choose the interpolation points to be evenly distributed, while the evaluation points (test points) are randomly distributed Halton quasi-points with the constraint of $N_t < N$. In a case of $N \leq N_t$, we split the evaluation points into subsets and perform the interpolation separately. We will explain how interpolation is done in such cases later in this section.

All numerical experiments have been performed in MATLAB on a MacBook Pro with a 3.2 GHz Apple M1 processor and 16 GB memory. The algorithm code has been parallelized using the Matlab Parallel Computing Toolbox for improving the performance. Moreover, construction of the local domains by searching the nearest evaluation points has been done using the Matlab built-in function *knnsearch* from the statistics and machine learning toolbox.

**Example 4.1.** In the first example, we investigate the performances of the proposed methods on Franke's six test functions [34] on the unit square $[0,1]\times[0,1]$. For simplicity, we mainly show details on the test function $F_1$, the algorithms behave in a similar way on all other five test functions. The test function $F_1$ is provided as follows.

$$F_1(x,y) = \frac{3}{4}\exp\left(-\frac{1}{4}\left((9x-2)^2+(9y-2)^2\right)\right)$$
$$+\frac{3}{4}\exp\left(-\frac{1}{49}(9x+1)^2-\frac{1}{10}(9y+1)^2\right)$$
$$+\frac{1}{2}\exp\left(-\frac{1}{4}\left((9x-7)^2+(9y-3)^2\right)\right)$$
$$-\frac{1}{5}\exp\left(-(9x-4)^2-(9y-7)^2\right).$$

The left of Fig. 4 shows the profile of test function $F_1$. On the right of Fig. 4, the rate of convergence is displayed for the two proposed algorithms and the reference implicit method, with respect to spatial discretization. From the figure, it can be seen that Algorithm 2 has the highest convergence rate with the highest accuracy.

One of the main advantage of PS is that it does not have a shape parameter that needs to be determined during the interpolation process. However, we still need to select other parameters such as the order of the PS RBF ($k$), the order of the polynomial basis ($m$), and the number of local points ($n$). Since both algorithms behave similar and Algorithm 2 performs better in terms of accuracy without loss of computational efficiency, we examine the performance of Algorithm 1 (the worse case scenario) as we vary $k, m$, and $n$. Note that the findings presented here are comparable to those of Algorithm 2.

From the left of Fig. 5, we can see that interpolation error decreases as the number of interpolation points increases, and three separate lines

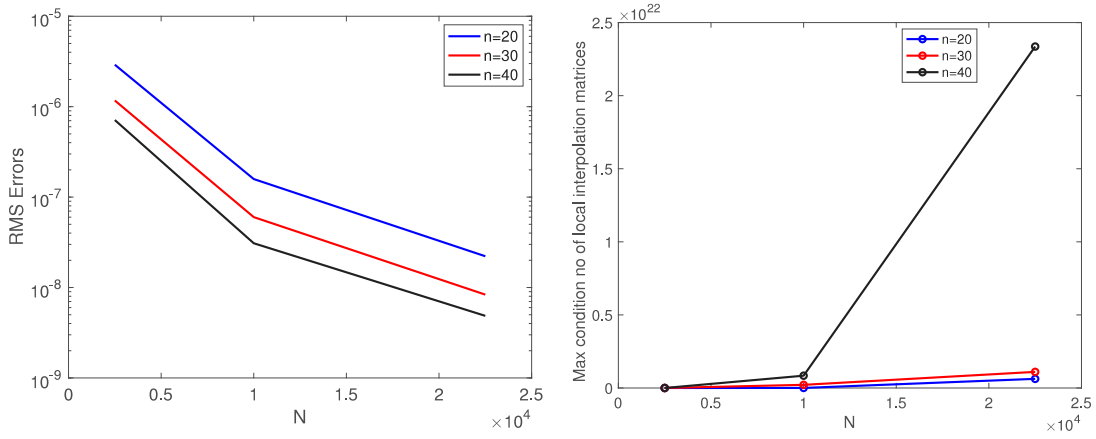



**Fig. 5.** RMS errors versus $N$ (left) and maximum condition number of local interpolation matrices versus $N$ (right) for various $n$ on $F_1$ with $k = 4, m = 3$ in Example 4.1 using Algorithm 1.

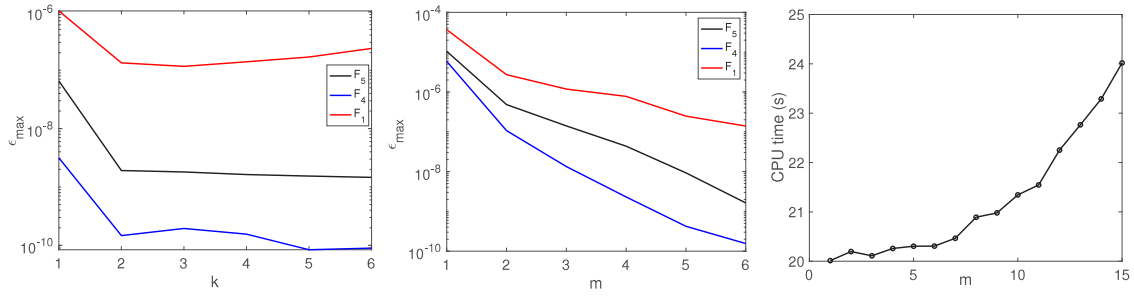



**Fig. 6.** Left: maximum errors versus the order of PS $k$ for $N = 100^2$, $N_t = 9000$, $n = 30$ and $m = 6$ using different test functions and Algorithm 1. Middle: maximum errors versus the polynomial order $m$ for $N = 100^2$, $N_t = 9000$, $n = 30$ and $k = 4$ using different test functions. Right: CPU time (in seconds) versus $m$ for test function $F_1$ using Algorithm 1 in Example 4.1.

