INTERNATIONAL JOURNAL FOR NUMERICAL METHODS IN FLUIDS
Int. J. Numer. Meth. Fluids 2007; **55**:367–386
Published online 14 March 2007 in Wiley InterScience (www.interscience.wiley.com). DOI: 10.1002/fld.1467

Development of a local MQ-DQ-based stencil adaptive method and its application to solve incompressible Navier–Stokes equations

C. Shu[1,*,†], Y. Y. Shan[1] and N. Qin[2]

[1]*Department of Mechanical Engineering, National University of Singapore, 10 Kent Ridge Crescent, Singapore 117576, Singapore*
[2]*Department of Mechanical Engineering, University of Sheffield, Mappin Street, Sheffield S1 3JD, U.K.*

SUMMARY

In this paper, a local stencil adaptive method is presented, which is designed for solving computational fluid dynamics (CFD) problems with *curved boundaries* accurately. A local multiquadric-differential quadrature (MQ-DQ) method is used to discretize the governing equations, taking advantage of its meshless nature. The present method bears the properties of both local MQ-DQ method and local stencil adaptive method and is thus named the local MQ-DQ-based stencil adaptive method. Two test problems with *curved boundaries* are solved to investigate the performance of this solution-adaptive method. The numerical results indicate that the proposed method is effective and efficient by combining the advantages of meshless property for complex geometries and local adaptation for accuracy improvement. Copyright © 2007 John Wiley & Sons, Ltd.

Received 9 September 2006; Revised 24 January 2007; Accepted 27 January 2007

KEY WORDS: local stencil adaptive method; local MQ-DQ method; complex geometry

1. INTRODUCTION

It is well known that the accuracy and resolution of numerical results are strongly dependent on the grid generated in the flow field. Therefore, how to generate a grid of high quality is of paramount importance in the numerical simulation of fluid flow problems. For well-understood physical problems, a non-uniform mesh can be designed to reflect the resolution requirement of the given problems. For example, for the boundary layer problems, fine resolution is typically required for regions near the wall boundaries. For more complicated problems, such as flows with complicated domains or evolving interfacial flows, however, we do not know the properties of their solutions beforehand or the changes of the solutions with time. Thus, we cannot generate

*Correspondence to: C. Shu, Department of Mechanical Engineering, National University of Singapore, 10 Kent Ridge Crescent, Singapore 117576, Singapore.
†E-mail: mpeshuc@nus.edu.sg

an appropriate fixed grid *a priori*. For these problems, the so-called adaptive mesh refinement (AMR) techniques are usually used to generate grids which can capture changes of the flow field accurately.

Considerable attention has been devoted in the past few decades to the development of adaptive refinement procedures and a large number of literature were published. There are two basic approaches of grid adaptation: one is the so-called moving mesh methods and the other is the mesh refinement methods. Moving mesh methods generally use a fixed number of mesh points, and the points are continuously relocated as time evolves so that, at any instant in time, the spatial density of the mesh points is proportional in some sense to the solution variation or the local solution errors estimated. Several methods have been developed to determine mesh movement. The moving finite-element method of Miller and Miller [1], for example, determines the mesh movement by minimizing the residuals of the governing partial differential equations (PDEs). A review of solution-adaptive methods of the moving grid approach has been reported by Eiseman [2]. Further applications of the moving mesh methods can be found in [3–5]. Compared with moving mesh methods, mesh refinement methods are more popular in solving practical problems. In mesh refinement methods, the number of mesh points used is not fixed. During the computational procedure, additional points will be inserted into regions where solutions are varying rapidly and points are deleted from regions where solutions are less active. A lot of work has been done on the development of local mesh refinement methods, see, for example, [6–9], with their applications reported in [10–12].

Recently, Ding and Shu [9] proposed an efficient solution-adaptive procedure for two-dimensional viscous incompressible flows. The key idea of their adaptive stencil algorithm is to build up an adaptive hierarchy of symmetric 5-point stencils in the domain, so that central differencing can be constructed at each interior node. The method combines the advantages of the central difference method and an adaptive mesh method in a very smart way and is therefore very effective and efficient in solving viscous incompressible flows. Despite its advantages, however, the method as proposed cannot be used to solve problems with *curved boundaries* directly due to the use of the central difference method. For problems with *curved boundaries*, before using the adaptive method, we have to implement the so-called coordinate transformation technique to map the physical domain into the computational domain. Furthermore, the original governing equations in the physical domain should also be transformed to the forms in the computational domain. The transformation procedure is usually trivial and the transformed governing equations can be very complicated. In addition, further numerical errors will be brought into the discretization process. Hence, the accuracy of the solutions will be degraded when compared with those obtained directly in the physical domain. To avoid the above drawbacks, in this paper, we will develop a local stencil adaptive method which can be directly applied to solve problems with *curved boundaries* in the physical domain. How to effectively discretize the governing equations in complex physical domains is the key point for this method. The so-called meshless methods seem to be a very promising choice. The term meshless refers to the ability of the methods to construct functional approximation or interpolation entirely from information at a set of scattered nodes, among which there is no pre-specified connectivity or relationship. A number of meshless methods have been proposed in the literature, such as the smoothed particle hydrodynamics method [13], the element-free Galerkin method [14], the reproducing kernel particle method [15], the partition of unity method [16], the hp-clouds method [17], and the meshless local Petrov–Galerkin method [18]. Recently, the so-called local radial basis function-based differential quadrature (RBF-DQ) method has been proposed by Shu *et al.* [19]. The method is a natural mesh-free approach. It can be regarded as a combination of the conventional DQ method with the RBFs by means of taking RBFs as the trial functions in the DQ scheme. As a result, it combines the mesh-free nature of RBFs approximation with the derivative approximation of DQ method. In the present paper, the local multiquadric-DQ (MQ-DQ) method is chosen to discretize the governing equations, which is combined with the stencil adaptive method for accurate solution *for problems with curved boundaries*.

