Solving Advection-diffusion Equation with Local Radial Basis Function-based Differential Quadrature Method

Hongwei Tang, Jie Wu

Student Number: BX1701906,
Department of Aerodynamics,
Nanjing University of Aeronautics and Astronautics

Correspondence
Email: thw1021@nuaa.edu.cn

Summary

In the past decades, intensive research have been put into radial basis functions (RBFs) for multivariate data and function interpolation. Their results show that RBFs constitute a powerful framework for interpolating or approximating data on non-uniform grids. While differential quadrature (DQ) method is also a widely-used numerical approach for derivative approximation, which can obtain accurate results by using a considerably small number of grid points. In this paper, the advantages of the DQ approximation and RBFs are combined to provide an efficient numerical discretization method for solving partial differential equations, which is a derivative approximation approach and is newly mesh-free. In this method, the RBFs are taken as the test functions in the DQ approximation to compute the weighting coefficients. Then the partial derivative of a function with respect to an independent variable is approximated by a linear weighted sum of functional values at local grid points. Once the derivatives are fully discretized, the rusultant algebraic equations can be solved either by direct methods or iterative methods.

KEYWORDS:
RBFs, DQ method, mesh-free method

1 | BASIC THEORY

1.1 | Radial Basis Functions (RBFs) and Function Approximation

A radical basis function, denoted by $\varphi(||\mathbf{x}-\mathbf{x}_j||_2)$, is a continuous spline which depends on the seperation distances of subset of scatted points $\mathbf{x} \in \Omega \subset R^d$, $d = 1, 2, 3$ denotes the spatial dimension. There are many RBFs (expression of φ) available. The most commonly used RBFs are

- Multiquadrics (MQ): $\varphi(r) = \sqrt{r^2 + c^2}$
- This-plate splines (TPS): $\varphi(r) = r^2 \log(r)$
- Gaussians: $\varphi(r) = e^{-cr^2}$
- Inverse multiquadrics: $\varphi(r) = \frac{1}{\sqrt{r^2+c^2}}$

where $r = ||\mathbf{x}-\mathbf{x}_j||_2$ and shape parameter c is a positive constant.

Simply, the RBF interpolation technique can be described as following: if the function values of a function $f(\mathbf{x})$ are known on a set of scattered points $\mathbf{x} \in \Omega \subset R^d$, the approximation of $f(\mathbf{x})$ can be written as a linear combination of N radial basis functions,

$$f(x) \cong \sum_{j=1}^{N} \lambda_j \varphi(||\mathbf{x}-\mathbf{x}_j||_2) + \psi(x) \tag{1}$$

where N is the number of centers or sometimes called knots $\mathbf{x}$, $\mathbf{x} = (x_1, x_2, \ldots, x_d)$, d is the dimension of the problem, λ's are coefficients to be determined and φ is the radial basis function. Equation (1) can be written without the additional polynominal ψ.

Among the RBFs tested by Franke, Hardy's multiquadrics (MQ) were ranked the best in accuracy, followed by thin plate splines (TPS). In this paper, I choose MQ as radial basis function for calculating the weighting coefficients. In practice, it was found that the choice of c can greatly affect the accuracy of approximation. By increasing c, the root-mean-square error of the goodness-of-fit dropped to a minimum value and then grew rapidly thereafter. This is due to the fact that the MQ coefficient matrix becomes ill-conditioned when $c^2 \gg r^2$. How to choose the optimal shape parameter remains an open problem. No mathematical theory has been developed so far to determine such an optimal value.

1.2 | Differential Quadrature (DQ) Method for Derivative Approximation

The basic idea of the DQ mehod is that any derivative can be approximated by a linear weighted sum of functional values at soem mesh points. This ieda is modified that any spatial derivative can be approximated by a linear weighted sum of all the functional values in the whole two-dimensional domain. The new DQ approximation for the mth order derivative with respect to x, $f_x^{(m)}$, and the nth order derivative with respect to y, $f_y^{(n)}$ can be written as

$$f_x^{(m)}(x_k, y_k) = \sum_{kl=1}^{N} w_{k,kl}^{(m)} f(x_{kl}, y_{kl}) \tag{2}$$

$$f_y^{(n)}(x_k, y_k) = \sum_{kl=1}^{N} \overline{w}_{k,kl}^{(n)} f(x_{kl}, y_{kl}) \tag{3}$$

where N is the total number of points in the whole computational domain.