indicate that one may obtain even more accurate results by increasing the number of points in the local domains. From the right of Fig. 5, we observe that the ill-conditioning grows as the number of interpolation points increases, although the error decreases with increasing $N$ as shown in the left of Fig. 5.

The left of Fig. 6 indicates that the accuracy does not change significantly when the order of PS changes (about one order of magnitude differences). In the middle of Fig. 6, it indicates that when the order of the polynomial basis $m$ increases, the accuracy improves. Yet, one would not need to increase the order of the basis too much as it will lead to a high computational cost as evident from the right of Fig. 6. However, we can reduce the simulation time significantly by using parallel computing while calculating the weights associated with each node in the global sparse system. For instance, when using $k = 4, m = 6, N = 40{,}000$ for $F_1$, the CPU time is reduced from 1050 s to 250 s. Note that the results from Franke's test functions $F_4$ and $F_5$ [34] are also included, and the test functions are presented below for reference: $F_4(x,y) = \frac{1}{3}\exp\left[-\frac{81}{16}\left(\left(x-\frac{1}{2}\right)^2+\left(y-\frac{1}{2}\right)^2\right)\right]$ and $F_5(x,y) = \frac{1}{3}\exp\left[-\frac{81}{4}\left(\left(x-\frac{1}{2}\right)^2+\left(y-\frac{1}{2}\right)^2\right)\right]$. Due to the relative smoothness of those two functions in comparison with $F_1$, the accuracy of interpolation results from those two functions is much better than those from $F_1$.

To validate the stability of the methods, we examine the sensitivity of the algorithms with respect to the locations of the points. Here we consider the perturbed grid points provided by

$$\tilde{x_i} = x_i + \rho x_i^{rand}\ \delta x$$
$$\tilde{y_i} = y_i + \rho y_i^{rand}\ \delta y$$

where $(x_i, y_i) \in (0,1)^2$ are uniformly distributed grid points. Let $\rho$ denote the degree of randomness and $\delta x, \delta y$ are the shortest distance between two points in $x$ and $y$ directions. Lastly, $x_i^{rand}, y_i^{rand}$ are uniformly distributed random numbers in $(0,1)$. Fig. 7 shows the RMS error obtained using Algorithm 1 and Algorithm 2 on the left and on the right, respectively, for various degrees of randomness. When $\rho$ is zero, the points are evenly distributed and as $\rho$ increases, the interpolation points become more distorted. However, the error does not change much for all three test functions. Hence, the stability with regard to the point distribution is validated.

Table 2 shows the condition number of the global sparse matrix, and the maximum condition number of all local collocation matrices when using PS of order $k = 4$ and $m = 6$, $n = 30$ on $F_1$ for two algorithms. This case was chosen mainly because this is a general case when reasonable accuracy and efficiency are achieved. Clearly, the small collocation matrices have higher condition numbers when the number of interpolation points and evaluation points increases, even becoming ill-conditioned in many cases. However, the global sparse matrices always have excellent condition numbers regardless of which algorithm and what parameters are used. When dealing with ill-conditioned small local systems, we would recommend a singular value decomposition, TSVD, or pre-conditioning techniques [20].

Furthermore, when comparing the CPU time, Algorithm 1 demonstrates greater efficiency compared to Algorithm 2. By examining the CPU time of the proposed methods, the most time-consuming part of the algorithms is identified as the construction of the local matrices while the final sparse system solving time is negligible. The algorithm *knnsearch* in MATLAB for nearest-neighbor classification is not the most efficient such algorithm. The highly efficient kd-tree algorithm [35] can be used if desired computational efficiency is needed.

The algorithms introduced in this paper are very much similar to the global RBF interpolation methods using CS-RBF. The Wendland's CS-RBFs were first introduced in [36], which is piecewise polynomial