2. DEVELOPMENT OF A LOCAL MQ-DQ-BASED STENCIL ADAPTIVE METHOD

In this section, we will present in detail the development of a local MQ-DQ-based stencil adaptive method. To demonstrate the procedure clearly, we will describe the method in a step-by-step manner.

2.1. Local MQ-DQ method

The details of the local MQ-DQ method can be found in [19]. Here, only a brief description is given to illustrate the idea and formulation of the local MQ-DQ method. If a function $f(x, y)$ is assumed to be sufficiently smooth, its *n*th order derivative with respect to x, and *m*th order derivative with respect to y, at a point (x_i, y_i) can be approximated by the local MQ-DQ method as

$$f_x^{(n)}(x_i, y_i) = \sum_{k=1}^{N} w_{i,k}^{(n)} f(x_{ik}, y_{ik}) \tag{1}$$

$$f_y^{(m)}(x_i, y_i) = \sum_{k=1}^{N} \overline{w}_{i,k}^{(m)} f(x_{ik}, y_{ik}) \tag{2}$$

where N represents the number of points used in derivative approximation. The subscript i is a global index which refers to the reference point and k is a local index which stands for the *k*th supporting point for point i (its global index can be written as ik). In our code, the *N*th supporting point for point i is the reference point itself. $w_{i,k}^{(n)}$, $\overline{w}_{i,k}^{(m)}$ are the related weighting coefficients in the x and y directions, which need to be determined. In the local MQ-DQ method, the determination of weighting coefficients is based on the analysis of a local MQ approximation in a linear vector space.

In the local MQ-DQ method, at any given reference point (x_i, y_i), there is a supporting region, in which there are N points randomly distributed. The function in this region can be locally approximated by MQ-RBFs as

$$f(x, y) = \sum_{j=1}^{N-1} \lambda_j g_j(x, y) + \lambda_N \tag{3}$$

where

$$g_j(x, y) = \sqrt{(x - x_j)^2 + (y - y_j)^2 + c^2} - \sqrt{(x - x_N)^2 + (y - y_N)^2 + c^2} \tag{4}$$

c is a shape parameter given by the user. The subscript j is a local index which refers to the *j*th supporting point for the reference point (x_i, y_i) and the subscript N stands for the reference point itself.

It is easy to see that $f(x, y)$ in Equation (3) constitutes an 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 $g_N(x, y) = 1$ and $g_j(x, y)$, $j = 1, \ldots, N-1$, given by Equation (4).

From the property of a linear vector space, if all the base functions satisfy the linear equation (1) or (2), so does any function in the space $\mathbf{V}^N$ represented by Equation (3). There is an interesting feature. From Equation (3), when 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 (1) or (2), we can guarantee that $f(x, y)$ also satisfies Equation (1) or (2). In other words, we can guarantee that the solution of a PDE which can be represented by Equation (3) satisfies Equation (1) or (2). 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 PDE. This is the essence of the local MQ-DQ method.

Substituting all the base functions into Equations (1) and (2), we can obtain a set of linear equations, which can be expressed in the matrix form as

$$[D] = [G][W] \tag{5}$$

where

$$[D] = \begin{bmatrix} \dfrac{\partial^n g_1(x_i, y_i)}{\partial x^n} & \dfrac{\partial^m g_1(x_i, y_i)}{\partial y^m} \\ \vdots & \vdots \\ \dfrac{\partial^n g_{N-1}(x_i, y_i)}{\partial x^n} & \dfrac{\partial^m g_{N-1}(x_i, y_i)}{\partial y^m} \\ 0 & 0 \end{bmatrix}$$

$$[G] = \begin{bmatrix} g_1(x_1, y_1) & g_1(x_2, y_2) & \cdots & g_1(x_N, y_N) \\ \vdots & \vdots & \cdots & \vdots \\ g_{N-1}(x_1, y_1) & g_{N-1}(x_2, y_2) & \ddots & g_{N-1}(x_N, y_N) \\ 1 & 1 & \cdots & 1 \end{bmatrix}$$

$$[W] = \begin{bmatrix} w_{i1}^{(n)} & \overline{w}_{i1}^{(m)} \\ w_{i2}^{(n)} & \overline{w}_{i2}^{(m)} \\ \vdots & \vdots \\ w_{iN}^{(n)} & \overline{w}_{iN}^{(m)} \end{bmatrix}$$

For matrix $[D]$, we can successively differentiate Equation (4) to get its elements. For example, the first-order derivative of $g_j(x, y)$ with respect to x can be written as

$$\frac{\partial g_j(x,y)}{\partial x} = \frac{x - x_j}{\sqrt{(x-x_j)^2 + (y-y_j)^2 + c^2}} - \frac{x - x_N}{\sqrt{(x-x_N)^2 + (y-y_N)^2 + c^2}} \tag{6}$$

With the known matrices $[D]$ and $[G]$, the weighting coefficient matrix $[W]$ can be obtained by using a direct method such as LU decomposition.