1.2.1 | Global Radial Basis Function-based Differential Quadrature (RBF-DQ) Method

Suppose that the solution of a partial differential equation is continuous, which can be approximated by MQ RBFs, and only a constant is included in the polynomial term $\varphi(\mathbf{x})$. Then the function in the domain can be approximated by MQ RBFs as

$$f(x, y) = \sum_{j=1, j\neq i}^{N} \lambda_j g_j(x, y) + \lambda_i \tag{4}$$

where $g_j(x, y) = \sqrt{(x - x_j)^2 + (y - y_j)^2 + c_j^2} - \sqrt{(x - x_i)^2 + (y - y_i)^2 + c_i^2}$.

It is easy to see that $f(x, y)$ in equation (4) constitutes N-dimensional linear vector space $\mathbf{V}^N$ with respect to the operation of addition and multiplication. From the concept of linear independence, the bases of a vector space can be considered as linearly independent subset that spans the entire space. In the space $\mathbf{V}^N$, one set of base vectors is $g_i(x, y) = 1$, and $g_j(x, y), j = 1, 2, \cdots, N, j \neq i$.

From the property of a linear vector space, if all the base functions satisfy the linear equation (2) and (3), so does any function in the space $\mathbf{V}^N$ represented by equation (4). From equation (4), while all the base functions are given, the function $f(x, y)$ is still unknown since the coefficients Λ_i are unknown. However, when all the base functions satisfy equation (2) and (3), we can guarantee that the solution of a partial differential equation approximated by the radial basis function satisfies equation (2) and (3). Thus, when the weighting coefficients of DQ approximation are determined by all the base functions, they can be used to discretize the derivatives in a spatial differential equation. That is the essence of the RBF-DQ method.

Substituting all the base functions into (2) as an example, we can obtain

$$0 = \sum_{k=1}^{N} w_{i,k}^{(m)} \tag{5}$$

$$\frac{\partial^m g_j(x_i, y_i)}{\partial x^m} = \sum_{k=1}^{N} w_{i,k}^{(m)} g_j(x_k, y_k), j = 1, 2, \cdots, N, j \neq i \tag{6}$$

For the given i, equation systems (5) and (6) have N unknowns with N equations. SO, solving this equation system can obtain the weighting coefficients $w_{i,k}^{(m)}$. Using these weighting coefficients, we can discretize the spatial derivatives, and transform the governing equations into a system of algebraic equations, which can be solved by iterative or direct method.

1.2.2 | Local RBF-DQ Method

In global RBF-DQ method, the function approximation form (4) uses all the knots in the computational domain. When the number of knots, N, is large, the matrix formed by base vectors $g_i(x, y) = 1$, and $g_j(x, y), j = 1, 2, \cdots, N, j \neq i$ may be ill-conditioned. To improve it, we can construct a local support region as shown in Fig. 1. In the local RBF-DQ method, at any knot, there is a supporting region, in which there are N knots randomly

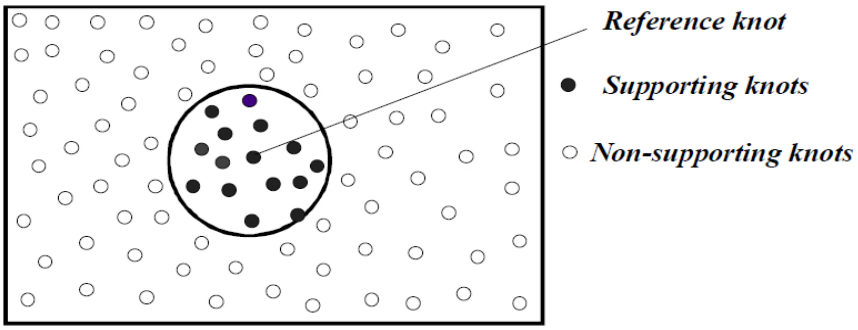

FIGURE 1 Supporting knots around a reference knot.

distributed. So equation (4) is applied in the local support. That is the only difference between the local RBF-DQ method and the global RBF-DQ method. All the related formulations are the same for these two versions of RBF-DQ method.

Since the konts are randomly distributed, the scale of supporting region for each reference knot could be different, and the optimal shape parameter c for accurate numerical results may also be different. This difficulty can be removed from the normalization of scale in the supporting region. The coordinate transformation has the form

$$\bar{x} = \frac{x}{D_i}, \quad \bar{y} = \frac{y}{D_i} \tag{7}$$

where (x, y) represents the coordinates of supporting region in the physical space, $(\bar{x}, \bar{y})$ denotes the coordinates in the unit square, D_i is the diameter of the minimal circle enclosing all knots in the supporting region for the knot i. The corresponding MQ test functions in the local support now become

$$\phi = \sqrt{(\bar{x} - \frac{x_i}{D_i})^2 + (\bar{y} - \frac{y_i}{D_i})^2 + \bar{c}^2}, \quad i = 1, 2, \cdots, N \tag{8}$$