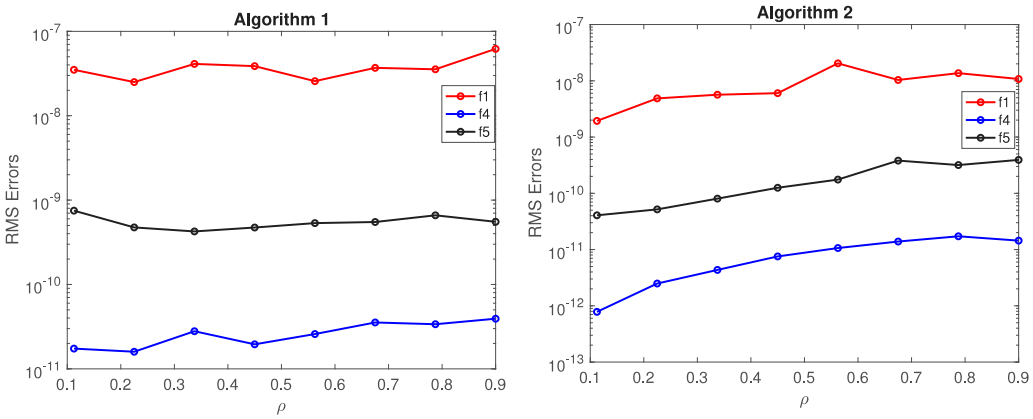



**Fig. 7.** Sensitivity respect to points distribution: RMS errors versus $\rho$ with $N = 100^2, N_t = 90^2$, and $n = 30$ using PS of order $k = 4$ and polynomial basis of order $m = 6$ in Example 4.1.

**Table 2**
Condition numbers and CPU time for Algorithm 1 and Algorithm 2 using PS of order $k = 4$ and $m = 6$, $n = 30$ on $F_1$ in Example 4.1.

| | Algorithm 1 | | Algorithm 2 | |
|---|---|---|---|---|
| $(N, N_t)$ | $(50^2, 2000)$ | $(150^2, 20000)$ | $(50^2, 2000)$ | $(150^2, 20000)$ |
| Condition No. of sparse matrix | 2.89E+03 | 4.81E+04 | 3.78E+03 | 8.13E+06 |
| Maximum condition No. of local matrices | 1.89E+14 | 2.8405E+20 | 3.96E+18 | 1.04E+22 |
| CPU time (s) | 1.71 | 75.60 | 3.09 | 166.37 |

**Table 3**
Comparison of CS-RBF method and the proposed methods with $k = 4, m = 3, n = 30$ for function $F_1$ in Example 4.1.

| $N$ | $N_t$ | CS-RBF | | Proposed Algorithm1 | | Proposed Algorithm2 | |
|---|---|---|---|---|---|---|---|
| | | $\epsilon_{rms}$ | CPU time | $\epsilon_{rms}$ | CPU time | $\epsilon_{rms}$ | CPU time |
| $100^2$ | 9000 | 7.66E−07 | 186.0 | 6.00E−08 | 15.10 | 4.02E−08 | 30.27 |
| $150^2$ | 20000 | 2.01E−07 | 1454.8 | 8.34E−09 | 71.2 | 5.26E−09 | 136.57 |

**Table 4**
Comparison of $\epsilon_{rms}$ and $\epsilon_{max}$ using various RBF with $N = 100^2$, $N_t = 9000$ and $n = 30$ for $F_6$ Example 4.1.

| RBF | $\epsilon_{rms}$ | $\epsilon_{max}$ | $c$ |
|---|---|---|---|
| Gaussian | 2.15E−13 | 4.06E−12 | 4.994 |
| MQ | 1.17E−12 | 2.97E−11 | 3.208 |
| IMQ | 3.29E−13 | 9.93E−12 | 2.629 |
| Matern (order 2) | 1.71E−12 | 3.43E−11 | 1.909 |
| PS4 | 3.22E−14 | 6.26E−13 | – |

**Table 5**
RMS errors and CPU time of $F_7$ interpolated with $N = 22,500, n = 30$ and various $N_t$ $(N < N_t)$ in Example 4.2.

| $N_t$ | No. of subsets | $\epsilon_{rms}$ | CPU time |
|---|---|---|---|
| 30,000 | 2 | 2.70E−05 | 126.32 |
| 40,000 | 2 | 2.19E−05 | 151.40 |
| 50,000 | 3 | 2.14E−05 | 196.44 |
| 60,000 | 3 | 2.06E−05 | 220.11 |

basis such as $\phi(r) = (1-r)_+^4(4r+1)$. Interpolation with CS-RBFs globally also resulted in a global sparse system [37]. Thus, we examine our algorithms against CS-RBFs interpolation. It is worth mentioning that the CS-RBF interpolation is still a global method since the interpolation matrix is created using CS-RBFs with all interpolation points as the centers. Table 3 shows performance in terms of accuracy and efficiency. In this example, PS of order 4 is used with a polynomial basis of order 3. From previous analysis, we know that we can improve the accuracy using a higher-order of polynomial basis or more neighboring points. With slightly higher accuracy, we still able to be much more efficient in terms of computational time.

Table 4 shows that the interpolations achieved with PS RBF have better accuracy than the other RBFs, Algorithm 1 is utilized to interpolate the test function $F_6 = \frac{1}{9}\left[64 - 81\left(\left(x-\frac{1}{2}\right)^2 + \left(y-\frac{1}{2}\right)^2\right)\right] - \frac{1}{2}$. The Leave-One-Out-Cross-Validation method (LOOCV) [25] is employed to determine the best shape parameter associated with RBFs with shape parameters to ensure that we choose the "optimal" value to the best of our ability. In LOOCV, we aim to fit data several times using a different training set (all but one data point) and testing set (one of the data points) each time, then calculating the test root mean squared error to be the average of all of the test.