In the local MQ-DQ method, the shape parameter c has a strong influence on the accuracy of numerical results. The optimal value of c is affected by the number of supporting points and the size of supporting region. Usually, the number of supporting points is fixed for an application. The effect of supporting region on c can be removed by normalizing the scale of the supporting domain. The essence of this idea is to transform the local support region to a unit square for the two-dimensional case. The normalization can be carried out by the following transformation:

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

where (x, y) represents the coordinates of supporting region in the physical space, $(\overline{x}, \overline{y})$ denotes the coordinates in the unit square, D_i is the side length of the minimal square enclosing all nodes in the supporting region for the point i. The corresponding MQ basis functions in the local support now become

$$\varphi = \sqrt{\left(\overline{x} - \frac{x_i}{D_i}\right)^2 + \left(\overline{y} - \frac{y_i}{D_i}\right)^2 + \overline{c}^2}, \quad i = 1, \ldots, N \tag{8}$$

Compared with the traditional MQ-RBF method, we can find that the shape parameter c is actually equivalent to $\overline{c}D_i$. The coordinate transformation (7) 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

$$\frac{\partial f}{\partial x} = \frac{\partial f}{\partial \overline{x}}\frac{\mathrm{d}\overline{x}}{\mathrm{d}x} = \frac{1}{D_i}\frac{\partial f}{\partial \overline{x}} = \frac{1}{D_i}\sum_{j=1}^{N} w_{i,j}^{(1)} f_{ij} = \sum_{j=1}^{N} \frac{w_{i,j}^{(1)}}{D_i} f_{ij} \tag{9}$$

where $w_{i,j}^{(1)}$ are the weighting coefficients computed in the unit square, $w_{i,j}^{(1)}/D_i$ are the actual weighting coefficients in the physical domain. Clearly, when D_i is changed, the equivalent c in the physical space is automatically changed. The choice of constant $\overline{c}$ can be referred to the work of Shu *et al.* [19], which shows an efficient way to select $\overline{c}$. For the present work, nine local nodes are used in MQ-DQ discretization. It was found that for this case, the shape parameter $\overline{c}$ can be chosen in a wide range to get a converged solution, in which the accuracy of numerical solution does not change much. This means that the results are not very sensitive to $\overline{c}$. In our numerical examples, $\overline{c}$ is taken as 4.

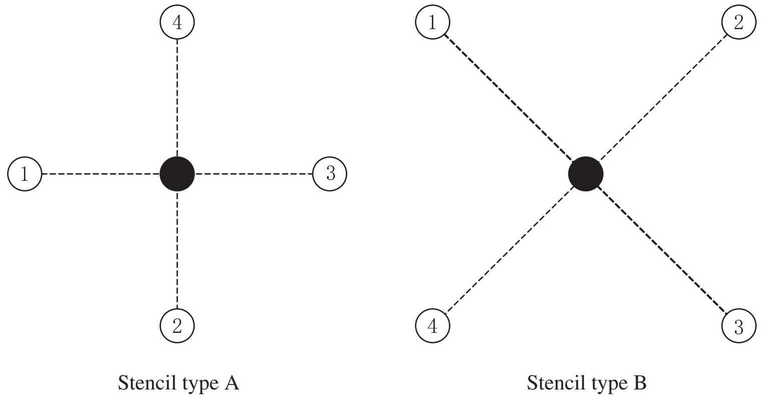

Figure 1. Configuration of two types of stencils.

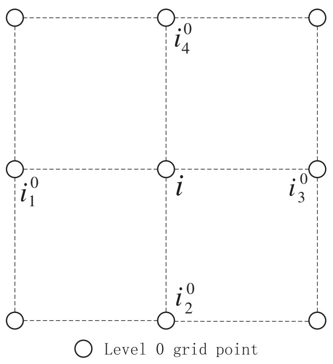

Figure 2. Configuration of an initial stencil.

2.2. Finite difference-based stencil adaptive algorithm

The details of the finite difference-based stencil adaptive algorithm can be found in [9]. This algorithm is based on the local stencil refinement and coarsening. For any interior point in the domain, there is a local stencil associated with it. In general, there are two types of stencils encountered in this adaptive algorithm, as shown in Figure 1. For the convenience of inserting and deleting nodes from the adaptive stencils, only one index is used to identify the node in the domain, i.e. a global nodal index. For an arbitrary reference node i, its stencil can be symbolized as i_n^m and the position of the nodes in the stencil are denoted by X_n^m, where the superscript m denotes the resolution level, and the subscript $n = 0, 1, \ldots, 4$ denotes the local index of the member nodes in the stencil.

The stencil adaptation procedure is shown below. Initially, it is easy to construct the stencil of 'A' configuration around an arbitrary interior node i, as shown in Figure 2. For the convenience of illustration, we will take node i and its attached stencil as an example to demonstrate the whole procedure.

