Research Paper
Computational Methods for Differential Equations
http://cmde.tabrizu.ac.ir
Vol. 11, No. 1, 2023, pp. 108-129
DOI:10.22034/CMDE.2022.46208.1939

Choosing the best value of shape parameter in radial basis functions by Leave-P-Out Cross Validation

Mohammad Reza Yaghouti* **and Farnaz Farshadmoghadam**
Faculty of Mathematical Sciences, University of Guilan, Rasht, Iran.

Abstract

The radial basis functions (RBFs) meshless method has high accuracy for the interpolation of scattered data in high dimensions. Most of the RBFs depend on a parameter, called shape parameter which plays a significant role to specify the accuracy of the RBF method. In this paper, we present three algorithms to choose the optimal value of the shape parameter. These are based on Rippa's theory to remove data points from the data set and results obtained by Fasshauer and Zhang for the iterative approximate moving least square (AMLS) method. Numerical experiments confirm stable solutions with high accuracy compared to other methods.

Keywords. Radial basis functions, Shape parameter, Leave-One-Out cross validation, Leave-Two-Out cross validation, Approximate moving least squares.

2010 Mathematics Subject Classification. 65D05, 65D15, 65M70.

1. Introduction

An important class of numerical methods for the solution of partial differential equations and scattered data interpolation is the meshless methods. Some of them are the point interpolation method [25], the local point interpolation method [20], the smooth particle hydrodynamics method [18], the diffuse element method [17], the element free Galerkin method [24], the meshless local Petrov-Galerkin method [1], and the radial basis functions method [3, 5, 15].

Among those, the radial basis functions (RBFs) method is the most reliable one in many fields. In this method, we do not mesh the geometric space of the problem, but instead, use some scattered points in the desired domain. Since the RBF method uses pairwise distances between points, so it is efficient for problems with high dimensions and irregular domains. Radial basis functions are classified into two main classes [6, 21].
Class 1. Infinitely smooth RBFs.
These basis functions are infinitely differentiable and depend on a parameter, called shape parameter ε (such as Multiquadric(MQ), Inverse Multiquadric (IMQ) and Gaussian (GA)).
Class 2. Finitely smooth RBFs.
These basis functions are not infinitely differentiable and are lacking the shape parameter. Also, these functions have less accurate than ones in Class 1. (such as Thin Plate Splines (TPS)).
The shape parameter is determined by the user and can also be used to increase the accuracy of the method, which is often considered as a drawback since the proper selection of ε is still an open question. To solve this issue, several attempts are made such as Hardy [19] employed $\varepsilon = 0.815d$ for MQ basis in $\mathbb{R}^2$ where $d = \frac{1}{N}\sum_{i=1}^{N} d_i$ and d_i is the distance between the i^{th} data point and its closest neighborhood, also N is the number of data. Franke [16] defined $\varepsilon = \frac{1.25D}{\sqrt{N}}$ in which D is the diameter of the smallest circle containing all data points. Carlson and Foley [4] recommended that the optimal value of ε is independent of the number and position of data points. Furthermore,

Received: 23 May 2021 ; Accepted: 22 July 2022.
*Corresponding author. Email: yaghouti@guilan.ac.ir .

they suggested using the same value of ε for MQ and IMQ RBF interpolation. Rippa [29] observed that the optimal value depends on the number and position of grid points, the approximated function, the precision of the calculation, and the kind of RBF interpolation. Moreover, Rippa introduced an algorithm made on the Leave-One-Out Cross Validation (LOOCV) method to pick the optimal value of ε for any RBF and any dimension by minimizing a cost function. Scheuerer [30] improved Rippa's algorithm for a cost function by using some new weighted error and showed to be equivalent to the so-called maximum likelihood estimator method that is used to compute covariance parameters of stochastic processes in spatial statistics.

Small values of the shape parameter are well known to make highly accurate results, but the coefficient matrix of interpolation becomes ill-conditioned. One alternative method to solve this problem is the approximate moving least squares method which is a quasi-interpolation approach. Lancaster and Salkauskas [22] provided the moving least squares (MLS) method for smoothing and interpolating data. In [27], Maz'ya suggested the approximate MLS (AMLS) method that avoids solving any linear systems. Fasshauer and Zhang later did more research on the AMLS method and used it instead of the RBF approximation [7–9, 11, 14]. The AMLS method is principally suitable for regular grid points because the formulation for irregular grid points is more difficult [23, 26]. In [14], Fasshauer and Zhang showed that the AMLS method performs well for both regular and irregular grid points as an RBF approximation.

Fasshauer and Zhang [14] stated an algorithm based on the residual iteration of an AMLS method that converges to the RBF approximation. Then they used this algorithm and Rippa's theory to propose two different LOOCV schemes in order to find an optimal shape parameter [13]. These methods do not require costly matrix calculations and perform well for the approximation of noisy data.

Azarboni et al. [2] declared Leave-Two-Out Cross Validation (LTOCV) method by removing two data from the data set and utilized the results of Fasshauer and Zhang. They indicated that the accuracy of the LTOCV method is better than the LOOCV method but the computational time is increased.

In this paper, we use and extend the results obtained from elimination methods and the AMLS method. We introduce an iterative LTOCV algorithm by the iterative AMLS method. Moreover, we offer a generalized formula for the cost function to remove P data from the data set, this method is called Leave-P-Out Cross Validation method, and apply this method for P = 3. Also, we gain an iterative Leave-Three-Out Cross Validation algorithm by the iterative AMLS method. Numerical results illustrate the accuracy, efficiency, and stability of our methods in comparison with other methods, but the computational time with the growth of deleted points increases.

The rest of the paper is structured as follows. In section 2, a review of the previous studies is mentioned. In section 3, some new approximation methods for selecting the optimal shape parameter are explained. Numerical examples are included in section 4.

2. Preliminaries

In this section, we will briefly explain some basic definitions and items required in this study.

2.1. RBF Approximation.

An RBF approximation is a linear combination of radial basis functions of the form

$$\mathbf{W}(x)=\sum_{i=1}^{N} c_i\varphi(\|x-x_i\|), \qquad x\in\mathbb{R}^d. \tag{2.1}$$

To interpolate the given values $f(x_i)=f_i, i=1,2,\ldots,N$, at scattered data set $\mathbf{x}=\{x_i; x_i\in\mathbb{R}^d, i=1,2,\ldots,N\}$ such that

$$\mathbf{W}(x_i)=f_i, \quad i=1,2,\ldots,N. \tag{2.2}$$

Scattered data set $\mathbf{x}$ is called the center nodes of the RBF approximation, $\|\cdot\|$ is usually the Euclidean norm and $\varphi(r)$, such that $r\geq 0$, is a radial basis function. Some popular choices of radial basis functions are summarized in Table 1. The unknown vector $\mathbf{c}=(c_1,c_2,\ldots,c_N)^T$ is specified by applying the interpolation conditions (2.2), which is corresponding to solve the linear system of equations

$$\mathbf{Ac}=\mathbf{f}, \tag{2.3}$$

where $\mathbf{A}_{ij} = \varphi(\|x_i - x_j\|)$ and $\mathbf{f} = (f_1, f_2, \ldots, f_N)^T$. In [28], Micchelli investigated the non-singularity of the matrix $\mathbf{A}$ for MQ-RBF. Also, Schoenberg [31] showed that the matrix $\mathbf{A}$ is positive definite and so nonsingular for GA, IMQ, IQ RBFs.

TABLE 1. Some radial basis functions.

Name of the function	RBF form
Gaussian (GA)	$\varphi(r) = e^{-\varepsilon^2 r^2}$
Multiquadric (MQ)	$\varphi(r) = \sqrt{1+\varepsilon^2 r^2}$
Inverse Multiquadric (IMQ)	$\varphi(r) = \frac{1}{\sqrt{1+\varepsilon^2 r^2}}$
Inverse Quadric (IQ)	$\varphi(r) = \frac{1}{1+\varepsilon^2 r^2}$

2.2. **Leave-One-Out Cross Validation Method.** In [29], Rippa suggested an algorithm made on the LOOCV method which determines an optimal value for the shape parameter by minimizing a cost function. The cost function is defined by getting the norm of error vector $\mathbf{e} = (e_1, e_2, \ldots, e_N)^T$ in which

$$e_k = f_k - \mathbf{W}^{(k)}(x_k), \qquad k = 1, 2, \ldots, N,$$

and $\mathbf{W}^{(k)}$ is the approximate function to a decreased data set obtained by deleting the point x_k from the data set as follows:

$$\mathbf{W}^{(k)}(x) = \sum_{i=1, i\neq k}^{N} c_i^{(k)} \varphi(\|x - x_i\|).$$

The unknown vector

$$\mathbf{c}^{(k)} = (c_1^{(k)}, \ldots, c_{k-1}^{(k)}, c_{k+1}^{(k)}, \ldots, c_N^{(k)})^T,$$

is resulted by employing the interpolation conditions

$$\mathbf{W}^{(k)}(x_i) = f_i, \quad i = 1, 2, \ldots, N, \quad i \neq k,$$

which is corresponding to solve the nonsingular linear system $\mathbf{A}^{(k)}\mathbf{c}^{(k)} = \mathbf{f}^{(k)}$, where

$$\mathbf{f}^{(k)} = (f_1^{(k)}, \ldots, f_{k-1}^{(k)}, f_{k+1}^{(k)}, \ldots, f_N^{(k)})^T,$$

and $\mathbf{A}^{(k)}$ is a submatrix of the matrix $\mathbf{A}$ by deleting the k^{th} row and column. This algorithm is costly and requires $O(N^4)$ computational operations. Rippa decreased its computational operations to $O(N^3)$. He attained a simple formula for the error term as follows:

$$e_k = \frac{c_k}{\mathbf{A}_{kk}^{-1}}, \tag{2.4}$$

where c_k is the k^{th} coefficient of the vector $\mathbf{c}$ of the RBF interpolation on full data set, and $\mathbf{A}_{kk}^{-1}$ is k^{th} diagonal element of the inverse matrix obtained in the interpolation. Rippa used Brent's method to derive the minimum of $\|\mathbf{e}\|$.