**Example 4.2.** In this example, we further investigate the performance of the proposed Algorithm 1 using the function:

$$F_7(x, y) = \sqrt{x^2 + y^2} + 0.2, \tag{20}$$

which has a singularity and is more challenging to interpolate.

Fig. 8 shows the profile of $F_7$ in $[-1, 1] \times [-1, 1]$ on the left, and a profile of absolute error for test function $F_7$ on the right, where $k = 4$, $m = 6$, $N = 150^2, N_t = 20{,}000$ and $n = 30$. The absolute error is reasonably accurate, and it can be improved by employing more points in the local domains. The proposed method can certainly be successfully used for the interpolation of challenging functions without implementing any special treatments such as grid refinements.

We noticed that the higher the number of evaluation points, $N_t$, the greater the accuracy of the algorithms, but $N_t \leq N$. However, when there $N_t > N$, we can extract sub-samples of dimension $\bar{N}_t \ll N$ which can be interpolated group by group using the proposed method. These subsets need to be extracted as uniformly as possible from the domain to avoid any discontinuity between interpolated values (see Fig. 9). We implement this for the function $F_7$ with $N = 22{,}500$, and various $N_t$ that are higher than the given $N$. As shown in Table 5, the proposed method can maintain a similar level of accuracy for all the cases despite the number of evaluation points. An increase in the CPU time is expected as the number of sub-samples gets higher. However, this algorithm is suitable for parallel implementation as these groups are chosen independently, thus computer running time can be reduced significantly.

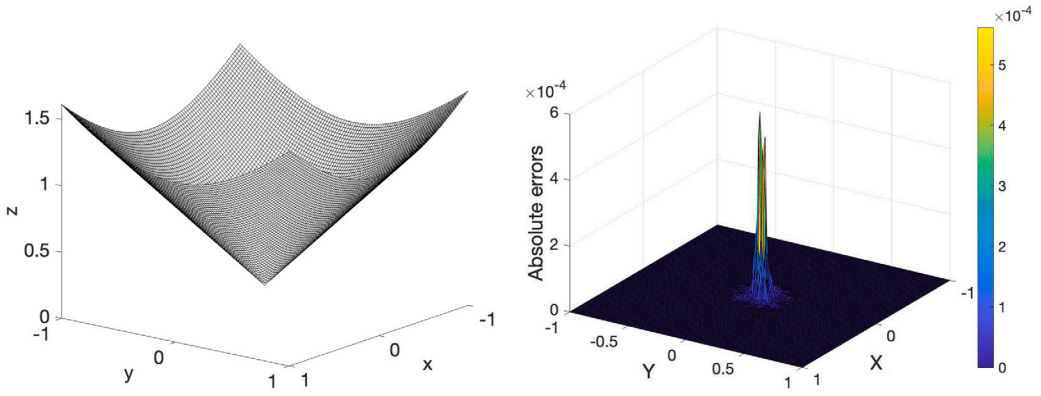



**Fig. 8.** Profile of test function $F_7$ on the left and absolute errors of interpolation with $N = 150^2, N_t = 20,000, n = 30, k = 4, m = 6$ on the right in Example 4.2.

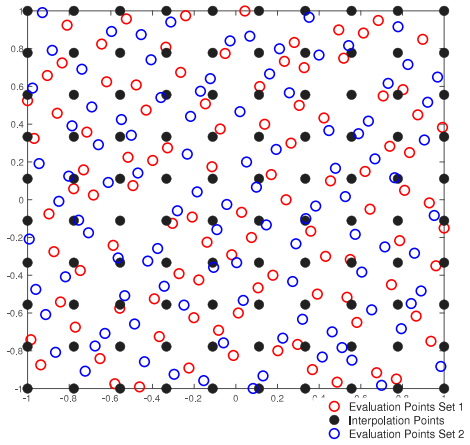



**Fig. 9.** Set of evaluation points and consistently distributed two sets of interpolation points.

**Example 4.3.** Interpolation is one of the common methods used to enhance image quality. In this example, 256 × 256 pixels 2D gray-scale Lena image shown on the left of Fig. 10 is enhanced. The enhanced 512 × 512 pixels image is shown on the right. Note that the enhanced image is obtained by using Algorithm 1 with PS of order $k = 1$, a polynomial of order $m = 0$, and $n = 3$. As the set of evaluation points is higher than the interpolation points in this problem, interpolation is performed group-wise. By maintaining a low order of polynomial basis and restricting the size of the local domains, we achieved visible improvements to the image while keeping the computational time low.