2.2.1. Criteria for stencil refinement/coarsening. First, we have to determine whether the stencil should be refined or coarsened. In this procedure, an action indicator is adopted to monitor the variation of the parameter of interest and send the commands to perform the corresponding process. Ding and Shu [9] proposed two monitor parameters to measure the local variation of the solution. They are defined as follows:

1. Absolute difference, which is defined by $\Delta_1 = \max(u_i) - \min(u_i)$.
2. Relative difference, which is defined by $\Delta_2 = \max(u_i) - \min(u_i)/\max|u_i|$ where u is the parameter of interest, and the subscript $i = 0, 1, \ldots, 4$ denotes the local index of nodes in one stencil.

The action indicator is constructed by two thresholds. One is the upper bound $\theta_{\max}$ and the other is the lower bound $\theta_{\min}$. If $\theta_{\min} < \Delta < \theta_{\max}$, the magnitude of local variation with respect to the parameter of interest is in the range given *a priori* and no adaptation procedure will be implemented on the stencil. Otherwise, if $\Delta > \theta_{\max}$, the local variation of the parameter is relatively large and the stencil needs to be refined to improve the accuracy. Similarly, if $\Delta < \theta_{\min}$, the local variation of the parameter is relatively small and the stencil needs to be coarsened to improve the efficiency. To simplify the adaptation procedure, some constraints are introduced and checked before the process of stencil refinement/coarsening, which can be found in [9]. From the above, whether the stencil should be refined or coarsened has been determined. In the following, we will show the stencil refinement and coarsening algorithms.

2.2.2. Stencil refinement algorithm. If stencil refinement at node i is determined by the local resolution requirement, the stencil resolution level of node i needs to be advanced from level 0 (i_n^0) to level 1 (i_n^1). Stencil refinement is achieved by the injection of extra grid points in the old stencil region to form a new stencil. In this case four new nodes are inserted. The positions of the four newly generated nodes are actually the midpoints of the stencil edges, as shown in Figure 3. More specifically, they yield

$$X_1^1 = \frac{X_1^0 + X_4^0}{2}, \quad X_2^1 = \frac{X_4^0 + X_3^0}{2}, \quad X_3^1 = \frac{X_3^0 + X_2^0}{2} \quad \text{and} \quad X_4^1 = \frac{X_2^0 + X_1^0}{2} \tag{10}$$

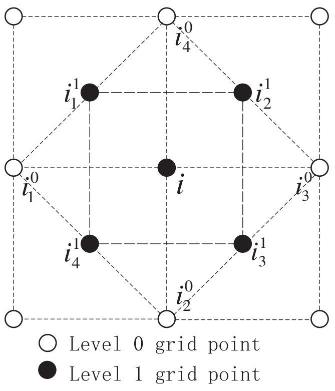

Figure 3. Stencil refinement from resolution level 0 to 1.

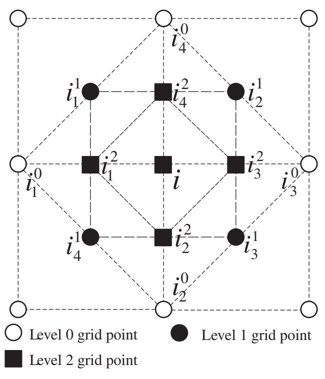

Figure 4. Stencil refinement from resolution level 1 to 2.

It can be clearly seen that after the refinement, the reference node remains at the centre of the stencil. Furthermore, the stencil for node i naturally evolves from the 'A' configuration of i_n^0 into 'B' configuration of i_n^1. It is very interesting to observe the stencils for the newly added nodes. Take the member node i_1^1 of node i as an example. From Figure 3, we can see that the stencil for node i_1^1 falls into type 'B' category, and it also has the same stencil size as the refined stencil of the node i at resolution level 1. In other words, node i_1^1 generated during the level 1 adaptation also possesses the stencil of that resolution level. This is also the case for the other newly inserted points.

If further stencil refinement at node i is required, the stencil resolution level of node i needs to be advanced from level 1 (i_n^1) to level 2 (i_n^2). As a consequence, four more nodes are inserted in the domain. The positions of the four nodes are also the midpoints of the stencil edges, as shown in Figure 4. More specifically, they yield

$$X_1^2 = \frac{X_1^1 + X_4^1}{2}, \quad X_2^2 = \frac{X_4^1 + X_3^1}{2}, \quad X_3^2 = \frac{X_3^1 + X_2^1}{2} \quad \text{and} \quad X_4^2 = \frac{X_2^1 + X_1^1}{2} \tag{11}$$

It can be clearly seen from Figure 4 that the stencil of level 2 has the 'A' configuration. We also observe that the newly added nodes also possess level 2 stencils.

If we follow the same procedure and carry on the adaptation, we can find that the stencil of the even resolution levels has the 'A' configuration while the stencil of the odd resolution levels has the 'B' configuration. As the adaptation continues, the two types of stencil appear alternatively. It should be pointed out that during the stencil refinement, only the stencils of the reference node and its stencil points are affected or changed. The stencils of other nodes in the domain remain the same.

2.2.3. Local stencil coarsening. If a node stencil is required to be coarsened, its stencil will be recovered to the configuration of a stencil at a coarser level. As compared with the stencil refinement, the process of stencil coarsening is much easier since the adaptation information at every interior node has been recorded during previous stencil refinement. Note that the coarsening procedure can only be carried out on resolution levels higher than the initial one, i.e. $m>0$.