The coordinate transformation also changes the formulation of the weighting coefficients in the local MQ-DQ approximation. For example, by using the differential chain rule, the first order partial derivative with respect to x can be written as

$$\frac{df}{dx} = \frac{df}{d\bar{x}}\frac{d\bar{x}}{dx} = \frac{1}{D_i}\sum_{j=1}^{N} w_j^{(1x)} f_j = \sum_{j=1}^{N} \frac{w_j^{(1x)}}{D_i} f_i \tag{9}$$

where $w_j^{(1x)}$ are the weighting coefficients computed in the unit square, $\frac{w_j^{(1x)}}{D_i}$ are the actual weighting coefficients in the physical domain. In our application, $\bar{c}$ is chosen as a constant.

2 | APPLICATION OF LOCAL RBF-DQ METHOD

2.1 | Description of The Problem

Considering the following nonlinear equation,

$$\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} + u(\frac{\partial u}{\partial x} + \frac{\partial u}{\partial x}) - 2(x + y)u = 4 \tag{10}$$

The computational domain is shown in Fig. 2.

The boundary conditions for the problem are

$$u(x, y) = y^2, \text{ at } x = 0 \tag{11}$$

$$u(x, y) = 1 + y^2, \text{ at } x = 1 \tag{12}$$

$$u(x, y) = x^2 + [y_b(x)]^2, \text{ at } y = y_b(x) \tag{13}$$

$$u(x, y) = 1 + x^2, \text{ at } y = y_t(x) \tag{14}$$

In this study, the random model included in python is used to randomly generate different number of knots in the computational domain. Fifteen different sizes of knots in the internal of computational domain are used, and they are 500, 600, 700, 800, 850, 900, 950, 1000, 1050, 1100, 1150,

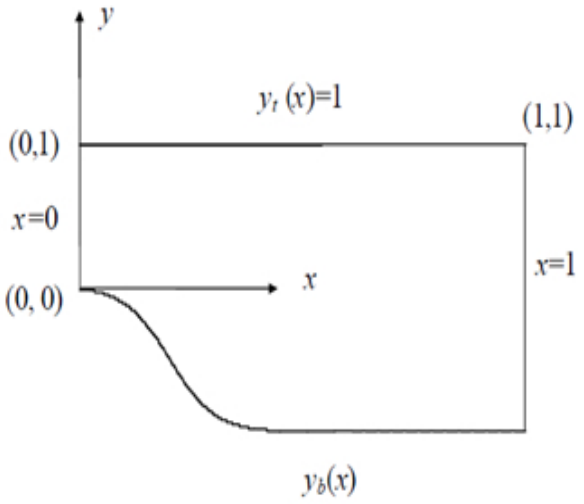

FIGURE 2 Computational domain for the sample problem.

1200, 1300, 1400, 1500. Fifteen different support sizes (number of supporting knots) are used for discretization, and they are 7, 9, 11, 13, 14, 15, 16, 17, 18, 19, 20, 22, 24, 26, 28. The initial value of u of every knot is set as 0.5. The discretized form of contrl equation is

$$\sum_{k=1}^{n} w_{i,k}^{(2x)} u_k + \sum_{k=1}^{n} w_{i,k}^{(2y)} u_k + u_i\left(\sum_{k=1}^{n} w_{i,k}^{(1x)} u_k + \sum_{k=1}^{n} w_{i,k}^{(1y)} u_k\right) - 2(x_i + y_i)u_i = 4 \tag{15}$$

where n is the number of supporting knots in local supporting region. u_k is the value of u on kth knot.

2.2 | Numerical Solution of Nonlinear Systems of Equations

In this study, we use Broyden's method to solve systems of nonlinear equations (15). The method requires only n scalar functinal evaluations per interation and also reduces the number of arithmetic calculations to $O(n^2)$. It belongs to a class of methods known as least-change secant updates that produce algorithms called quasi-Newton. The convergence speed of this method is superlinear.

The calculation steps of Broyden algorithm are listed as follow:

INPUT number n of equations and unknowns; initial approximation $\mathbf{x} = (x_1, \cdots, x_n)^t$; tolerance TOL; maximum number of iterations N.

OUTPUT approximate solution $\mathbf{x} = (x_1, \cdots, x_n)^t$ or a message that the number of iterations was exceeded.

Step1 Set $A_0 = J(\mathbf{x})$ where $J(\mathbf{x})_{i,j} = \frac{\partial f_i}{\partial x_j}(\mathbf{x})$ for $1 \le i, j \le n$; $\mathbf{v} = \mathbf{F}(\mathbf{x})$. (Note: $\mathbf{v} = \mathbf{F}(\mathbf{x}^{(0)})$.)