2.3. **Approximate Moving Least Squares Method.** The approximate moving least squares (AMLS) method is an approximation technique that avoids solving the system of linear equations [10, 26]. In [14], Fasshauer and Zhang proposed a sequence of AMLS method which starts with a quasi-interpolant of the form

$$\psi_f^{(0)}(x) = \sum_{i=1}^{N} f(x_i)\varphi(\|x - x_i\|),$$

then, continues with the iteration on residuals as follows:

$$\psi_f^{(n)}(x) = \psi_f^{(n-1)}(x) + \sum_{i=1}^{N}\left[f(x_i) - \psi_f^{(n-1)}(x_j)\right]\varphi\left(\|x - x_i\|\right),$$

where $f(x_i) = f_i,\ i = 1,\dots,N$ are the given data values and the function φ is strictly positive definite and satisfies continuous moment conditions. The Laguerre-Gaussians in the following form have the conditions listed

$$\varphi(r) = \frac{1}{\sqrt{\pi^d}}e^{-r^2}L_n^{d/2}\left(r^2\right), \quad r = \|x\|\,; x \in \mathbb{R}^d,$$

in which $L_n^{d/2}(\cdot)$ are Laguerre polynomials of degree n and order $d/2$.

Theorem 2.1. *Let φ be a strictly positive definite function and generates an interpolation matrix $\mathbf{A}$ and*

$$\max_{i=1,2,\dots,N}\left\{\sum_{j=1}^{N}|\mathbf{B}_{ij}|\right\} < 2,$$

so that the matrix $\mathbf{B}$ is a scaled version of $\mathbf{A}$, i.e.,

$$\mathbf{B}_{ij} = \varepsilon^d\varphi\left(\frac{\varepsilon}{h}\|x_i - x_j\|\right).$$

Here $h = 1/\left(N^{1/d} - 1\right)$, then $\psi_f^{(n)}$ converges to the RBF approximation as $n \to \infty$.

Proof. Complete explanations are given by [13, 14]. □

Using the above content, the following basis function can be used

$$\varphi(r) = \frac{\varepsilon^d}{\sqrt{\pi^d}}e^{-\varepsilon^2 r^2/h^2}. \tag{2.5}$$

The formulation of this method is as follows:

$$\mathbf{A}\sum_{i=0}^{n}\left(\mathbf{I} - \mathbf{A}\right)^i\mathbf{f} = \psi_f^{(n)}(\cdot), \tag{2.6}$$

in which the matrix $\mathbf{A}$ is symmetric and positive definite ($\mathbf{A}_{ij} = \varphi\left(\|x_i - x_j\|\right)$), $\mathbf{I}$ is the identity matrix and $\mathbf{f} = \left(f_1, f_2, \dots, f_N\right)^T$. Eq. (2.6) is a linear system with the coefficient matrix $\mathbf{A}$, the unknown vector $\sum_{i=0}^{n}\left(\mathbf{I} - \mathbf{A}\right)^i\mathbf{f}$ and right-hand side vector $\psi_f^{(n)}(\cdot)$, by multiplying both sides of it into

$$\left[\sum_{i=0}^{n}\left(\mathbf{I} - \mathbf{A}\right)^i\right]^{-1}\mathbf{A}^{-1},$$

the following result will be obtained

$$\left[\sum_{i=0}^{n}\left(\mathbf{I} - \mathbf{A}\right)^i\right]^{-1}\left[\sum_{i=0}^{n}\left(\mathbf{I} - \mathbf{A}\right)^i\mathbf{f}\right] = \mathbf{f}. \tag{2.7}$$

In accordance with the Neumann series $\sum_{i=0}^{\infty}\left(\mathbf{I} - \mathbf{A}\right)^i = \mathbf{A}^{-1}$, Eq. (2.7) is an estimation of the linear system (2.3) of the RBF interpolation.

In [13], Fasshauer and Zhang used Eq. (2.7) to compute the error term (2.4) as follows:

$$e_k = \frac{\left[\sum_{i=0}^{n}\left(\mathbf{I} - \mathbf{A}\right)^i\mathbf{f}\right]_k}{\left[\sum_{i=0}^{n}\left(\mathbf{I} - \mathbf{A}\right)^i\right]_{kk}}, \tag{2.8}$$

where there is no the inverse matrix. Moreover, they defined some suitable recursion relations that decreased the computational costs in (2.8). They assumed that $\mathbf{V}^{(0)} = \mathbf{f}$, and utilized the recursive formula

$$\mathbf{V}^{(n)} = \mathbf{f} + (\mathbf{I} - \mathbf{A})\, \mathbf{V}^{(n-1)},$$

for gaining the unknown vector $\sum_{i=0}^{n} (\mathbf{I} - \mathbf{A})^i \mathbf{f}$ such that $(\mathbf{V}^{(n)})_k$ is the k^{th} component of the numerator in (2.8). Also, using the eigen-decomposition of the matrix, costly calculations of matrix powers in the denominator of (2.8) decreased. They put

$$\mathbf{I} - \mathbf{A} = \mathbf{X}\boldsymbol{\Lambda}\mathbf{X}^{-1},$$

where $\boldsymbol{\Lambda}$ is a diagonal matrix of the eigenvalues and the columns of the matrix $\mathbf{X}$ are the eigenvectors of $\mathbf{I} - \mathbf{A}$. Assuming $\mathbf{M}^{(0)} = \mathbf{I}$, the denominator of (2.8) can be calculated by the following formula

$$\mathbf{M}^{(n)} = \boldsymbol{\Lambda}\mathbf{M}^{(n-1)} + \mathbf{I}. \tag{2.9}$$

Consequently, the following relation is satisfied for each n

$$\left[\sum_{i=0}^{n} (\mathbf{I} - \mathbf{A})^i\right] = \mathbf{X}\mathbf{M}^{(n)}\mathbf{X}^{-1}. \tag{2.10}$$

In Eq. (2.9), the matrix $\mathbf{M}^{(n)}$ is diagonal, so the diagonal elements in (2.10) are computed without the matrix-matrix multiplication. Fasshauer and Zhang used the Matlab function `fminbnd` to get the minimum of $\|\mathbf{e}\|$ in order to find the optimal shape parameter, also the number of iterations is specified.

2.4. **Leave-Two-Out Cross Validation Method.** In [2], Azarboni et al. used Rippa's results and algorithms introduced by Fasshauer and Zhang, and stated an algorithm made on the LTOCV method by minimizing the cost function which is done by getting the ℓ_2 norm of the error matrix $\mathbf{E}$ with entries $e_{kl} = \|\mathbf{E}_{kl}\|_2$, such that

$$\mathbf{E}_{kl} = \begin{bmatrix} f_k - \mathbf{W}^{(k,l)}(x_k) \\ f_l - \mathbf{W}^{(k,l)}(x_l) \end{bmatrix}, \quad k = 1, \ldots, N-1, \quad l = k+1, \ldots, N, \tag{2.11}$$

and $\mathbf{W}^{(k,l)}$ is the approximate function to a decreased data set obtained by deleting the points x_k and x_l as follows:

$$\mathbf{W}^{(k,l)}(x) = \sum_{i=1, i\neq k,l}^{N} c_i^{(k,l)} \varphi(\|x - x_i\|).$$

The unknown vector

$$\mathbf{c}^{(k,l)} = \left(c_1^{(k,l)}, \ldots, c_{k-1}^{(k,l)}, c_{k+1}^{(k,l)}, \ldots, c_{l-1}^{(k,l)}, c_{l+1}^{(k,l)}, \ldots, c_N^{(k,l)}\right)^T,$$

is determined by applying the interpolation conditions

$$\mathbf{W}^{(k,l)}(x_i) = f_i, \quad i = 1, 2, \ldots, N, \quad i \neq k, l,$$

which is corresponding to solve the nonsingular linear system of equations

$$\mathbf{A}^{(k,l)}\mathbf{c}^{(k,l)} = \mathbf{f}^{(k,l)},$$

where

$$\mathbf{f}^{(k,l)} = \left(f_1^{(k,l)}, \ldots, f_{k-1}^{(k,l)}, f_{k+1}^{(k,l)}, \ldots, f_{l-1}^{(k,l)}, f_{l+1}^{(k,l)}, \ldots, f_N^{(k,l)}\right)^T,$$

and $\mathbf{A}^{(k,l)}$ is a submatrix of the matrix $\mathbf{A}$ by deleting k^{th} and l^{th} rows and columns. They presented a formula for $\mathbf{E}_{kl}$ (2.11), minimized the cost function by using the Matlab function `fminbnd` and obtained an optimal value of ε.

3. Some New Approximation Methods

In this section, firstly, we propose an iterative LTOCV algorithm by the iterative AMLS method, then, we present Leave-P-Out Cross Validation method and explain it for P = 3, finally, express an iterative Leave-Three-Out Cross Validation algorithm by the iterative AMLS method.

3.1. **Iterative LTOCV (ItLTOCV) for iterative AMLS Method.** In [2], Azarboni et al. introduced a direct LTOCV method and then used the results of Fasshauer and Zhang. Since the AMLS method avoids solving the system of linear equations, one can consider a straight execution from it for the LTOCV method. In the following, we propose the iterative LTOCV (ItLTOCV) algorithm for the iterative AMLS method, in the form of Algorithm 1.