2.3. Local MQ-DQ-based stencil adaptive method

Although the finite difference-based stencil adaptive method is very effective and efficient in solving problems with regular domains, it is not applicable to solve problems with *curved boundaries*. Hence, the application of the method is severely limited. Here, we extend the capability of the local stencil adaptive algorithm to solve problems with *curved boundaries*.

For problems with *curved boundaries*, the background mesh will normally be non-uniform. Therefore, the initial stencils for the interior nodes will not be geometrically symmetric, as shown in Figure 5. On the other hand, we can still use the stencil refinement process of [9] to effectively refine the stencil for problems with *curved boundaries*. If stencil refinement at node i is required, similar to the process in the finite difference-based stencil adaptive method, extra four nodes will be injected to form a new stencil. Before injection of each of the four nodes, however, we will first search the four member nodes of the stencil of that point. Fortunately, it is not difficult due to our stencil design. Here, we will take the injection of i_1^1 as an example to illustrate the process, as shown in Figure 6. From Figure 6, we can see that three of the four member nodes of stencil i_1^1 can be immediately determined since they are actually among the old stencil i_n^0 for node i. The only exception is the member node $(i_1^1)_1^1$. We note that it is also located in the stencil for nodes

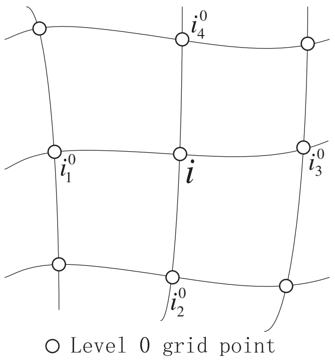

Figure 5. Configuration of an initial stencil in complex geometries.

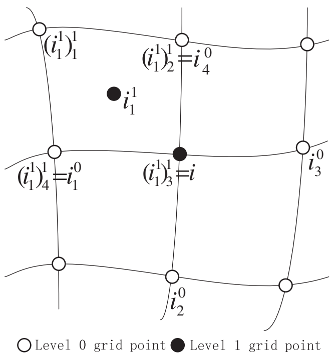

Figure 6. Injection of grid point i_1^1.

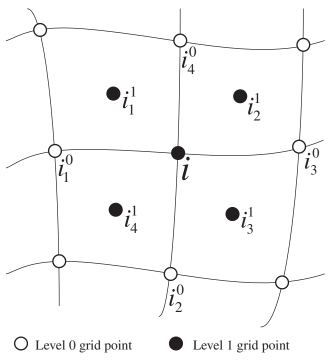

Figure 7. Stencil refinement from resolution level 0 to 1.

i	i_1	i_2	i_3	i_4	i_5	i_6	i_7	i_8

Figure 8. A one-dimensional array for storing supporting points.

i_1^0 and i_4^0 of resolution level 0. Therefore, it can be accessed by either $(i_1^0)_4^0$ or $(i_4^0)_1^0$. After finding the four member nodes, the position of node i_1^1 can be easily determined. It is reasonable to insert the point to the midpoint of the four member nodes. More specifically, it yields

$$X_1^1 = \frac{(X_1^1)_1^1 + (X_1^1)_2^1 + (X_1^1)_3^1 + (X_1^1)_4^1}{4} \tag{12}$$

Now the first grid point i_1^1 has been inserted and its stencil has been determined. The same process applies for the injection of other grid points. When all the four grid points are inserted, the new stencil of resolution 1 for node i has been formed, as shown in Figure 7.

As mentioned above, when we solve problems in rectangular domain, the central difference method can be used to approximate the spatial derivatives because each stencil is a 5-point symmetric stencil. In complex geometries, however, stencils are not geometrically symmetric and the central difference method cannot be directly used without sacrificing the accuracy. Thus, we have to look for another way to approximate the derivatives. In this stencil adaptive algorithm, for a reference node at which derivatives need to be approximated, what we have is only the information about the positions of the member nodes of the stencil. Fortunately, we can use the so-called meshless methods to approximate the derivatives with high accuracy and here the local MQ-DQ method is chosen, which has been described above.

With only four supporting points, the local MQ-DQ method may not approximate derivatives very accurately. To improve the accuracy of the method, more points should be used in derivative approximation. In our work, the number of supporting points in derivative approximation is chosen to be eight, for accuracy comparable or superior to the central difference scheme on a regular Cartesian grid, based on our experience. We will show in detail how to efficiently search for the eight supporting points. For the convenience of illustration, a one-dimensional array, which is shown in Figure 8, is assigned to each point to store the eight supporting points used in the local MQ-DQ approximation.

i	i_1^0	i_2^0	i_3^0	i_4^0	$(i_4^0)_1^0$	$(i_3^0)_4^0$	$(i_2^0)_3^0$	$(i_1^0)_2^0$

Figure 9. Initial eight supporting points for point i.

i	i_1^1	i_2^1	i_3^1	i_4^1	i_1^0	i_2^0	i_3^0	i_4^0

Figure 10. Eight supporting points for point i in resolution level 1.

In the array, the first four supporting points, from i_1 to i_4, are the member points of the stencil for reference point i. Comparatively, the search of the other four supporting points, from i_5 to i_8, is not so easy. Now, we will show our method of searching supporting points through the refinement procedure of an initial grid point i with the help of Figure 7.