Step2 Set $A = A_0^{-1}$. (Use Gaussian elimination)

Step3 Set $\mathbf{s} = -\mathbf{A}\mathbf{v}$; (Note: $\mathbf{s} = s_1$) $\mathbf{x} = \mathbf{x} + \mathbf{s}$; k=1.

Step4 While $(k \le N)$ do Steps 5-13.

Step5 Set $\mathbf{w} = \mathbf{v}$; (Svae $\mathbf{v}$.) $\mathbf{V} = \mathbf{F}(\mathbf{x})$; (Note: $\mathbf{v} = \mathbf{F}(\mathbf{x}^{(k)})$.) $\mathbf{y} = \mathbf{v} - \mathbf{w}$. (Note: $\mathbf{y} = \mathbf{y}_k$.)

Step6 Set $\mathbf{z} = -A\mathbf{y}$. (Note: $\mathbf{z} = -A_{k-1}^{-1}\mathbf{y}_k$.)

Step7 Set $p = -\mathbf{s}^t\mathbf{z}$. (Note: $p = \mathbf{s}_k^t A_{k-1}^{-1}\mathbf{y}_k$.)

Step8 Set $\mathbf{u}^t = \mathbf{s}^t A$.

Step9 Set $A = A + \frac{1}{p}(\mathbf{s} + \mathbf{z})\mathbf{u}^t$. (Note: $A = A_k^{-1}$.)

Step10 Set $\mathbf{s} = -A\mathbf{v}$. (Note: $\mathbf{s} = -A_k^{-1}\mathbf{F}(\mathbf{x}^{(k)})$.)

Step11 Set $\mathbf{x} = \mathbf{x} + \mathbf{s}$. (Note: $\mathbf{x} = \mathbf{x}^{(k+1)}$.)

Step12 If $||\mathbf{s}|| \leq$ TOL then OUTPUT(**x**); (The Procedure was Successful.) STOP.

Step13 Set $k = k + 1$.

Step14 OUTPUT("Maximum number of iterations exceeded"); (The Procedure was Unsuccessful.) STOP.

3 | NUMERICAL RESULTS

For all cases, the tolerance for numeral solution of nonlinear system of equations is set as 10^{-8}.

3.1 | Effect of Increasing the Number of Points on the numerical solution

This effect is studied by considering the solution at the point $x = 0.5, y = 0.5$, as shown in TABLE. 1. Due to all konts are randomly generated, the numerical solution at $x = 0.5, y = 0.5$ is calculated by using RBF interpolation scheme. For all cases, the value of shape parameter is 4.

TABLE 1 Numerical solution at $x = 0.5, y = 0.5$ for different number of points in the whole domain.

total number	numerical solution	relative error
1096	0.496240	-0.00752
1146	0.532273	0.06455
1246	0.500198	0.00040
1346	0.462581	-0.07484

The analytical solution at $x = 0.5, y = 0.5$ is 0.5.
Formulation for calculating relative error:

$$\frac{u_{numerical} - u_{analytical}}{u_{analytical}} \tag{16}$$

The total number of points could affect the distribution of supporting knots, which is a key factor for selecting a g̈oodv̈alue of shape parameter so that the numerical solution of partial differential equations (PDEs) can achieve satisfactory accuracy. From the results in TABEL. 1, we can also guess that there may exist a best total number of points to minimize relative error.

TABLE 2 Comparison of numerical accuracy for different total number of points in the whole domain.

total number	L_2 norm of relative error
1096	0.00034
1146	0.00913
1246	0.00398
1346	0.02265

Formulation for calculating L_2 norm of relative error:

$$\sqrt{\frac{\sum_{i=1}^{N} \left(\frac{u_{numerical} - u_{analytical}}{u_{analytical} + 10^{-8}}\right)^2}{N}} \tag{17}$$

The contours of numerical solution in the whole domain with different total number of points are shown in Figure. 3. The convergence of history of the L_2 norm of residuals with different total number of points are shown in Figure. 4. The curves of solution along the vertical line of $x = 0.5$ and the horizontal line of $y = 0.5$ are shown in Figure. 5.