Algorithm 1

```
Fix $\varepsilon$
for $n = 1, 2, ...$ do
  for $k = 1, 2, ..., N-1$ do
    for $l = k+1, \ldots, N$ do
      Let
      $$\psi_f^{(0)(k,l)}(x) = \sum_{j=1}^{N-2} f_j^{(k,l)} \varphi\left(\left\|x - x_j^{(k,l)}\right\|\right)$$
      for $j = 1, 2, ..., N-2$ do
        Compute residuals at the datasites
        $$r_j^{(n)(k,l)} = f_j^{(k,l)} - \psi_f^{(n-1)(k,l)}\left(x_j^{(k,l)}\right)$$
      end for
      Compute the correction
      $$u(x) = \sum_{j=1}^{N-2} r_j^{(n)(k,l)} \varphi\left(\left\|x - x_j^{(k,l)}\right\|\right)$$
      Update
      $$\psi_f^{(n)(k,l)}(x) = \psi_f^{(n-1)(k,l)}(x) + u(x) \qquad (3.1)$$
      Compute the error estimate for the $k^{th}$ and $l^{th}$ data points
      $$\mathbf{E}_{kl}^{(n)(k,l)} = \begin{bmatrix} f(x_k) - \psi_f^{(n)(k,l)}(x_k) \\ f(x_l) - \psi_f^{(n)(k,l)}(x_l) \end{bmatrix}$$
      $e_{kl}^{(n)(k,l)}$ is the $\ell_2$ norm of $\mathbf{E}_{kl}^{(n)(k,l)}$.
    end for
    From the cost matrix $\mathbf{E}^{(n)} = (e_{kl}^{(n)(k,l)})_{k<l}$
      $$ceps(n) = norm(\mathbf{E}^{(n)}, 2)$$
    if $(ceps(n) - ceps(n-1)) < tol$ then
      Stop the iteration
    end if
  end for
end for
```

Algorithm 1 removes points x_k and x_l from data set such that $k = 1, \ldots, N-1$, and $l = k+1, \ldots, N$, then applies the iterative AMLS method to this set of new data points until the stopping criterion is satisfied. Since this method converges to the RBF approximation, so in Eq. (3.1) the RBF interpolation is derived by the removal of two points. The cost function "$ceps(n)$" (See Program 1) is determined by getting the ℓ_2 norm of the upper triangular matrix

$\mathbf{E}^{(n)}$, with zero diagonal elements and entries $e_{kl}^{(n)(k,l)} = \left\|\mathbf{E}_{kl}^{(n)(k,l)}\right\|_2$, such that

$$\mathbf{E}_{kl}^{(n)(k,l)} = \left[\begin{array}{c} f(x_k) - \psi_f^{(n)(k,l)}(x_k) \\ f(x_l) - \psi_f^{(n)(k,l)}(x_l) \end{array}\right].$$

To speed up the computational operation we use the eigen-decomposition for the interpolation matrix, and the residual value is calculated for all points as follows:

$$f(x_j) - \psi_f^{(n)(k,l)}(x_j), \quad j = 1, \ldots, N.$$

Then, the Matlab function `fminbnd` minimizes the cost function in the following form

```
ep=fminbnd(@(ep1)CostEpsItLTOCV(ep1,phi,T,f,m),minep,maxep)
```

here, phi is a radial basis function, the matrix $\mathtt{T} = (t_{ij})$ is in the form of $t_{ij} = \|x_i - x_j\|$ and $(\mathtt{minep}, \mathtt{maxep})$ is the desired interval for determining the optimal shape parameter.
Finally, an optimal value of ε is obtained and n as the number of iterations is achieved. All Programs in this study are listed in the appendix.

3.2. **Leave-P-Out Cross Validation Method.** In this subsection, we present a generalized formula to obtain the error term by removing P data from the data set, this method called Leave-P-Out Cross Validation (LPOCV) method. Let $\mathbf{x} = \{x_i; x_i \in \mathbb{R}^d, i = 1, 2, \ldots, N\}$ be the data set and $\mathbf{W}(x)$ be the RBF approximation centering on $\mathbf{x}$ to interpolate the given values $f(x_i) = f_i, i = 1, 2, \ldots, N$, like (2.1). Now, we define a new data set

$$\mathbf{x}^{(k_1,k_2,\ldots,k_P)} = (x_1, \ldots, x_{k_1-1}, x_{k_1+1}, \ldots, x_{k_2-1}, x_{k_2+1}, \ldots, x_{k_P-1}, x_{k_P+1}, \ldots, x_N),$$

as a vector of $\mathbf{x}$ such that the points $x_{k_1}, x_{k_2}, \ldots, x_{k_P}$, are removed, also $k_1 < k_2 < \cdots < k_P$, and $k_1 = 1, \ldots, N-(P-1)$, $\ldots$, $k_P = k_{P-1}+1, \ldots, N$, likewise, $\mathbf{f}^{(k_1,k_2,\ldots,k_P)}$ is the subvector of the data vector $\mathbf{f} = (f_1, f_2, \ldots, f_N)^T$, such that the elements $f_{k_1}, f_{k_2}, \ldots, f_{k_P}$ are removed. The interpolation function on the new data set is as follows:

$$\mathbf{W}^{(k_1,k_2,\ldots,k_P)}(x) = \sum_{\substack{i=1, \\ i \neq k_1,k_2,\ldots,k_P}}^{N} c_i^{(k_1,k_2,\ldots,k_P)} \varphi(\|x - x_i\|).$$

The unknown vector

$$\mathbf{c}^{(k_1,k_2,\ldots,k_P)} = (c_1, \ldots, c_{k_1-1}, c_{k_1+1}, \ldots, c_{k_2-1}, c_{k_2+1}, \ldots, c_{k_P-1}, c_{k_P+1}, \ldots, c_N),$$

is specified by enforcing the interpolation conditions

$$\mathbf{W}^{(k_1,k_2,\ldots,k_P)}(x_i) = f_i, \quad i = 1, 2, \ldots, N, \quad i \neq k_1, k_2, \ldots, k_P,$$

which is corresponding to solve the nonsingular linear system of equations

$$\mathbf{A}^{(k_1,k_2,\ldots,k_P)} \mathbf{c}^{(k_1,k_2,\ldots,k_P)} = \mathbf{f}^{(k_1,k_2,\ldots,k_P)},$$

where $\mathbf{A}^{(k_1,k_2,\ldots,k_P)}$ is a submatrix of the matrix $\mathbf{A}$ by deleting $k_1^{th}, k_2^{th}, \ldots, k_P^{th}$ rows and columns.

Definition 3.1. Assume that $\mathbf{y}, \mathbf{z} \in \mathbb{R}^N$ and $\mathbf{A}\mathbf{y} = \mathbf{z}$. If $y_{k_1} = y_{k_2} = \cdots = y_{k_P} = 0$, then

$$\mathbf{A}^{(k_1,k_2,\ldots,k_P)} \left(\mathbf{y}^{(k_1,k_2,\ldots,k_P)}\right)^T = \left(\mathbf{z}^{(k_1,k_2,\ldots,k_P)}\right)^T.$$

Lemma 3.2. *Suppose that* $x^{[k]}$ *be the solution of the system*

$$\mathbf{A} x^{[k]} = e^{[k]},$$

where $e^{[k]}$ *is the* k^{th} *column of the* $N \times N$ *identity matrix, then* $x_k^{[k]} \neq 0$.

Proof. The proof is straightforward in [29]. □

Theorem 3.3. *Let* $\mathbf{B}$ *be a* $P \times P$ *matrix with entries* $\mathbf{B}_{ij} = x_{k_i}^{[k_j]}$ *which are defined with the help of Lemma 3.2 and* $i, j = 1, \dots, P$, *such that* $k_1 < k_2 < \dots < k_P$, $k_1 = 1, \dots, N - (P-1), \dots, k_P = k_{P-1} + 1, \dots, N$, *and* $|\mathbf{B}| \neq 0$, ($|\mathbf{B}|$ *is the determinants of the matrix* $\mathbf{B}$). *Also, the cofficient vector* $\mathbf{c} = (c_1, c_2, \dots, c_N)^T$ *is obtained by solving the linear system of equations of the RBF interpolation on full data set, and the constants* y_{k_i}, *are defined by*

$$y_{k_i} = \frac{\begin{vmatrix} x_{k_1}^{[k_1]} & \dots & c_{k_1} & \dots & x_{k_1}^{[k_P]} \\ x_{k_2}^{[k_1]} & \dots & c_{k_2} & \dots & x_{k_2}^{[k_P]} \\ \vdots & \dots & \vdots & \dots & \vdots \\ x_{k_P}^{[k_1]} & \dots & c_{k_P} & \dots & x_{k_P}^{[k_P]} \end{vmatrix}}{|\mathbf{B}|}, \quad i = 1, \dots, P, \tag{3.2}$$

the numerator in (3.2) *is determined by replacing the* k_i^{th} *column of the matrix* $\mathbf{B}$ *with the subvector of the cofficient vector* $\mathbf{c}$. *Then, the error vector* $\mathbf{E}_{k_1 k_2 \dots k_P}$ *in the LPOCV method is given by*

$$\mathbf{E}_{k_1 k_2 \dots k_P} = \begin{bmatrix} f_{k_1} - \mathbf{W}^{(k_1, k_2, \dots, k_P)}(x_{k_1}) \\ f_{k_2} - \mathbf{W}^{(k_1, k_2, \dots, k_P)}(x_{k_2}) \\ \vdots \\ f_{k_P} - \mathbf{W}^{(k_1, k_2, \dots, k_P)}(x_{k_P}) \end{bmatrix} = \begin{bmatrix} y_{k_1} \\ y_{k_2} \\ \vdots \\ y_{k_P} \end{bmatrix}.$$

Proof. Let us consider $\mathbf{b}^{[k_1, k_2, \dots, k_P]} \in \mathbb{R}^N$ as follows:

$$\mathbf{b}^{[k_1, k_2, \dots, k_P]} = \mathbf{c} - y_{k_1} x^{[k_1]} - y_{k_2} x^{[k_2]} - \dots - y_{k_P} x^{[k_P]}, \tag{3.3}$$

then we multiply the matrix $\mathbf{A}$ on both sides of Eq. (3.3)

$$\mathbf{A}\mathbf{b}^{[k_1, k_2, \dots, k_P]} = \mathbf{A}\mathbf{c} - y_{k_1} \mathbf{A} x^{[k_1]} - y_{k_2} \mathbf{A} x^{[k_2]} - \dots - y_{k_P} \mathbf{A} x^{[k_P]}.$$

Now by applying Lemma 3.2 we have

$$\begin{aligned} \mathbf{A}\mathbf{b}^{[k_1, k_2, \dots, k_P]} &= \mathbf{f} - y_{k_1} e^{[k_1]} - y_{k_2} e^{[k_2]} - \dots - y_{k_P} e^{[k_P]} \\ &= (f_1, \dots, f_{k_1 - 1}, f_{k_1} - y_{k_1}, f_{k_1 + 1}, \dots, f_{k_2 - 1}, f_{k_2} - y_{k_2}, f_{k_2 + 1}, \dots, f_{k_P - 1}, f_{k_P} - y_{k_P}, f_{k_P + 1}, \dots, f_N), \end{aligned} \tag{3.4}$$

given that

$$\mathbf{b}_{k_1}^{[k_1, k_2, \dots, k_P]} = \mathbf{b}_{k_2}^{[k_1, k_2, \dots, k_P]} = \dots = \mathbf{b}_{k_P}^{[k_1, k_2, \dots, k_P]} = 0,$$