**Example 4.4.** The purpose of this example is to test Algorithm 1 on three-dimensional space. We use a computational domain of the Stanford Bunny, which is more challenging and irregular. The boundary data points for this domain are available at the website of the Stanford Computer Graphics Laboratory [38]. Algorithm 1 was tested on function $H$ in (21), which is a trivariate test function used in [13]. The function plotted on the bunny surface can be found in Fig. 11 (right):

$$H(x, y, z) = \frac{1}{3}\exp\left[-\frac{81}{16}\left((x-0.5)^2 + (y-0.5)^2 + (z-0.5)^2\right)\right]. \quad (21)$$

In our numerical simulations, $N = 4234$ of interpolation points which consist of 2345 interior points and 1889 boundary points are used to interpolate $N_t = 1143$ of test points. The computational domain with interpolation points lying on the boundary surface is shown in Fig. 11 (left). Since the original bunny data scale is too small for this application, the coordinates of bunny points are multiplied by 10. Algorithm 1 is employed with PS of order $k = 4$, the polynomial basis of order $m = 3$, and $n = 55$. Fig. 12 (left) shows the profile of the absolute errors on the surface of the Bunny and Fig. 12 (right) further confirms that the absolute errors in the interior of the bunny remain as low as the order of 5. The largest absolute error in all evaluation points is within an order of $10^{-5}$, with the majority of evaluation points having absolute errors less than $10^{-6}$ (which can be found in the tallest bar in the histogram). It verifies the method's high accuracy despite the complexity of the domain.

On the left of Fig. 13, it shows the maximum absolute errors, $\epsilon_{max}$, versus the number of points in local domains, $n$. Note that the range of the number of points in local domains is large, spanning from 25 to 1000. Regardless of the size of $n$, the algorithm consistently improves in accuracy. Thus, the greater the number of points in local domains, the higher the accuracy of the algorithm. However, we recommend choosing $n$ to be less than 200, as larger values result in increased computational time (as shown on the right of Fig. 13).

Fig. 14 shows the maximum absolute errors using Algorithm 1 when 4234 interpolation points and 1143 evaluation points are used in Example 4.4. The left of Fig. 14 plots $\epsilon_{max}$ versus the polynomial order $m$ when the number of points in local domains is chosen as $n = 85$ and $n = 120$. It can be seen from the figure that the order of the polynomials basis, $m$ should not be too large. The algorithm easily achieves an accuracy of order 5 when $m$ is as small as 2.

On the right of Fig. 14, the plot displays $\epsilon_{max}$ versus the order of PS, $k$ when the number of points in local domains is chosen as $n = 25, 45$, and $n = 85$. Please note the following: (1) as we discussed before, increasing the number of local points improves the accuracy, hence we did not present results for $n = 120$; (2) $m = 3$, $n = 45$ is sufficient to achieve reasonable accuracy, so the effect of the order of PS is not evident; (3) when $n = 25$, $k$ has to be small. In this case, the best accuracy is obtained only when $k = 1$.

Thus, we recommend using enough number of points in local domains to achieve high accuracy while keeping the order of polynomial basis small. In addition, the order of PS does not need to be high as it does not significantly affect the accuracy when a sufficient number of local points is used.

MATLAB code for this example is shared on MATLAB Central File Exchange [39].

## 5. Conclusion

In this paper, we improved two implicit localized RBF interpolation methods using polyharmonic splines and a polynomial basis for scattered data interpolation that was proposed in [21]. The method uses evaluation points or a combination of evaluation and interpolation points as the search domain to create local domains of influences for each interpolation point. The resulting linear system is a sparse system with the evaluation points' function values as the unknowns.