Initially, it is convenient for us to search eight supporting points for point i, as shown in Figure 9. When the refinement procedure is implemented on point i, the stencil of this point will be advanced from resolution level 0 to resolution level 1 and four new points will be generated, i.e. from i_1^1 to i_4^1. Consequently, the eight supporting points for point i will be those shown in Figure 10. That is, the four member points of the stencil at resolution level 1 will be used as the first four supporting points and the four member points of the stencil at resolution level 0 will be chosen as the last four supporting points for derivative approximation.

As for the newly generated points, we will take point i_1^1 as an example to illustrate the way of searching its eight supporting points. For this point, the first four supporting points will also be the member points of its attached stencil and they are $(i_4^0)_1^0$, i_4^0, i_0^1 and i_1^0, respectively. For each of the four member points of the stencil for point i_1^1, there is a local stencil associated with it. The last four supporting points for i_1^1 can be chosen from the member points of these four stencils. When refinement procedure is implemented on point i_1^1, the member points of the new stencil will become the first four supporting points and the member points of the old stencil will become the last four supporting points for derivative approximation. As the adaptation continues, the way of searching supporting points will be carried out in the same manner for the refined resolution levels.

3. NUMERICAL EXPERIMENTS

In this section, two examples are presented to examine the performance of the local MQ-DQ-based stencil adaptive method presented in the paper. In the first test case, we investigate the accuracy of numerical solution using the method against an analytical solution for a boundary value problem governed by the Poisson equation. For the second case, we numerically simulate a natural convective heat transfer problem in a concentric annulus between a square outer cylinder and a circular inner cylinder.

3.1. Comparison with analytical solution of the Poisson equation

To illustrate the capability of the method *for problems with curved boundaries*, we take the computational domain to be an annulus between an inner airfoil and an outer circular cylinder.

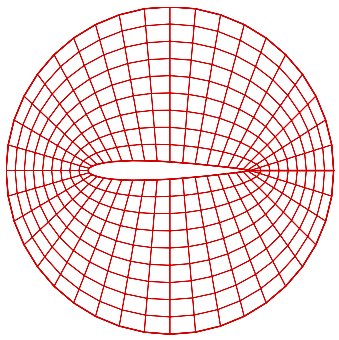

Figure 11. Domain around an airfoil and its background mesh.

The airfoil is chosen to be a modified NACA series airfoil with its profile given by

$$y(x) = \pm 5bt[0.2969\sqrt{x_{\text{int}}x} - 0.126x_{\text{int}}x - 0.3516(x_{\text{int}}x)^2 + 0.2843(x_{\text{int}}x)^3 - 0.1015(x_{\text{int}}x)^4]$$

where $x_{\text{int}} = 1.0089$. The '+' sign is used for the upper surface and the '−' sign is used for the lower surface of the airfoil. b is the chord of the airfoil and t represents the maximum thickness. In our work, we set $b = 1$ and $t = 0.12$. The radius of the outer circular cylinder is chosen to be the same as the chord of the airfoil, i.e. $r = b$. The computational domain and the background mesh are shown in Figure 11.

We study the solution of the Poisson equation with Dirichlet boundary conditions on all boundaries:

$$\frac{\partial^2 T}{\partial x^2} + \frac{\partial^2 T}{\partial y^2} = f(x, y) \tag{13}$$

that is, $T_{\text{B}}(x, y)$ is given at the domain boundaries.

For our numerical experiments, we can easily devise an analytical solution $T = \sin(\pi x)\sin(\pi y)$ for the above problem with a source term

$$f(x, y) = -2\pi^2 \sin(\pi x)\sin(\pi y) \tag{14}$$

and a boundary condition satisfied by $T = \sin(\pi x)\sin(\pi y)$ at the domain boundaries.

To generate a background mesh, we used a body-fitted grid generation technique. The background grid is a 40×10 mesh, as shown in Figure 11. After discretizing Equation (13) on all the interior points by the local MQ-DQ method, we get a set of linear algebraic equations. To solve the resultant equations, the Gauss–Seidel iterative method is used. In the iterative solution process, the dependent variable T at the interior points is initially set to zero. The convergence criterion is set to 10^{-10}, which is considered to be small enough for a converged solution. When the solution is converged, its accuracy is measured by the following L_2 error norm:

$$\sqrt{\frac{1}{S}\sum_{i=1}^{N}(|T_{i,\text{num}} - T_{i,\text{ana}}|^2\, \mathrm{d}S_i)} \tag{15}$$

where $T_{i,\text{num}}$ and $T_{i,\text{ana}}$ are the numerical solution and analytical solution at point i, respectively. $\mathrm{d}S_i$ represents the area surrounding the point i and S is the total area of the domain. For this