Definition 3.1 and relation (3.4) give

$$\mathbf{A}^{(k_1, k_2, \dots, k_P)} \mathbf{b}^{(k_1, k_2, \dots, k_P)} = \mathbf{f}^{(k_1, k_2, \dots, k_P)},$$

as a result

$$\mathbf{c}^{(k_1, k_2, \dots, k_P)} = \mathbf{b}^{(k_1, k_2, \dots, k_P)},$$

thus

$$\begin{bmatrix} \mathbf{W}^{(k_1,k_2,\dots,k_P)}(x_{k_1}) \\ \mathbf{W}^{(k_1,k_2,\dots,k_P)}(x_{k_2}) \\ \vdots \\ \mathbf{W}^{(k_1,k_2,\dots,k_P)}(x_{k_P}) \end{bmatrix} = \begin{bmatrix} \sum_{i=1,i\neq k_1,k_2,\dots,k_P}^{N} c_i^{(k_1,k_2,\dots,k_P)}\varphi(\|x_{k_1}-x_i\|) \\ \sum_{i=1,i\neq k_1,k_2,\dots,k_P}^{N} c_i^{(k_1,k_2,\dots,k_P)}\varphi(\|x_{k_2}-x_i\|) \\ \vdots \\ \sum_{i=1,i\neq k_1,k_2,\dots,k_P}^{N} c_i^{(k_1,k_2,\dots,k_P)}\varphi(\|x_{k_P}-x_i\|) \end{bmatrix}$$

$$= \begin{bmatrix} \sum_{i=1,i\neq k_1,k_2,\dots,k_P}^{N} b_i^{(k_1,k_2,\dots,k_P)}\varphi(\|x_{k_1}-x_i\|) \\ \sum_{i=1,i\neq k_1,k_2,\dots,k_P}^{N} b_i^{(k_1,k_2,\dots,k_P)}\varphi(\|x_{k_2}-x_i\|) \\ \vdots \\ \sum_{i=1,i\neq k_1,k_2,\dots,k_P}^{N} b_i^{(k_1,k_2,\dots,k_P)}\varphi(\|x_{k_P}-x_i\|) \end{bmatrix}$$

$$= \begin{bmatrix} \sum_{i=1}^{N} b_i^{(k_1,k_2,\dots,k_P)}\varphi(\|x_{k_1}-x_i\|) \\ \sum_{i=1}^{N} b_i^{(k_1,k_2,\dots,k_P)}\varphi(\|x_{k_2}-x_i\|) \\ \vdots \\ \sum_{i=1}^{N} b_i^{(k_1,k_2,\dots,k_P)}\varphi(\|x_{k_P}-x_i\|) \end{bmatrix}$$

$$= \begin{bmatrix} (\mathbf{A}\mathbf{b}^{[k_1,k_2,\dots,k_P]})_{k_1} \\ (\mathbf{A}\mathbf{b}^{[k_1,k_2,\dots,k_P]})_{k_2} \\ \vdots \\ (\mathbf{A}\mathbf{b}^{[k_1,k_2,\dots,k_P]})_{k_P} \end{bmatrix} = \begin{bmatrix} f_{k_1}-y_{k_1} \\ f_{k_2}-y_{k_2} \\ \vdots \\ f_{k_P}-y_{k_P} \end{bmatrix}.$$

Finally, the error vector $\mathbf{E}_{k_1k_2\dots k_P}$ is as follows:

$$\mathbf{E}_{k_1k_2\dots k_P} = \begin{bmatrix} f_{k_1}-\mathbf{W}^{(k_1,k_2,\dots,k_P)}(x_{k_1}) \\ f_{k_2}-\mathbf{W}^{(k_1,k_2,\dots,k_P)}(x_{k_2}) \\ \vdots \\ f_{k_P}-\mathbf{W}^{(k_1,k_2,\dots,k_P)}(x_{k_P}) \end{bmatrix} = \begin{bmatrix} y_{k_1} \\ y_{k_2} \\ \vdots \\ y_{k_P} \end{bmatrix}.$$

□

The cost function is defined by taking the ℓ_2 norm of P-dimensional error matrix $\mathbf{E}$ with entries $e_{k_1k_2\dots k_P}$ that are the ℓ_2 norm of $\mathbf{E}_{k_1k_2\dots k_P}$. Then, the Matlab function `fminbnd` minimizes the cost function and an optimal value of ε is obtained, also the number of iterations is achieved.

To clarify Theorem 3.3, we consider different values for P, $(P = 1, 2, 3)$.
By applying Theorem 3.3 for $P = 1$, we obtain

$$\mathbf{B}_{1\times 1} = x_k^{[k]}, \quad y_k = \frac{c_k}{x_k^{[k]}}, \quad k = 1, \dots, N,$$

and

$$\mathbf{E}_k = f_k - \mathbf{W}^{(k)}(x_k) = y_k, \quad k = 1, \dots, N.$$

These are equivalent to results obtained by Rippa's [29].
Also, for $P = 2$, we derive

$$\mathbf{B}_{2\times 2} = \begin{bmatrix} x_k^{[k]} & x_k^{[l]} \\ x_l^{[k]} & x_l^{[l]} \end{bmatrix}, \quad k = 1, \dots, N-1, \quad l = k+1, \dots, N,$$

and

$$y_k = \frac{\begin{vmatrix} c_k & x_k^{[l]} \\ c_l & x_l^{[l]} \end{vmatrix}}{|\mathbf{B}|}, \quad y_l = \frac{\begin{vmatrix} x_k^{[k]} & c_k \\ x_l^{[k]} & c_l \end{vmatrix}}{|\mathbf{B}|}.$$

Then

$$\mathbf{E}_{kl} = \begin{bmatrix} f_k - \mathbf{W}^{(k,l)}(x_k) \\ f_l - \mathbf{W}^{(k,l)}(x_l) \end{bmatrix} = \begin{bmatrix} y_k \\ y_l \end{bmatrix}, \; k = 1, \dots, N-1, \; l = k+1, \dots, N.$$

These are equivalent to results obtained by Azarboni et al. [2].

Now, we put P = 3, and present a new method that is called Leave-Three-Out Cross Validation (LThOCV) method. We have

$$\mathbf{B}_{3\times 3} = \begin{bmatrix} x_k^{[k]} & x_k^{[l]} & x_k^{[r]} \\ x_l^{[k]} & x_l^{[l]} & x_l^{[r]} \\ x_r^{[k]} & x_r^{[l]} & x_r^{[r]} \end{bmatrix},$$

such that $k < l < r, \quad k = 1, \dots, N-2, \quad l = k+1, \dots, N-1, \quad r = l+1, \dots, N.$ And

$$y_1 = \frac{\begin{vmatrix} c_k & x_k^{[l]} & x_k^{[r]} \\ c_l & x_l^{[l]} & x_l^{[r]} \\ c_r & x_r^{[l]} & x_r^{[r]} \end{vmatrix}}{|\mathbf{B}|}, \quad y_2 = \frac{\begin{vmatrix} x_k^{[k]} & c_k & x_k^{[r]} \\ x_l^{[k]} & c_l & x_l^{[r]} \\ x_r^{[k]} & c_r & x_r^{[r]} \end{vmatrix}}{|\mathbf{B}|}, \quad y_3 = \frac{\begin{vmatrix} x_k^{[k]} & x_k^{[l]} & c_k \\ x_l^{[k]} & x_l^{[l]} & c_l \\ x_r^{[k]} & x_r^{[l]} & c_r \end{vmatrix}}{|\mathbf{B}|}.$$

Using the AMLS method and the relation between Eqs. (2.4) and (2.8), $|\mathbf{B}|$, y_1, y_2 and y_3 are defined by

$$\begin{aligned}
|\mathbf{B}| = & \left[\sum_{i=0}^{n}(\mathbf{I}-\mathbf{A})^i\right]_{kk}\left[\sum_{i=0}^{n}(\mathbf{I}-\mathbf{A})^i\right]_{lr}\left[\sum_{i=0}^{n}(\mathbf{I}-\mathbf{A})^i\right]_{rl} - \left[\sum_{i=0}^{n}(\mathbf{I}-\mathbf{A})^i\right]_{kl}\left[\sum_{i=0}^{n}(\mathbf{I}-\mathbf{A})^i\right]_{lr}\left[\sum_{i=0}^{n}(\mathbf{I}-\mathbf{A})^i\right]_{rk} \\
& - \left[\sum_{i=0}^{n}(\mathbf{I}-\mathbf{A})^i\right]_{kk}\left[\sum_{i=0}^{n}(\mathbf{I}-\mathbf{A})^i\right]_{ll}\left[\sum_{i=0}^{n}(\mathbf{I}-\mathbf{A})^i\right]_{rr} + \left[\sum_{i=0}^{n}(\mathbf{I}-\mathbf{A})^i\right]_{kl}\left[\sum_{i=0}^{n}(\mathbf{I}-\mathbf{A})^i\right]_{lk}\left[\sum_{i=0}^{n}(\mathbf{I}-\mathbf{A})^i\right]_{rr} \\
& - \left[\sum_{i=0}^{n}(\mathbf{I}-\mathbf{A})^i\right]_{lk}\left[\sum_{i=0}^{n}(\mathbf{I}-\mathbf{A})^i\right]_{kr}\left[\sum_{i=0}^{n}(\mathbf{I}-\mathbf{A})^i\right]_{rl} + \left[\sum_{i=0}^{n}(\mathbf{I}-\mathbf{A})^i\right]_{ll}\left[\sum_{i=0}^{n}(\mathbf{I}-\mathbf{A})^i\right]_{kr}\left[\sum_{i=0}^{n}(\mathbf{I}-\mathbf{A})^i\right]_{rk},
\end{aligned}$$