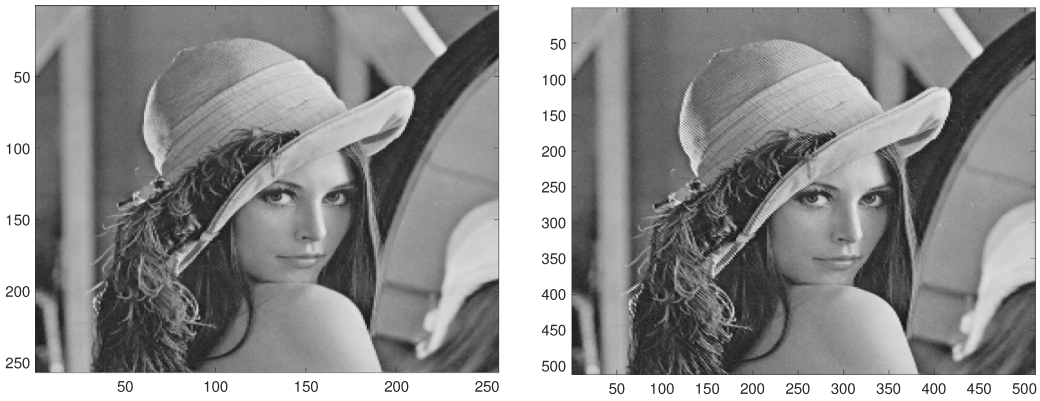



**Fig. 10.** Profile of the original Lena image on the left and the enhanced Lena image on the right inExample 4.3.

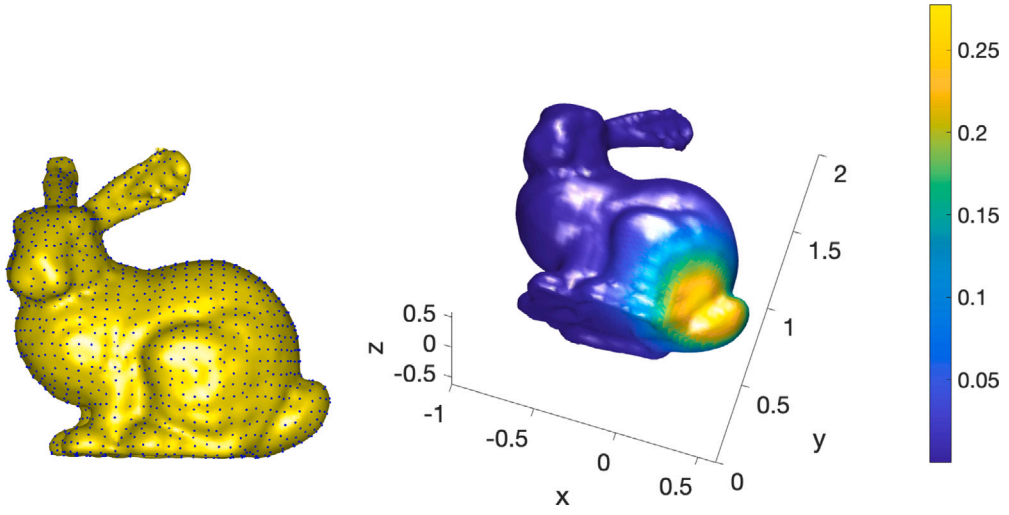



**Fig. 11.** Profile of the computational domain (Stanford Bunny) with the set of interpolation points on the boundary (left) and the profile of $H$ function on the surface of the bunny (right) in Example 4.4.

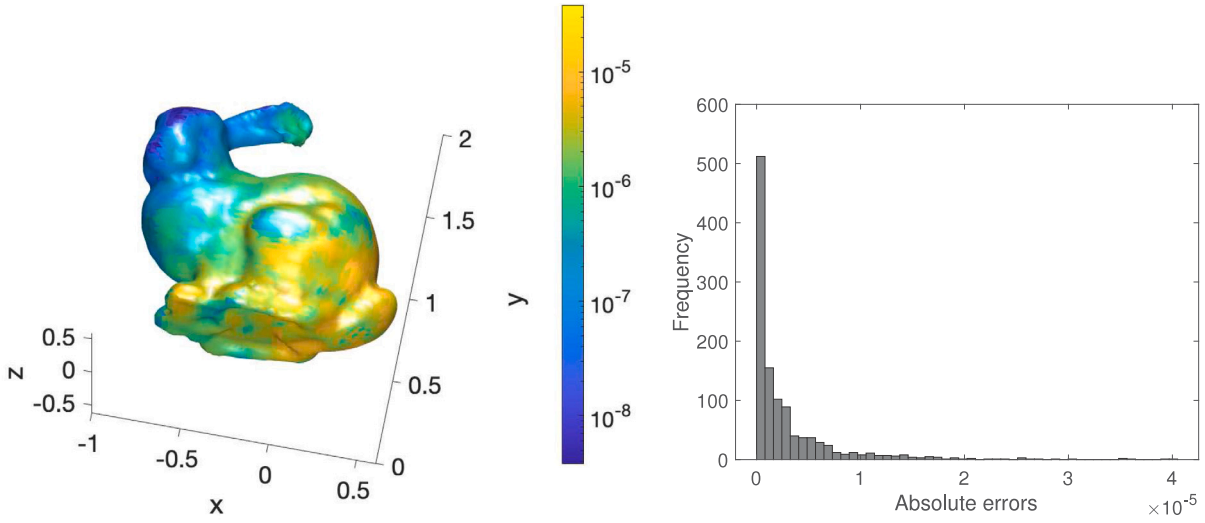



**Fig. 12.** The profile of the absolute errors on the surface of the bunny (left) and frequency or count of data points in the interior of the bunny whose absolute error are within each range (right) in Example 4.4.

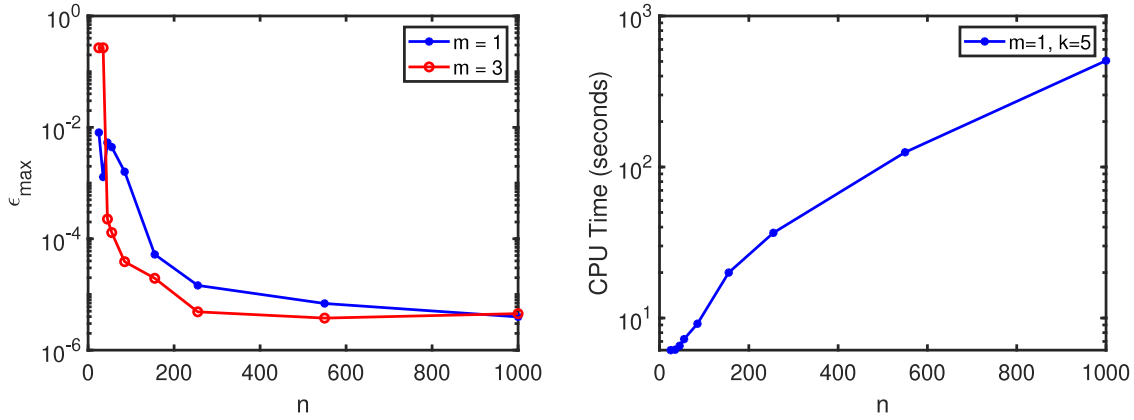



**Fig. 13.** The maximum absolute error, $\epsilon_{max}$ versus parameter $n$ on the left, and corresponding CPU time on the right in Example 4.4, where $N = 4234$, $N_t = 1143$.