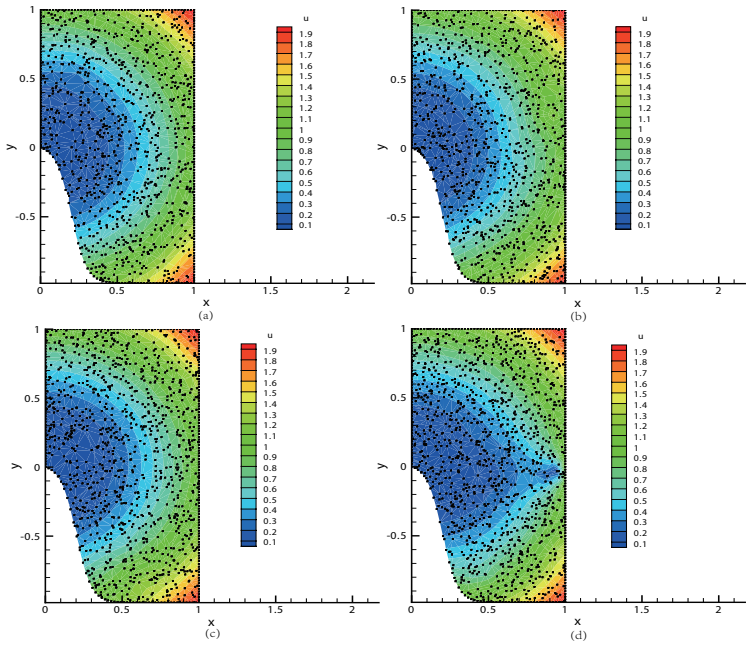

FIGURE 3 The contours of numerical solution in the whole domain with different total number of points. (a) 1096; (b) 1146; (c) 1246; (d) 1346.

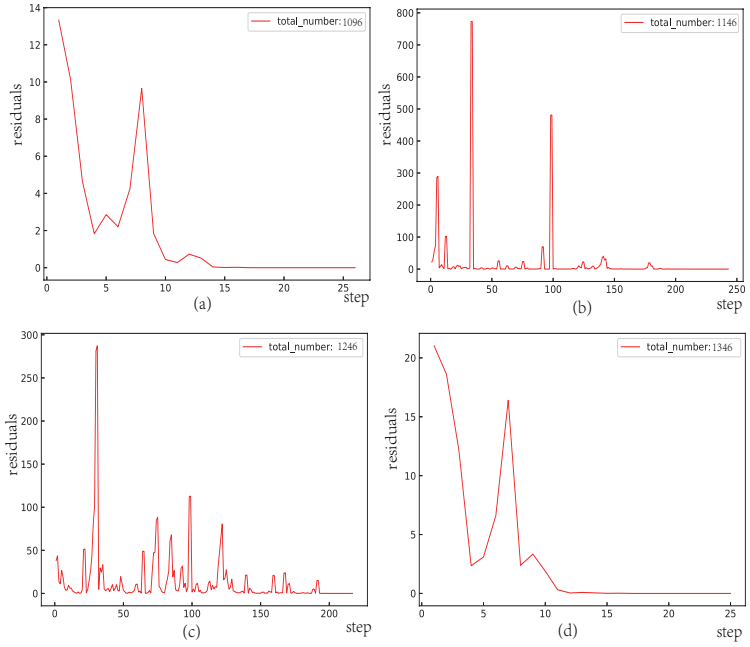

FIGURE 4 The convergence of history of the L_2 norm of residuals with different total number of points. (a) 1096; (b) 1146; (c) 1246; (d) 1346.