$$\begin{aligned}
y_1 = & \frac{-\left[\sum_{i=0}^{n}(\mathbf{I}-\mathbf{A})^i\right]_{kl}\left[\sum_{i=0}^{n}(\mathbf{I}-\mathbf{A})^i\right]_{lr}\left[\sum_{i=0}^{n}(\mathbf{I}-\mathbf{A})^i\mathbf{f}\right]_{r}}{|\mathbf{B}|} + \frac{\left[\sum_{i=0}^{n}(\mathbf{I}-\mathbf{A})^i\right]_{ll}\left[\sum_{i=0}^{n}(\mathbf{I}-\mathbf{A})^i\right]_{kr}\left[\sum_{i=0}^{n}(\mathbf{I}-\mathbf{A})^i\mathbf{f}\right]_{r}}{|\mathbf{B}|} \\
& + \frac{\left[\sum_{i=0}^{n}(\mathbf{I}-\mathbf{A})^i\right]_{lr}\left[\sum_{i=0}^{n}(\mathbf{I}-\mathbf{A})^i\right]_{rl}\left[\sum_{i=0}^{n}(\mathbf{I}-\mathbf{A})^i\mathbf{f}\right]_{k}}{|\mathbf{B}|} + \frac{\left[\sum_{i=0}^{n}(\mathbf{I}-\mathbf{A})^i\right]_{kl}\left[\sum_{i=0}^{n}(\mathbf{I}-\mathbf{A})^i\right]_{rr}\left[\sum_{i=0}^{n}(\mathbf{I}-\mathbf{A})^i\mathbf{f}\right]_{l}}{|\mathbf{B}|} \\
& - \frac{\left[\sum_{i=0}^{n}(\mathbf{I}-\mathbf{A})^i\right]_{kr}\left[\sum_{i=0}^{n}(\mathbf{I}-\mathbf{A})^i\right]_{rl}\left[\sum_{i=0}^{n}(\mathbf{I}-\mathbf{A})^i\mathbf{f}\right]_{l}}{|\mathbf{B}|} - \frac{\left[\sum_{i=0}^{n}(\mathbf{I}-\mathbf{A})^i\right]_{ll}\left[\sum_{i=0}^{n}(\mathbf{I}-\mathbf{A})^i\right]_{rr}\left[\sum_{i=0}^{n}(\mathbf{I}-\mathbf{A})^i\mathbf{f}\right]_{k}}{|\mathbf{B}|},
\end{aligned}$$

$$\begin{aligned}
y_2 = & \frac{\left[\sum_{i=0}^{n}(\mathbf{I}-\mathbf{A})^i\right]_{kk}\left[\sum_{i=0}^{n}(\mathbf{I}-\mathbf{A})^i\right]_{lr}\left[\sum_{i=0}^{n}(\mathbf{I}-\mathbf{A})^i\mathbf{f}\right]_{r}}{|\mathbf{B}|} - \frac{\left[\sum_{i=0}^{n}(\mathbf{I}-\mathbf{A})^i\right]_{lk}\left[\sum_{i=0}^{n}(\mathbf{I}-\mathbf{A})^i\right]_{kr}\left[\sum_{i=0}^{n}(\mathbf{I}-\mathbf{A})^i\mathbf{f}\right]_{r}}{|\mathbf{B}|} \\
& - \frac{\left[\sum_{i=0}^{n}(\mathbf{I}-\mathbf{A})^i\right]_{lr}\left[\sum_{i=0}^{n}(\mathbf{I}-\mathbf{A})^i\right]_{rk}\left[\sum_{i=0}^{n}(\mathbf{I}-\mathbf{A})^i\mathbf{f}\right]_{k}}{|\mathbf{B}|} - \frac{\left[\sum_{i=0}^{n}(\mathbf{I}-\mathbf{A})^i\right]_{kk}\left[\sum_{i=0}^{n}(\mathbf{I}-\mathbf{A})^i\right]_{rr}\left[\sum_{i=0}^{n}(\mathbf{I}-\mathbf{A})^i\mathbf{f}\right]_{l}}{|\mathbf{B}|} \\
& + \frac{\left[\sum_{i=0}^{n}(\mathbf{I}-\mathbf{A})^i\right]_{kr}\left[\sum_{i=0}^{n}(\mathbf{I}-\mathbf{A})^i\right]_{rk}\left[\sum_{i=0}^{n}(\mathbf{I}-\mathbf{A})^i\mathbf{f}\right]_{l}}{|\mathbf{B}|} + \frac{\left[\sum_{i=0}^{n}(\mathbf{I}-\mathbf{A})^i\right]_{lk}\left[\sum_{i=0}^{n}(\mathbf{I}-\mathbf{A})^i\right]_{rr}\left[\sum_{i=0}^{n}(\mathbf{I}-\mathbf{A})^i\mathbf{f}\right]_{k}}{|\mathbf{B}|},
\end{aligned}$$

and

$$\begin{aligned} y_3 = &\frac{-\left[\sum_{i=0}^{n}(\mathbf{I}-\mathbf{A})^i\right]_{kk}\left[\sum_{i=0}^{n}(\mathbf{I}-\mathbf{A})^i\right]_{ll}\left[\sum_{i=0}^{n}(\mathbf{I}-\mathbf{A})^i\mathbf{f}\right]_{r}}{|\mathbf{B}|} + \frac{\left[\sum_{i=0}^{n}(\mathbf{I}-\mathbf{A})^i\right]_{kl}\left[\sum_{i=0}^{n}(\mathbf{I}-\mathbf{A})^i\right]_{lk}\left[\sum_{i=0}^{n}(\mathbf{I}-\mathbf{A})^i\mathbf{f}\right]_{r}}{|\mathbf{B}|} \\ &+ \frac{\left[\sum_{i=0}^{n}(\mathbf{I}-\mathbf{A})^i\right]_{kk}\left[\sum_{i=0}^{n}(\mathbf{I}-\mathbf{A})^i\right]_{rl}\left[\sum_{i=0}^{n}(\mathbf{I}-\mathbf{A})^i\mathbf{f}\right]_{l}}{|\mathbf{B}|} - \frac{\left[\sum_{i=0}^{n}(\mathbf{I}-\mathbf{A})^i\right]_{kl}\left[\sum_{i=0}^{n}(\mathbf{I}-\mathbf{A})^i\right]_{rk}\left[\sum_{i=0}^{n}(\mathbf{I}-\mathbf{A})^i\mathbf{f}\right]_{l}}{|\mathbf{B}|} \\ &- \frac{\left[\sum_{i=0}^{n}(\mathbf{I}-\mathbf{A})^i\right]_{lk}\left[\sum_{i=0}^{n}(\mathbf{I}-\mathbf{A})^i\right]_{rl}\left[\sum_{i=0}^{n}(\mathbf{I}-\mathbf{A})^i\mathbf{f}\right]_{k}}{|\mathbf{B}|} + \frac{\left[\sum_{i=0}^{n}(\mathbf{I}-\mathbf{A})^i\right]_{ll}\left[\sum_{i=0}^{n}(\mathbf{I}-\mathbf{A})^i\right]_{rk}\left[\sum_{i=0}^{n}(\mathbf{I}-\mathbf{A})^i\mathbf{f}\right]_{k}}{|\mathbf{B}|}. \end{aligned}$$

Then, the error vector E_{klr} in the LThOCV method is given by

$$\mathbf{E}_{klr}=\begin{bmatrix} f_k - \mathbf{W}^{(k,l,r)}(x_k) \\ f_l - \mathbf{W}^{(k,l,r)}(x_l) \\ f_r - \mathbf{W}^{(k,l,r)}(x_r) \end{bmatrix} = \begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix}.$$

The basis of the LThOCV method is based on Algorithm 2.

Algorithm 2

```
Fix $\varepsilon$. Perform an eigen-decomposition
    $\mathbf{I}-\mathbf{A}=\mathbf{X}\Lambda\mathbf{X}^{-1}$
Initialize $\mathbf{V}^{(0)}=\mathbf{f}$ and $\mathbf{M}^{(0)}=\mathbf{I}$
for $n=1,2,...$ do
  Perform the updates
    $\mathbf{V}^{(n)}=\mathbf{f}+(\mathbf{I}-\mathbf{A})\mathbf{V}^{(n-1)}$
    $\mathbf{M}^{(n)}=\Lambda\mathbf{M}^{(n-1)}+\mathbf{I}$
    Compute the vector $\mathbf{E}_{klr}^{(n)}$ as the componentwise quotient
    of $\mathbf{V}^{(n)}$ and the diagonal of $\mathbf{X}\mathbf{M}^{(n)}\mathbf{X}^{-1}$
  $e_{klr}^{(n)}$ is the $\ell_2$ norm of $\mathbf{E}_{klr}^{(n)}$
  From the 3-dimensional cost matrix $\mathbf{E}^{(n)}=(e_{klr}^{(n)})_{k<l<r}$
    $ceps(n)=norm(\mathbf{E}^{(\mathbf{n})},2)$
  if $(ceps(n)-ceps(n-1))<tol$ then
    Stop the iteration
  end if
end for
```

To speed up the computational operation we use the eigen-decomposition for the matrix $\mathbf{I}-\mathbf{A}$. By using Algorithm 2 the cost function "$ceps(n)$" is defined (See Program 2). Utilizing the Matlab function `fminbnd` this cost function is minimized, and an optimal value of ε is specified, also the number of iterations is obtained.

3.3. Iterative LThOCV (ItLThOCV) for iterative AMLS Method.

In this subsection, we declare an iterative LThOCV (ItLThOCV) algorithm for the iterative AMLS method as Algorithm 3.

For this approach, we remove three points x_k, x_l and x_r from the data set such that $k=1,\dots,N-2,\quad l=k+1,\dots,N-1$, and $r=l+1,\dots,N$, then, use the iterative AMLS method to this set of new data points until the stopping criterion is satisfied. In Eq. (3.5), an RBF interpolation is obtained by deleting three points. The cost function "$ceps(n)$" (See Program 3) is defined by getting the ℓ_2 norm of the 3-dimensional error matrix $\mathbf{E}^{(n)}$ with

Algorithm 3

```
Fix $\varepsilon$
for $n = 1, 2, ...$ do
  for $k = 1, 2, ..., N-2$ do
    for $l = k+1, \dots, N-1$ do
      for $r = l+1, \dots, N$ do
        Let
          $$\psi_f^{(0)(k,l,r)}(x) = \sum_{j=1}^{N-3} f_j^{(k,r,l)} \varphi\left(\left\|x - x_j^{(k,l,r)}\right\|\right)$$
        for $j = 1, 2, ..., N-3$ do
          Compute residuals at the datasites
            $$r_j^{(n)(k,l,r)} = f_j^{(k,l,r)} - \psi_f^{(n-1)(k,l,r)}\left(x_j^{(k,l,r)}\right)$$
        end for
        Compute the correction
          $$u(x) = \sum_{j=1}^{N-3} r_j^{(n)(k,l,r)} \varphi\left(\left\|x - x_j^{(k,l,r)}\right\|\right)$$
        Update
          $$\psi_f^{(n)(k,l,r)}(x) = \psi_f^{(n-1)(k,l,r)}(x) + u(x) \qquad (3.5)$$
        Compute the error estimate for the $k^{th}$ , $l^{th}$ and $r^{th}$ data points
          $$\mathbf{E}_{klr}^{(n)(k,l,r)} = \begin{bmatrix} f(x_k) - \psi_f^{(n)(k,l,r)}(x_k) \\ f(x_l) - \psi_f^{(n)(k,l,r)}(x_l) \\ f(x_r) - \psi_f^{(n)(k,l,r)}(x_r) \end{bmatrix}$$
        $e_{klr}^{(n)(k,l,r)}$ is the $\ell_2$ norm of $\mathbf{E}_{klr}^{(n)(k,l,r)}$
      end for
      From  the 3-dimensional cost matrix $\mathbf{E}^{(n)} = (e_{klr}^{(n)(k,l,r)})_{k<l<r}$ $ceps(n) = norm(\mathbf{E}^{(n)}, 2)$
      if $(ceps(n) - ceps(n-1)) < tol$ then
        Stop the iteration
      end if
    end for
  end for
end for
```

entries $e_{klr}^{(n)(k,l,r)} = \left\|\mathbf{E}_{klr}^{(n)(k,l,r)}\right\|_2$ in which

$$\mathbf{E}_{klr}^{(n)(k,l,r)} = \begin{bmatrix} f(x_k) - \psi_f^{(n)(k,l,r)}(x_k) \\ f(x_l) - \psi_f^{(n)(k,l,r)}(x_l) \\ f(x_r) - \psi_f^{(n)(k,l,r)}(x_r) \end{bmatrix}.$$

In order to expedite up the computational operation we use the eigen-decomposition for the interpolation matrix, and the residual value is calculated for all points as follows:

$$f(x_j) - \psi_f^{(n)(k,l,r)}(x_j), \quad j = 1, \dots, N.$$

Applying the Matlab function `fminbnd` the cost function is minimized, and an optimal value of ε is specified, also the number of iterations is achieved.