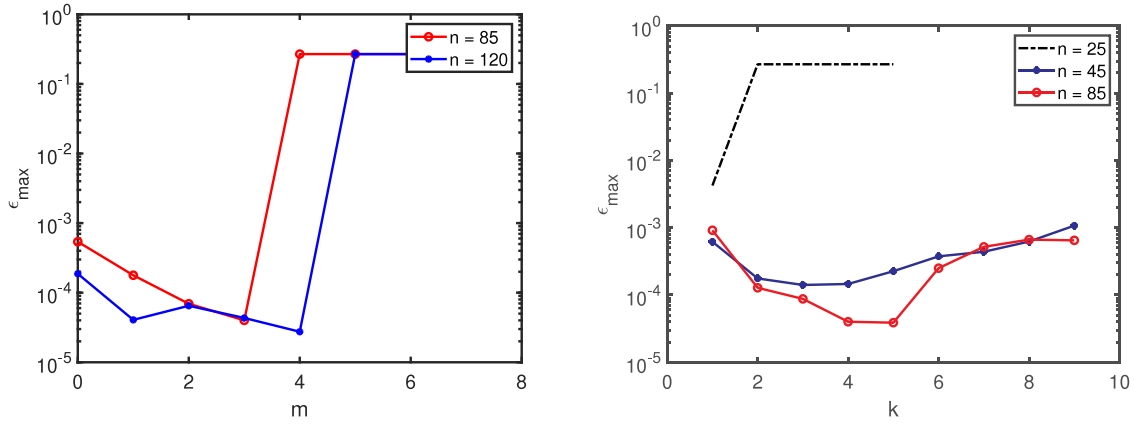



**Fig. 14.** The maximum absolute error, $\epsilon_{max}$ versus parameter $m$ on the left, parameter $k$ on the right in Example 4.4, where $N = 4234$, $N_t = 1143$.

The original paper claims the method "does not produce reasonable results when using the polyharmonic splines". However, we discovered in this paper, the claims are not true.

The interpolation with polyharmonic splines and low order polynomials can be solved with great accuracy and efficiency. The higher the order of polyharmonic splines or the number of points in local domains, the better the accuracy becomes, under the assumption that the points in local domains are sufficient. Detailed conclusions are drawn below: (1) RBFs such as polyharmonic splines and polynomials contain no shape parameter and produce higher accuracy compared to other RBFs; (2) Algorithm 1 is based on the construction of local influence domains entirely from the evaluation points and Algorithm 2 takes both interpolation and evaluation points into consideration. Both algorithms were found to be very attractive, easy to use in 2D and 3D problems, and mostly able to overcome the downsides of global RBF interpolation using polyharmonic splines and polynomial basis; (3) The computational complexities of Algorithm 1 and Algorithm 2 are very close. Additionally, Algorithm 2 is more accurate than Algorithm 1 but we mainly focused on Algorithm 1 due to its minor improvement of efficiency.

In summary, our proposed interpolation algorithms can deal with high-dimensional data interpolation on complicated domains, without the hassle of searching for shape parameters and without loss of accuracy and can be parallelized easily. This is a great improvement in terms of computational efficiency. Future work concerns deeper theoretical analysis of the convergence analysis and error estimates. In addition, we aim to further reduce the computational complexity of Algorithm 2 and apply it to classification problems.

## Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

## Data availability

I have shared my code on MATLAB File Exchange online.

## Acknowledgments

The second author and the fourth author, Guangming Yao, and Gantumur Tsogtgerel, acknowledge the support of the Simons CRM Scholar-in-Residence Program. The third author, Jing Niu, acknowledges the support of the National Natural Science Foundation of China via grant 12101164. The fourth author, Gantumur Tsogtgerel, was also supported by an NSERC Discovery Grant, and by the fellowship grant P2021-4201 of the National University of Mongolia.

## References

[1] Steffensen Johan Frederik. Interpolation. Courier Corporation; 2006.
[2] Davis Philip J. Interpolation and approximation. Courier Corporation; 1975.
[3] Lunardi Alessandra. Interpolation theory, Vol. 9. Springer; 2009.
[4] Szabados József, Vértesi Péter. Interpolation of functions. World Scientific; 1990.
[5] Skala Vaclav. RBF interpolation with CSRBF of large data sets. Procedia Comput Sci 2017;108:2433–7.
[6] Romani Lucia, Rossini Milvia, Schenone Daniela. Edge detection methods based on RBF interpolation. J Comput Appl Math 2019;349:532–47.