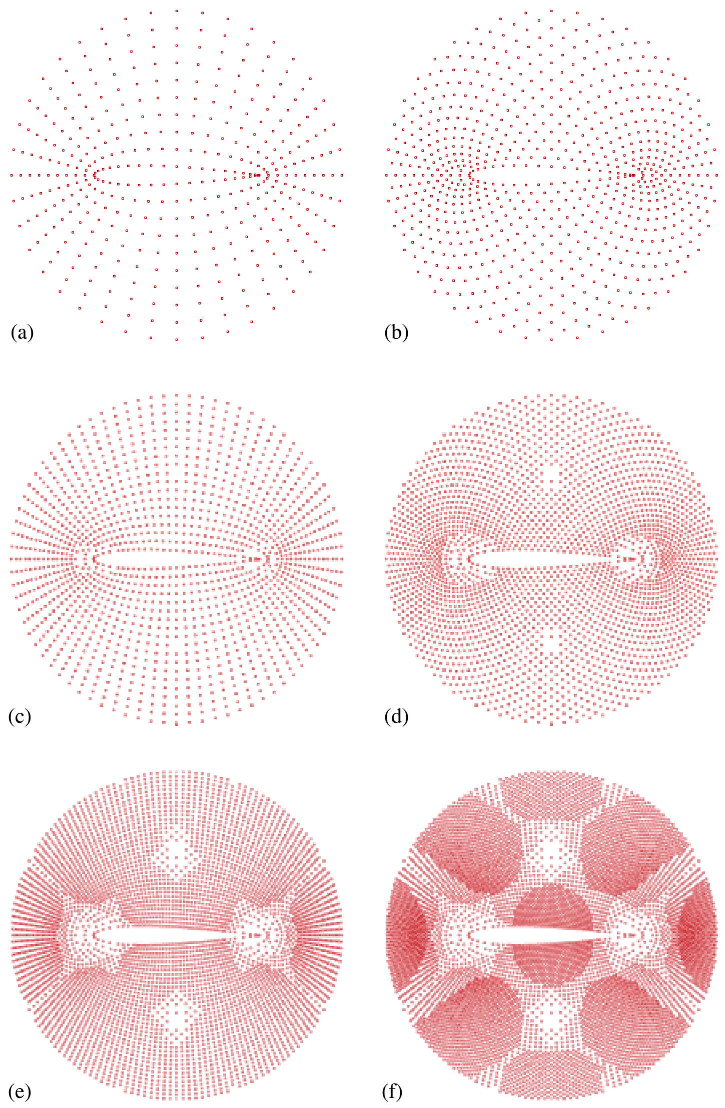

Figure 12. Final node distributions with different highest resolution levels. (a) $\text{Level}_{\max}=0$; (b) $\text{Level}_{\max}=1$; (c) $\text{Level}_{\max}=2$; (d) $\text{Level}_{\max}=3$; (e) $\text{Level}_{\max}=4$; and (f) $\text{Level}_{\max}=5$.

case, six levels of refinement have been used (from level 0 to level 5). The upper and lower bound of the action indicator is set to 0.1 and 0.01. During the computation, the adaptation checking procedure is activated for every 1500 iterations, to determine whether to insert or delete points. After the solution is converged, the node distributions using different resolution levels are shown in Figure 12, and the numerical performance of the stencil adaptive method is quantitatively studied in Table I. It can be observed from Figure 12 that with increasing resolution level, more nodes

Table I. Numerical results of the adaptive Poisson solver.

Maximum resolution level	Grid points	Error_L2	Running time (s)
0	$40\times 10=400$	9.02×10^{-3}	0.6
1	744	4.33×10^{-3}	1.3
2	1428	1.44×10^{-3}	5.2
3	2650	1.09×10^{-3}	17.5
4	4817	6.89×10^{-4}	71.0
Fixed node method	$160\times 40=6400$	6.14×10^{-4}	177.0

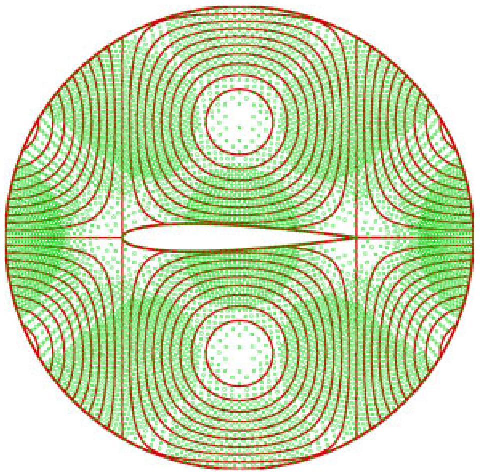

Figure 13. Contour of the variable T versus the node distribution with highest resolution level to 5, i.e. $\text{Level}_{\text{max}}=5$.

are added into the regions where high gradient of the solution is detected and that the node distributions can clearly display the distribution of the solution gradients. At the same time, it can be seen from Table I that increasing the resolution level also improves the accuracy of numerical solution. The level 0 grid achieves a solution with error of 9.02×10^{-3} while the level 4 grid achieves a solution with error of 6.89×10^{-4}. As compared with the fixed grid method (the local MQ-DQ method is also used) on a 160×40 grid, our adaptive algorithm with four resolution levels (4817 points totally) requires much less time to achieve a solution of similar accuracy. The solution of Equation (13) is shown in Figure 13.