4. Numerical Experiments

In this section, three examples are tested to assess the performance of present methods. First, these methods are used for interpolating two test functions and then, a partial differential problem is solved. We consider a uniform data set from size N in two dimensions, and examine the accuracy of present methods by applying the absolute error, namely $e(x)$, and the Root-Mean-Square-Error (RMSE) overall N data points as follows:

$$e(x_i) = |\mathbf{F}(x_i) - \mathbf{W}(x_i)|, \quad i = 1, \dots, N, \tag{4.1}$$

$$RMSE(\varepsilon) = \sqrt{\frac{\sum_{i=1}^{N} e^2(x_i)}{N}}, \tag{4.2}$$

where $\mathbf{W}$ is the RBF approximation and $\mathbf{F}$ is the exact solution. All of the numerical computations are performed in MATLAB R2018b, on a PC with an Intel(R) Core(TM) i5-7500, CPU 3.40GHz, 8GB(RAM).

Example 1. [12–14] Let us consider a modified Franke function on the unit square as follows:

$$\begin{aligned} g(x,y) =& 0.75\exp(-\frac{(9x-2)^2+(9y-2)^2}{4}) + 0.75\exp(-\frac{(9x+1)^2}{49} - \frac{9y+1}{10}) \\ &+0.5\exp(-\frac{(9x-7)^2+(9y-3)^2}{4}) - 0.2\exp(-(9x-4)^2 - (9y-7)^2), \end{aligned}$$

$$f(x,y) = 15g(x,y)\exp(-\frac{1}{1-(2x-1)^2})\exp(-\frac{1}{1-(2y-1)^2}).$$

Numerical results in Table 2 show the RMSE values, the optimal shape parameter ε, the number of repetitions required for the AMLS method and CPU time for various values of N. The RMSE rate is reduced by increasing N for each method. We use the radial basis function in the form of (2.5). Figure 1 presents the approximate and exact solutions along with the absolute error with $N = 49$ and $\varepsilon = 0.8195$. Figure 2 displays the RMSE as a function of $\varepsilon \in [0.5, 3]$ for $N = 49$, where the optimal shape parameter is $\varepsilon = 0.9167$ and the RMSE is $2.441e-18$. Furthermore, in [13], the RMSE for $N = 1089$ of the AMLS direct LOOCV method is $2.48e-4$, of the AMLS iterative LOOCV method is $2.30e-4$, of the Shepard direct LOOCV method is $2.73e-4$, of the Shepard iterative LOOCV method is $2.77e-4$ and of the Ridge method is $2.50e-4$. Numerical results show that our methods have very high accuracy with a small number of data points compared to the methods mentioned in [13].

Table 2. Numerical results of present methods for Example 1.

Present methods		$N=25$	$N=36$	$N=49$
ItLTOCV	RMSE	$1.7471e-16$	$1.0781e-16$	$8.7223e-17$
	ε	0.9431	0.8691	0.8279
	no. iter.	3	3	3
	CPU time (s)	0.11	0.15	0.17
LThOCV	RMSE	$9.8805e-17$	$9.6940e-17$	$9.1295e-17$
	ε	0.9713	0.9009	0.8585
	no. iter.	3	3	3
	CPU time (s)	0.18	0.44	0.58
ItLThOCV	RMSE	$2.4289e-16$	$7.6638e-17$	$7.0324e-17$
	ε	0.9232	0.8589	0.8195
	no. iter.	3	3	3
	CPU time (s)	0.31	0.58	3.09

Example 2. [2] In this example, we consider peaks function in Matlab.

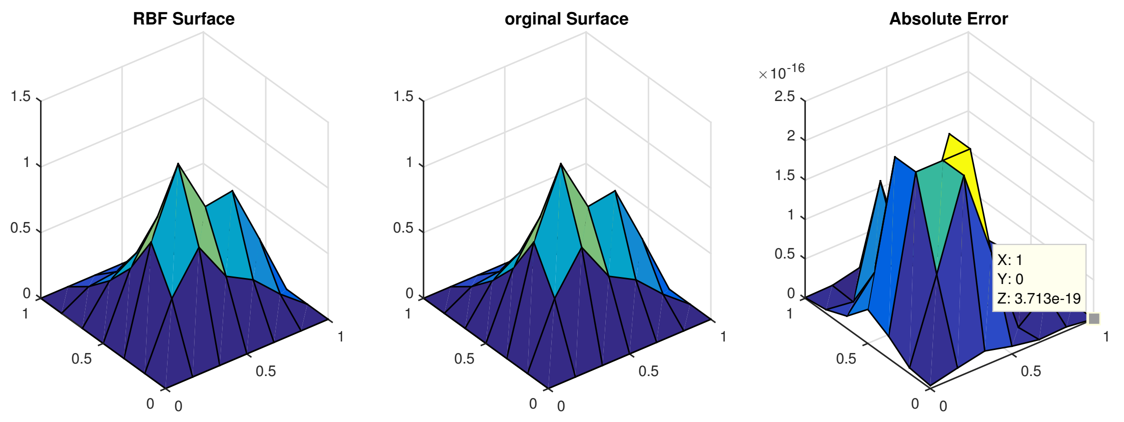

FIGURE 1. Approximate (left) and exact (middle) solutions along with the absolute error (right) with $N = 49$ and $\varepsilon = 0.8195$ for Example 1.

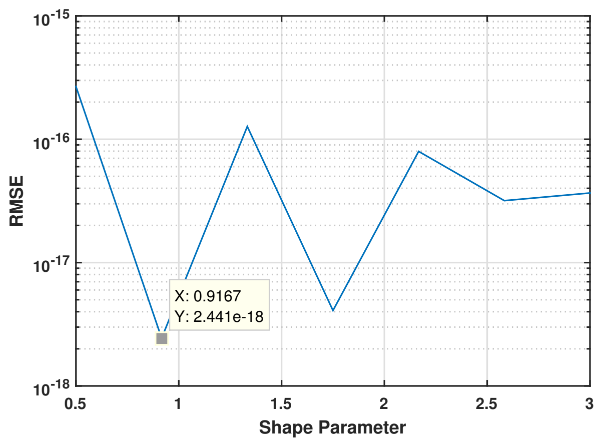

FIGURE 2. RMSE as a function of ε with $N = 49$ on a linear − log scale for Example 1.

Numerical results in Table 3 have a similar trend with Table 2. We use the radial basis function in the form of (2.5). Figure 3 portrays the approximate and exact solutions along with the absolute error with $N = 49$ and $\varepsilon = 0.1529$. Figure 4 demonstrates the RMSE as a function of $\varepsilon \in [0.001, 1]$ for $N = 49$, where the optimal shape parameter is $\varepsilon = 0.8335$ and the RMSE is $9.713e - 17$. Moreover, in [2], the RMSE for $N = 400$ of the LOOCV method is $1.1e - 5$ and of the LTOCV method is $1e - 7$. Numerical results show that our methods have very high accuracy with a small number of data points compared to the methods mentioned in [2].

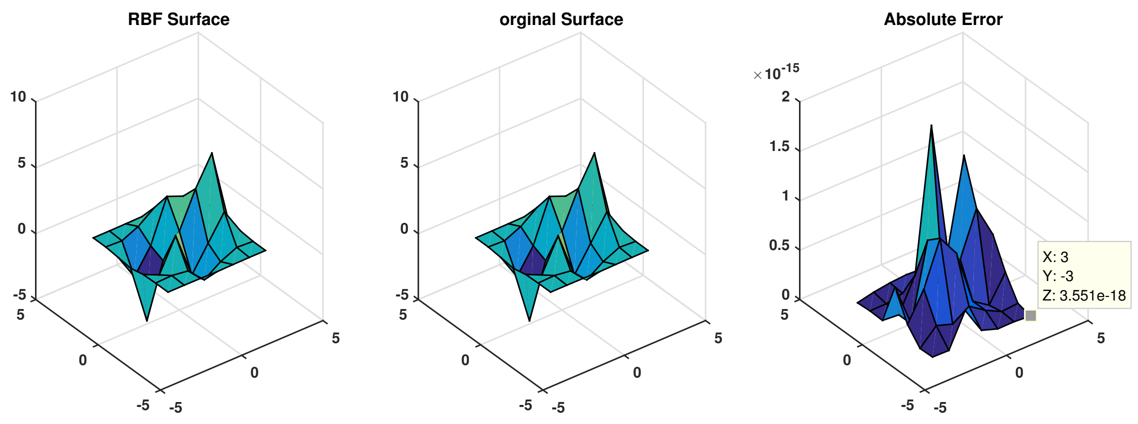

FIGURE 3. Approximate (left) and exact (middle) solutions along with the absolute error (right) with $N = 49$ and $\varepsilon = 0.1529$ for Example 2.