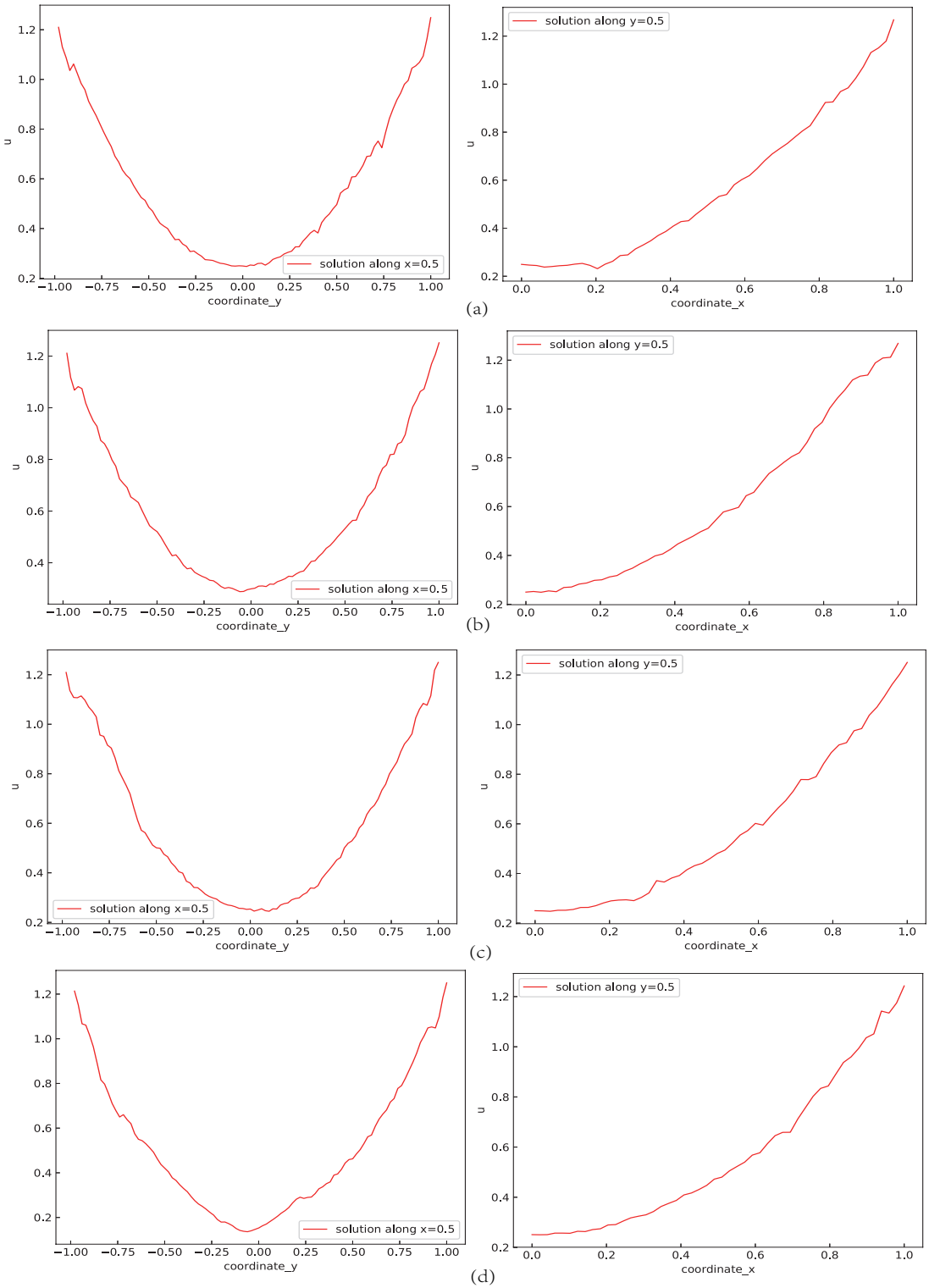

FIGURE 5 The solution along the vertical line of x = 0.5 and the horizontal line of y = 0.5 with different total number of points. (a) 1096; (b) 1146; (c) 1246; (d) 1346.

3.2 | Effect of Increasing the Number of Supporting Points on the numerical solution

This effect is studied by considering the solution at the point $x = 0.5, y = 0.5$, as shown in TABLE. 3. As mentioned previously, the numerical solution at $x = 0.5, y = 0.5$ is calculated by using RBF interpolation scheme. For all cases, value of shape parameter is 1, total number of points in the whole domain is 1246.

TABLE 3 Numerical solution at $x = 0.5, y = 0.5$ for different number of supporting points.

number of supporting knots	numerical solution	relative error
17	0.501875	0.00375
20	0.497096	-0.00581
23	0.498203	-0.00359
25	0.499474	-0.00105

The analytical solution at $x = 0.5, y = 0.5$ is 0.5.
Formulation for calculating relative error is Equation (16).

TABLE 4 Comparison of numerical accuracy for different number of supporting points.

number of supporting knots	L_2 norm of relative error
17	0.00004
20	0.00006
23	0.00005
25	0.000003

Formulation for calculating L_2 norm of relative error is Equation (17).

The contours of numerical solution in the whole domain with different number of supporting points are shown in Figure. 5. The convergence of history of the L_2 norm of residuals with different number of supporting points are shown in Figure. 6.

From the results above, we may guess that the numerical accuracy can be improved by increasing the number of supporting knots.

4 | DISCUSSION

From the results, we can condlude that the numerical accuracy can be improved by increasing the number of total points in the whole domain, or by increasing the number of supporting knots. Considering these two factors can affect the optimal shape parameter, however, there may exist optimal number of total points and optimal number of supporting knots, which remains to be studied in the future. Further more, although Broydeńs algorithm can reduce the amount of computation, the superlinear convergence is too slow in some applications. Finally, we use python language to do this research. Compared with C++, however, the calculation speed of python is significantly slower, implying that we should consider to use some database, like Cpython, to accelerate it.