[7] Chen Chuin-Shan, Noorizadegan Amir, Young DL, Chen CS. On the selection of a better radial basis function and its shape parameter in interpolation problems. Appl Math Comput 2023;442:127713.
[8] Franke Richard, Nielson Gregory M. Scattered data interpolation and applications: A tutorial and survey. In: Hagen Hans, Roller Dieter, editors. Geometric modeling. Berlin, Heidelberg: Springer Berlin Heidelberg; 1991, p. 131–60.
[9] Hardy Rolland L. Multiquadric equations of topography and other irregular surfaces. J Geophys Res 1971;76(8):1905–15.
[10] Lazzaro Damiana, Montefusco Laura B. Radial basis functions for the multivariate interpolation of large scattered data sets. J Comput Appl Math 2002;140(1):521–36, Int. Congress on Computational and Applied Mathematics 2000.
[11] Renka Robert J. Multivariate interpolation of large sets of scattered data. ACM Trans Math Software 1988;14(2):139–48.
[12] Wendland Holger. Scattered data approximation. Cambridge monographs on applied and computational mathematics, Cambridge University Press; 2004.
[13] Bozzini Mira, Rossini Milvia. Testing methods for 3D scattered data interpolation. Multivar Approx Interpolat Appl 2002;20.
[14] Beatson Richard K, Light WA, Billings S. Fast solution of the radial basis function interpolation equations: Domain decomposition methods. SIAM J Sci Comput 2001;22(5):1717–40.
[15] Ling Leevan, Kansa Edward J. Preconditioning for radial basis functions with domain decomposition methods. Math Comput Modelling 2004;40(13):1413–27.
[16] Fasshauer Gregory E, Zhang Jack G, Fasshauer GE. Preconditioning of radial basis function interpolation systems via accelerated iterated approximate moving least squares approximation. 2009.
[17] Fornberg Bengt, Larsson Elisabeth, Flyer Natasha. Stable computations with Gaussian radial basis functions. SIAM J Sci Comput 2011;33:869–92.
[18] Floater Michael S, Iske Armin. Multistep scattered data interpolation using compactly supported radial basis functions. J Comput Appl Math 1996;73(1):65–78.
[19] Flyer Natasha, Fornberg Bengt, Bayona Victor, Barnett Gregory A. On the role of polynomials in RBF-FD approximations: I. Interpolation and accuracy. J Comput Phys 2016;321:21–38.
[20] Hansen Per Christian. MATLAB central file exchange. Regularization tools: A MATLAB package for analysis and solution of discrete ill-posed problems. Version 4.1.. 2023.
[21] Yao Guangming, Duo Jia, Chen CS, Shen LH. Implicit local radial basis function interpolations based on function values. Appl Math Comput 2015;265:91–102.
[22] Micchelli Charles A. Interpolation of scattered data: distance matrices and conditionally positive definite functions. Constr Approx 1986;2:11–22.
[23] Buhmann Martin D. Radial basis functions: theory and implementations. Cambridge Monogr Appl Comput Math 2003;12:147–65.
[24] Rippa Shmuel. An algorithm for selecting a good value for the parameter c in radial basis function interpolation. Adv Comput Math 1999;11(2–3):193–210.
[25] Fasshauer Gregory E, Zhang Jack G. On choosing "optimal" shape parameters for RBF approximation. Numer Algorithms 2007;45(1–4):345–68.
[26] Kansa EJ, Carlson RE. Improved accuracy of multiquadric interpolation using variable shape parameters. Comput Math Appl 1992;24(12):99–120.
[27] Fornberg B, Wright G. Stable computation of multiquadric interpolants for all values of the shape parameter. Comput Math Appl 2004;48(5–6):853–67.
[28] Chen CS, Karageorghis Andreas, Zheng Hui. Improved RBF collocation methods for fourth order boundary value problems. Commun Comput Phys 2020;27:1530–49.
[29] Zheng Hui, Lu Xujie, Jiang Pengfei, Yang Yabin. Numerical simulation of 3D double-nozzles printing by considering a stabilized localized radial basis function collocation method. Addit Manuf 2022;58:103040.
[30] Fedoseyev AI, Friedman MJ, Kansa EJ1883578. Improved multiquadric method for elliptic partial differential equations via PDE collocation on the boundary. Comput Math Appl 2002;43(3–5):439–55.
[31] Liu Chein-Shan, Liu Dongjie. Optimal shape parameter in the MQ-RBF by minimizing an energy gap functional. Appl Math Lett 2018;86:157–65.
[32] Zheng Hui, Yao Guangming, Kuo Lei-Hsin, Li Xinxiang. On the selection of a good shape parameter of the localized method of approximated particular solutions. Adv Appl Math Mech 2018;10(4):896–911.
[33] Iske Armin. On the approximation order and numerical stability of local Lagrange interpolation by polyharmonic splines. In: Modern developments in multivariate approximation. Springer; 2003, p. 153–65.
[34] Franke Richard. Scattered data interpolation: tests of some methods. Math Comput 1982;38(157):181–200.
[35] Bentley Jon Louis. Multidimensional binary search trees used for associative searching. Commun ACM 1975;18(9):509–17.
[36] Wendland Holger. Piecewise polynomial, positive definite and compactly supported radial functions of minimal degree. Adv Comput Math 1995;4(1):389–96.
[37] Fasshauer Gregory E, McCourt Michael J. Kernel-based approximation methods using matlab, Vol. 19. World Scientific Publishing Company; 2015.
[38] Stanford Computer Graphics Laboratory. The Stanford 3D scanning repository.
[39] Yao Guangming. Polyharmonic splines interpolation on scattered data. 2023.