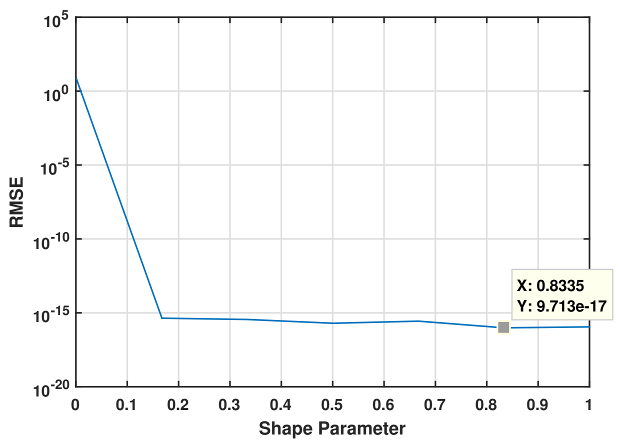

FIGURE 4. RMSE as a function of ε with $N = 49$ on a linear − log scale for Example 2.

TABLE 3. Numerical results of present methods for Example 2.

Present methods		$N = 25$	$N = 36$	$N = 49$
ItLTOCV	RMSE	$8.8350e-16$	$7.4998e-16$	$4.3312e-16$
	ε	0.1744	0.1601	0.1514
	no. iter.	3	3	3
	CPU time (s)	0.13	0.16	0.19
LThOCV	RMSE	$1.1077e-15$	$6.3897e-16$	$5.2103e-16$
	ε	0.1774	0.1636	0.1532
	no. iter.	3	3	3
	CPU time (s)	0.25	0.35	0.85
ItLThOCV	RMSE	$5.2799e-16$	$4.2763e-16$	$4.2006e-16$
	ε	0.1772	0.1631	0.1529
	no. iter.	3	3	3
	CPU time (s)	0.41	0.57	3.47

Example 3. [2] Let us consider Poisson equation:

$$-u_{xx} - u_{yy} = \pi^2 \sin(\pi x) \sin(\pi y); \qquad -1 \leq x, y \leq 1,$$

with boundary conditions

$$\begin{aligned}
&u(x,-1) = 1 - 2x; \qquad -1 \leq x \leq 1,\\
&u(x,1) = 1; \qquad -1 \leq x \leq 1,\\
&u(-1,y) = 2 - y; \qquad -1 \leq y \leq 1,\\
&u(1,y) = y; \qquad -1 \leq y \leq 1.
\end{aligned}$$

The exact solution is:

$$u(x,y) = 1 - x + xy + \frac{1}{2}\sin(\pi x)\sin(\pi y); \qquad -1 < x, y < 1.$$

Numerical results in Table 4 have a similar trend with Table 2. We use the Guassian radial basis function $\varphi(r) = e^{-\varepsilon^2 r^2}$. Figure 5 indicates the approximate and exact solutions along with the absolute error with $N = 100$ and $\varepsilon = 0.6001$. Figure 6 illustrates RMSE as a function of $\varepsilon \in [0.6, 1.4]$ for $N = 100$, where the optimal shape parameter is $\varepsilon = 0.6889$ and the RMSE is $8.587e-06$. Moreover, in [2], the RMSE for $N = 400$ of the LOOCV method is $2.1e-5$ and of the LTOCV method is $3.8178e-6$. Numerical results show that our methods are more accurate with a small number of data points compared to the methods mentioned in [2]. Note that the coefficient matrix of partial differential equations is non-symmetric, but present methods are still quite efficient using strictly positive definite RBFs, even if they are not theoretically guaranteed.

TABLE 4. Numerical results of present methods for Example 3.

Present methods		$N = 49$	$N = 81$	$N = 100$
ItLTOCV	RMSE	$3.2192e-04$	$3.2138e-05$	$3.7530e-06$
	ε	0.6001	0.6001	0.6001
	no. iter.	3	3	3
	CPU time (s)	1.04	3.50	6.57
LThOCV	RMSE	$4.4358e-03$	$5.2021e-04$	$6.6675e-05$
	ε	1.0021	0.9777	0.8781
	no. iter.	5	4	4
	CPU time (s)	5.10	26.27	38.64
ItLThOCV	RMSE	$3.2192e-04$	$3.2138e-05$	$3.7530e-06$
	ε	0.6001	0.6001	0.6001
	no. iter.	3	3	3
	CPU time (s)	15.77	104.16	305.0391

5. Conclusion

In this paper, we introduced three numerical schemes to select the optimal shape parameter for the RBF method. These are based on Rippa's theory to remove data points from the data set, and results obtained by Fasshauer and Zhang for the iterative AMLS method. First, we proposed the ItLTOCV algorithm for the iterative AMLS method. Then, we defined a generalized formula to obtain the error term by removing P data from the data set, this method called the LPOCV method and used it for P = 3. Finally, we presented the ItLThOCV algorithm for the iterative AMLS method. Using the Tables, it can be said that with increasing N, the RMSE decreases for each method. Also, the RMSE in our methods is less than the RMSE of other methods at low numbers of data points, but the computational time with the growth of deleted points is longer. All of the descriptions indicate the effectiveness of our methods.

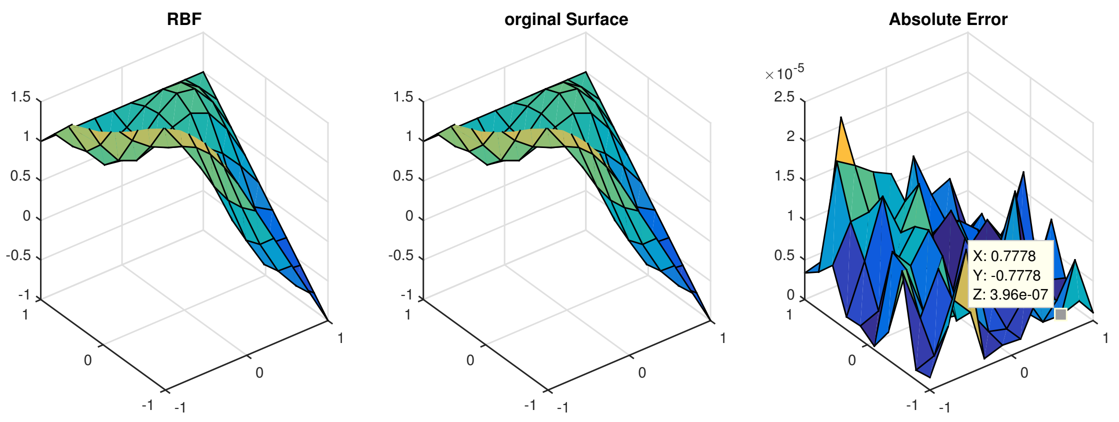

FIGURE 5. Approximate (left) and exact (middle) solutions along with the absolute error (right) with $N = 100$ and $\varepsilon = 0.6001$ for Example 3.

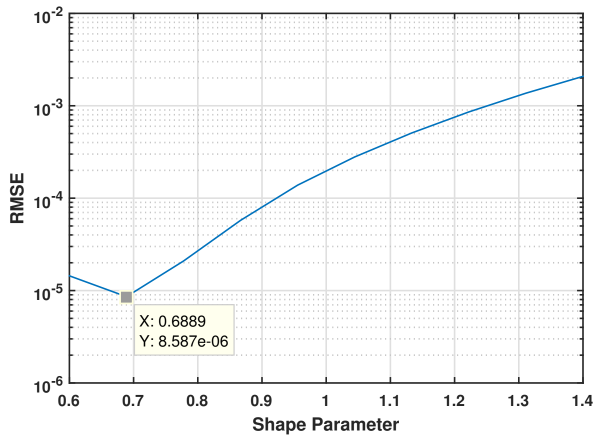

FIGURE 6. RMSE as a function of ε with $N = 100$ on a linear − log scale for Example 3.

APPENDIX: PROGRAMS IN MATLAB

Program 1:

```
function ceps=CostEpsItLTOCV(ep1,phi,T,f,m)
ep1;
A=phi(ep1,T);
I=eye((m+1)^2);
[x1,D1]=eig(A);
for j=1:(m+1)^2
for k=1:(m+1)^2
E(j,k)=f(j);
end
```

```
end
e1=zeros((m+1)^2);
ceps(1)=norm(e1,2);
e2=zeros((m+1)^2);
n=2;
S1=x1\E;
for k=1:((m+1)^2)-1
S=(I-D1)*S1+(D1/x1)*(diag(diag(E,-k),-k)+...
diag(diag(E)));
D=diag(diag(x1*S,-k),-k)+diag(diag(x1*S));
for i=1:((m+1)^2)-k
en(i,i+k)=sqrt(D(i,i)^2+D(i+k,i)^2);
end
end
ceps(n)=norm(en,2);
while (ceps(n)-ceps(n-1))>1e-1
n=n+1;
S1=x1\E;
for k=1:((m+1)^2)-1
S=(I-D1)*S1+(D1/x1)*(diag(diag(E,-k),-k)+...
diag(diag(E)));
D=diag(diag(x1*S,-k),-k)+diag(diag(x1*S));
for i=1:((m+1)^2)-k
en(i,i+k)=sqrt(D(i,i)^2+D(i+k,i)^2);
end
end
ceps(n)=norm(en,2);
end
n
ceps=ceps(n);
end
```