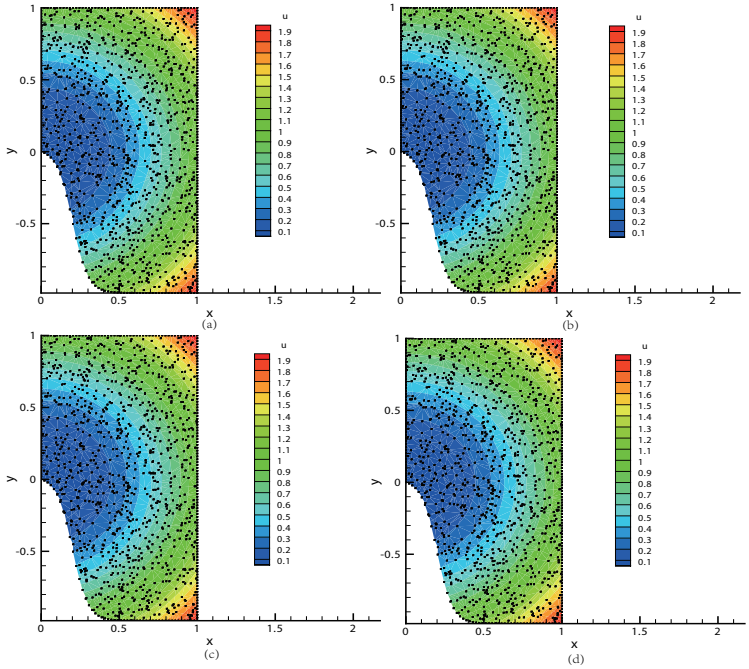

FIGURE 6 The contours of numerical solution in the whole domain with different number of supporting points. (a) 17; (b) 20; (c) 23; (d) 25.

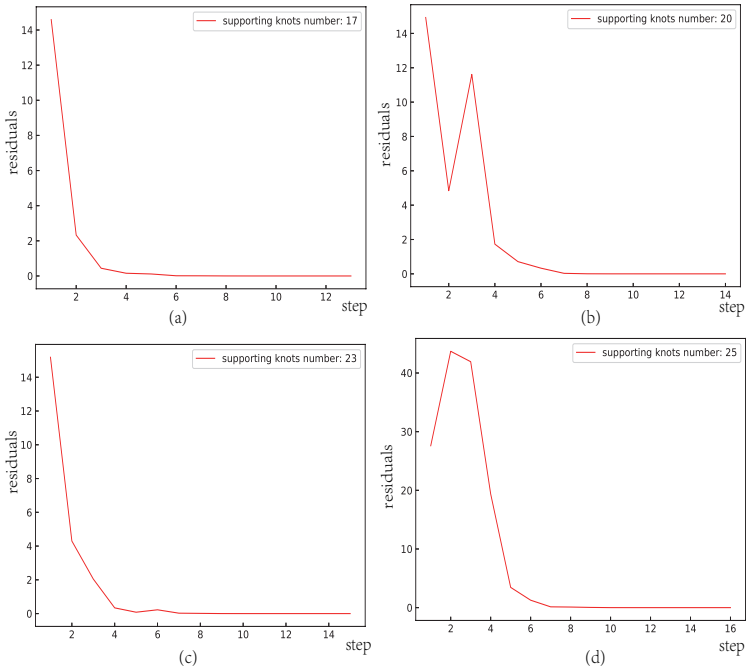

FIGURE 7 The convergence of history of the L_2 norm of residuals with different number of supporting points. (a) 17; (b) 20; (c) 23; (d) 25.