Program 2:

```
function ceps=CostEpsLThOCV(ep1,phi,T,f,m)
ep1;
A=phi(ep1,T);
I=eye((m+1)^2);
[x1,D]=eig(I-A);
T1=(x1*(D/x1));
v=f;
M=I;
v=T1*v+f;
M=D*M+I;
T2=(x1*(M/x1));
```

```
ceps1=zeros((m+1)^2,(m+1)^2,(m+1)^2);
ceps2=zeros((m+1)^2,1);
ceps(1)=0;
n=2;
for  p=1:((m+1)^2-2)
for  q=p+1:((m+1)^2-1)
for  u=q+1:(m+1)^2
B=T2(p,p)*T2(u,q)*T2(q,u)-T2(q,p)*T2(u,q)*T2(p,u) -...
T2(p,p)*T2(q,q)*T2(u,u)+T2(q,p)*T2(p,q)*T2(u,u) -...
T2(p,q)*T2(u,p)*T2(q,u)+T2(q,q)*T2(u,p)*T2(p,u);
if(B==0)
ceps1=ceps1;
else
ce1=(-v(u)*T2(q,p)*T2(u,q)+v(u)*T2(q,q)*T2(u,p)+...
v(p)*T2(u,p)*T2(q,u)+v(q)*T2(q,p)*T2(u,u) -...
v(q)*T2(u,p)*T2(q,u)-v(p)*T2(q,q)*T2(u,u))/B;
ce2=(v(u)*T2(p,p)*T2(u,q)-v(u)*T2(p,q)*T2(u,p) -...
v(p)*T2(u,q)*T2(p,u)-v(q)*T2(p,p)*T2(u,u)+...
v(q)*T2(u,p)*T2(p,u)+v(p)*T2(p,q)*T2(u,u))/B;
ce3=(-v(u)*T2(p,p)*T2(q,q)+v(u)*T2(q,p)*T2(p,q)+...
v(q)*T2(p,p)*T2(q,u)-v(q)*T2(q,p)*T2(p,u) -...
v(p)*T2(p,q)*T2(q,u)+v(p)*T2(q,q)*T2(p,u))/B;
ceps1(p,q,u)=sqrt((ce1)^2+(ce2)^2+(ce3)^2);
end
end
end
ceps2(p+2,1)=norm(ceps1(:,:,p+2),2);
end
ceps(n)=norm(ceps2,2);
while   (ceps(n)-ceps(n-1))>1e-1
        n=n+1;
        v=T1*v+f;
        M=D*M+I;
        T2=(x1*(M/x1));
for  p=1:((m+1)^2-2)
for  q=p+1:((m+1)^2-1)
for  u=q+1:(m+1)^2
B=T2(p,p)*T2(u,q)*T2(q,u)-T2(q,p)*T2(u,q)*T2(p,u) -...
T2(p,p)*T2(q,q)*T2(u,u)+T2(q,p)*T2(p,q)*T2(u,u) -...
T2(p,q)*T2(u,p)*T2(q,u)+T2(q,q)*T2(u,p)*T2(p,u);
if(B==0)
ceps1=ceps1;
else
ce1=(-v(u)*T2(q,p)*T2(u,q)+v(u)*T2(q,q)*T2(u,p)+...
v(p)*T2(u,p)*T2(q,u)+v(q)*T2(q,p)*T2(u,u) -...
v(q)*T2(u,p)*T2(q,u)-v(p)*T2(q,q)*T2(u,u))/B;
```

```
ce2=(v(u)*T2(p,p)*T2(u,q)-v(u)*T2(p,q)*T2(u,p) -...
v(p)*T2(u,q)*T2(p,u)-v(q)*T2(p,p)*T2(u,u)+...
v(q)*T2(u,p)*T2(p,u)+v(p)*T2(p,q)*T2(u,u))/B;
ce3=(-v(u)*T2(p,p)*T2(q,q)+v(u)*T2(q,p)*T2(p,q)+...
v(q)*T2(p,p)*T2(q,u)-v(q)*T2(q,p)*T2(p,u) -...
v(p)*T2(p,q)*T2(q,u)+v(p)*T2(q,q)*T2(p,u))/B;
ceps1(p,q,u)=sqrt((ce1)^2+(ce2)^2+(ce3)^2);
end
end
end
ceps2(p+2,1)=norm(ceps1(:,:,p+2),2);
end
ceps(n)=norm(ceps2,2);
end
n
ceps=ceps(n);
end
```

Program 3:

```
function ceps=CostEpsItLThOCV(ep1,phi,T,f,m)
ep1;
A=phi(ep1,T);
I=eye((m+1)^2);
[x1,D1]=eig(A);
for j=1:(m+1)^2
for k=1:(m+1)^2
E(j,k)=f(j);
end
end
e1=zeros((m+1)^2,1);
ceps(1)=norm(e1,2);
e2=zeros((m+1)^2,1);
n=2;
ee=zeros((m+1)^2,(m+1)^2,(m+1)^2);
S1=x1\E;
for k=1:((m+1)^2)-2
for r=k+1:((m+1)^2)-1
S=(I-D1)*S1+(D1/x1)*(diag(diag(E,-r),-r)+...
diag(diag(E,-k),-k)+diag(diag(E)));
D=diag(diag(x1*S,-r),-r)+diag(diag(x1*S,-k),-k)+...
diag(diag(x1*S));
for i=1:((m+1)^2)-r
ee(i,i+k,i+r)=sqrt(D(i,i)^2+D(i+k,i)^2+D(i+r,i)^2);
end
```

```
end
en(k+2,1)=norm(ee(:,:,k+2),2);
end
ceps(n)=norm(en,2);
while (ceps(n)-ceps(n-1))>1e-1
n=n+1;
S1=x1\E;
for k=1:((m+1)^2)-2
for r=k+1:((m+1)^2)-1
S=(I-D1)*S1+(D1/x1)*(diag(diag(E,-r),-r)+...
diag(diag(E,-k),-k)+diag(diag(E)));
D=diag(diag(x1*S,-r),-r)+diag(diag(x1*S,-k),-k)+...
diag(diag(x1*S));
for i=1:((m+1)^2)-r
ee(i,i+k,i+r)=sqrt(D(i,i)^2+D(i+k,i)^2+D(i+r,i)^2);
end
end
en(k+2,1)=norm(ee(:,:,k+2),2);
end
ceps(n)=norm(en,2);
end
ceps=ceps(n);
n
end
```

References

[1] M. Abbaszadeh and M. Dehghan, *Direct meshless local Petrov–Galerkin (DMLPG) method for time-fractional fourth-order reaction–diffusion problem on complex domains*, Computers & Mathematics with Applications, *79*(3) (2019), 876–888.

[2] H. R. Azarboni, M. Keyanpour, and M. Yaghouti, *Leave-Two-Out Cross Validation to optimal shape parameter in radial basis functions*, Engineering Analysis with Boundary Elements, *100* (2019), 204–210.

[3] M. D. Buhmann, *Radial basis functions: theory and implementations*, Cambridge university press, 2003.

[4] R. E. Carlson and T. A. Foley, *The parameter R^2 in multiquadric interpolation*, Computers & Mathematics with Applications, *21*(9) (1991), 29–42.

[5] W. Chen, Z.-J. Fu, and C.-S. Chen, *Recent advances in radial basis function collocation methods*, Springer, 2014.

[6] M. Dehghan and A. Shokri, *A meshless method for numerical solution of the one-dimensional wave equation with an integral condition using radial basis functions*, Numerical Algorithms, *52*(3) (2009), 461–477.

[7] G. E. Fasshauer, *Approximate moving least-squares approximation with compactly supported radial weights*, Meshfree methods for partial differential equations, Springer, 105–116, 2003.

[8] G. E. Fasshauer, *Approximate moving least-squares approximation: A fast and accurate multivariate approximation method*, Curve and surface fitting: Saint-Malo, (2002), 139–148.

[9] G. E. Fasshauer, *Toward approximate moving least squares approximation with irregularly spaced centers*, Computer Methods in Applied Mechanics and Engineering, *193*(12-14) (2004), 1231–1243.

[10] G. E. Fasshauer, *Meshfree Approximation Methods with MATLAB*, World Scientific, 2007.

[11] G. E. Fasshauer and J. G. Zhang, *Recent results for moving least squares approximation*, Geometric Modeling and Computing, (2004), 163–176.

[12] G. E. Fasshauer and J. G. Zhang, *Scattered data approximation of noisy data via iterated moving least squares*, Proceedings of Curve and Surface Fitting: Avignon, (2006), 150–159.
[13] G. E. Fasshauer and J. G. Zhang, *On choosing "optimal" shape parameters for RBF approximation*, Numerical Algorithms, *45*(1-4) (2007), 345–368.
[14] G. E. Fasshauer and J. G. Zhang, *Iterated approximate moving least squares approximation*, Advances in Meshfree Techniques, Springer, 221–239, 2007.
[15] B. Fornberg and N. Flyer, *Solving PDEs with radial basis functions*, Acta Numerica, *24* (2015), 215–258.
[16] R. Franke, *Scattered data interpolation: tests of some methods*, Mathematics of Computation, *38*(157) (1982), 181–200.
[17] M. Ghorbani, *Diffuse element kansa method*, Applied Mathematical Sciences, *4*(12) (2010), 583–594.
[18] E. Giannaros, A. Kotzakolios, V. Kostopoulos, and G. Campoli, *Hypervelocity impact response of CFRP laminates using smoothed particle hydrodynamics method: Implementation and validation*, International Journal of Impact Engineering, *123* (2019), 56–69.
[19] R. L. Hardy, *Multiquadric equations of topography and other irregular surfaces*, Journal of Geophysical Research, *76*(8) (1971), 1905–1915.
[20] V. R. Hosseini, E. Shivanian, and W. Chen, *Local radial point interpolation (MLRPI) method for solving time fractional diffusion-wave equation with damping*, Journal of Computational Physics, *312* (2016), 307–332.
[21] A. J. Khattak and S. Tirmizi, *Application of meshfree collocation method to a class of nonlinear partial differential equations*, Engineering analysis with boundary elements, *33*(5) (2009), 661–667.
[22] P. Lancaster and K. Salkauskas, *Surfaces generated by moving least squares methods*, Mathematics of computation, *37*(155) (1981), 141–158.
[23] F. Lanzara, V. Maz'ya, and G. Schmidt, *Approximate approximations from scattered data*, Journal of Approximation Theory, *145*(2) (2007), 141–170.
[24] X. Li and H. Dong, *Analysis of the element-free Galerkin method for Signorini problems*, Applied Mathematics and Computation, *346* (2019), 41–56.
[25] Q. Liu, F. Liu, Y. T. Gu, P. Zhuang, J. Chen, and I. Turner, *A meshless method based on Point Interpolation Method (PIM) for the space fractional diffusion equation*, Applied Mathematics and Computation, *256* (2015), 930–938.
[26] V. Maz'ya and G. Schmidt, *On quasi-interpolation with non-uniformly distributed centers on domains and manifolds*, Journal of Approximation Theory, *110*(2) (2001), 125–145.
[27] V. Maz'ya, *A new approximation method and its applications to the calculation of volume potentials. Boundary point method*, DFG-Kolloquium des DFG-Forschungsschwerpunktes "Randelementmethoden, 1991.
[28] C. A. Micchelli, *Interpolation of scattered data: distance matrices and conditionally positive definite functions*, Constructive Approximation, *2* (1986), 11–22.
[29] S. Rippa, *An algorithm for selecting a good value for the parameter c in radial basis function interpolation*, Advances in Computational Mathematics, *11*(2-3) (1999), 193–210.
[30] M. Scheuerer, *An alternative procedure for selecting a good value for the parameter c in RBF-interpolation*, Advances in Computational Mathematics, *34*(1) (2011), 105–126.
[31] I. J. Schoenberg, *Metric spaces and completely monotone functions*, Annals of Mathematics, *39* (1938), 811–841.