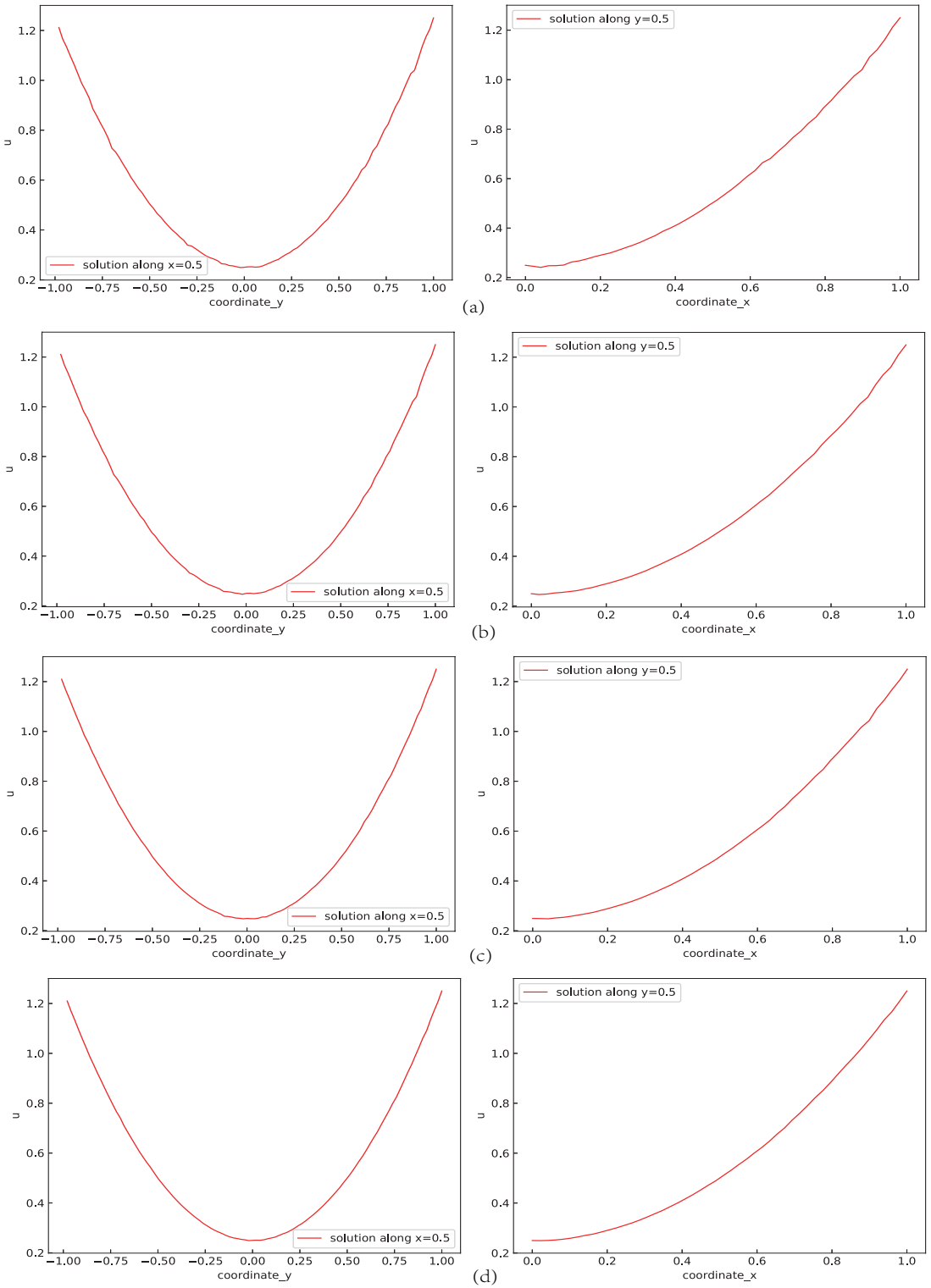

FIGURE 8 The solution along the vertical line of $x = 0.5$ and the horizontal line of $y = 0.5$ with different number of supporting points. (a) 17; (b) 20; (c) 23; (d) 25.

References

1. Bellman Richard, Kashef BG, Casti J. Differential quadrature: a technique for the rapid solution of nonlinear partial differential equations. *Journal of computational physics.* 1972;10(1):40-52.
2. Burden Richard L, Faires J Douglas. Numerical analysis. 2001. *Brooks/Cole, USA.* 2001;.
3. Franke Richard. Scattered data interpolation: tests of some methods. *Mathematics of computation.* 1982;38(157):181-200.
4. Hardy Rolland L. Multiquadric equations of topography and other irregular surfaces. *Journal of geophysical research.* 1971;76(8):1905-1915.
5. Kansa Edward J. Multiquadrics—A scattered data approximation scheme with applications to computational fluid-dynamics—I surface approximations and partial derivative estimates. *Computers and Mathematics with applications.* 1990;19(8-9):127-145.
6. Kansa Edward J. Multiquadrics—A scattered data approximation scheme with applications to computational fluid-dynamics—II solutions to parabolic, hyperbolic and elliptic partial differential equations. *Computers and mathematics with applications.* 1990;19(8-9):147-161.
7. Shu Chang. *Differential quadrature and its application in engineering*. Springer Science and Business Media; 2012.
8. Shu C, Chew YT. Fourier expansion-based differential quadrature and its application to Helmholtz eigenvalue problems. *Communications in Numerical Methods in Engineering.* 1997;13(8):643-653.
9. Shu C, Ding H, Yeo KS. Local radial basis function-based differential quadrature method and its application to solve two-dimensional incompressible Navier–Stokes equations. *Computer Methods in Applied Mechanics and Engineering.* 2003;192(7-8):941-954.
10. Shu Chang, Richards Bryan E. Application of generalized differential quadrature to solve two-dimensional incompressible Navier-Stokes equations. *International Journal for Numerical Methods in Fluids.* 1992;15(7):791-798.