3.2. Natural convective heat transfer in a concentric annulus between a square outer cylinder and a circular inner cylinder

Natural convective heat transfer in enclosures is of great importance due to its wide applications in engineering. In this section, we will apply our developed local MQ-DQ-based stencil adaptive

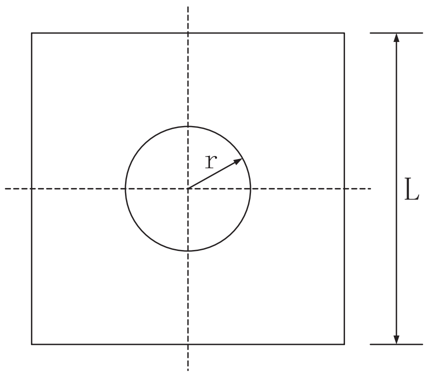

Figure 14. Sketch of physical domain of natural convection between a square outer cylinder and a circular inner cylinder.

method to simulate the natural convective heat transfer in a concentric annulus between a square outer cylinder and a circular inner cylinder.

3.2.1. Governing equations and boundary conditions. A schematic view of a horizontal concentric annulus between a square outer cylinder and a heated circular inner cylinder is shown in Figure 14. Heat is generated uniformly within the circular inner cylinder (constant heat source), which is placed concentrically within the cold square cylinder (constant heat sink). If the cylinders are long enough and the flow reaches its equilibrium state, it can be considered to be steady, laminar and two dimensional. The buoyancy force is the driven force for the flow.

Based on the Boussinesq approximation, the non-dimensional governing equations for the problem are written in the vorticity-stream function formulation as

$$\frac{\partial^2 \psi}{\partial x^2} + \frac{\partial^2 \psi}{\partial y^2} = \omega \tag{16}$$

$$u\frac{\partial \omega}{\partial x} + v\frac{\partial \omega}{\partial y} = Pr\left(\frac{\partial^2 \omega}{\partial x^2} + \frac{\partial^2 \omega}{\partial y^2}\right) - Pr\,Ra\frac{\partial T}{\partial x} \tag{17}$$

$$u\frac{\partial T}{\partial x} + v\frac{\partial T}{\partial y} = \frac{\partial^2 T}{\partial x^2} + \frac{\partial^2 T}{\partial y^2} \tag{18}$$

where ψ denotes the stream function, ω represents the vorticity, and T is the non-dimensional temperature. The Prandtl number is defined as $Pr = \mu C_p/k$, and the Rayleigh number is defined as $Ra = C_p\rho_0 g\beta L^3 \Delta T/kv$. Here, μ is the viscosity, C_p is the specific heat at constant pressure, k is the thermal conductivity, ρ_0 is the reference density, g is the gravitational acceleration, β is the thermal expansion coefficient, L is the side length of the square outer cylinder, ΔT is the temperature difference between the inner and outer cylinders, and υ is the kinematic viscosity.

Velocity components u and v can be computed from the stream function ψ as

$$u = \frac{\partial \psi}{\partial y}, \quad v = -\frac{\partial \psi}{\partial x} \tag{19}$$

Using the expressions in Equation (19), Equation (16) can also be written as

$$\omega = \frac{\partial u}{\partial y} - \frac{\partial v}{\partial x} \tag{20}$$

The governing equations (16)–(19) are discretized by the local MQ-DQ method. The discretization form of the governing equations at a general node i can be written as follows:

$$\sum_{k=1}^{N} w_{i,k}^{(2)} \psi_i^k + \sum_{k=1}^{N} \overline{w}_{i,k}^{(2)} \psi_i^k = \omega_i \tag{21}$$

$$u_i \sum_{k=1}^{N} w_{i,k}^{(1)} \omega_i^k + v_i \sum_{k=1}^{N} \overline{w}_{i,k}^{(1)} \omega_i^k = Pr \left(\sum_{k=1}^{N} w_{i,k}^{(2)} \omega_i^k + \sum_{k=1}^{N} \overline{w}_{i,k}^{(2)} \omega_i^k \right) - Pr\, Ra \sum_{k=1}^{N} w_{i,k}^{(1)} T_i^k \tag{22}$$

$$u_i \sum_{k=1}^{N} w_{i,k}^{(1)} T_i^k + v_i \sum_{k=1}^{N} \overline{w}_{i,k}^{(1)} T_i^k = \sum_{k=1}^{N} w_{i,k}^{(2)} T_i^k + \sum_{k=1}^{N} \overline{w}_{i,k}^{(2)} T_i^k \tag{23}$$

$$u_i = \sum_{k=1}^{N} \overline{w}_{i,k}^{(1)} \psi_i^k, \quad v_i = -\sum_{k=1}^{N} w_{i,k}^{(1)} \psi_i^k \tag{24}$$

where k is the kth supporting point of reference node i. As for the boundary conditions, the non-slip conditions are imposed on the wall, and both cylinders are considered isothermal. From the non-slip condition, the velocities u and v on both the inner and outer cylinder walls are zero. Thus the boundary condition can be written as follows:

$$u|_{\text{inner\_wall}} = u|_{\text{outer\_wall}} = 0, \quad v|_{\text{inner\_wall}} = v|_{\text{outer\_wall}} = 0 \tag{25}$$

$$\psi|_{\text{inner\_wall}} = 0, \quad \psi|_{\text{outer\_wall}} = 0 \tag{26}$$

$$T|_{\text{inner\_wall}} = 1, \quad T|_{\text{outer\_wall}} = 0 \tag{27}$$

The boundary condition for vorticity ω can be derived from Equation (20),

$$\omega|_{\text{wall}} = \left.\frac{\partial u}{\partial y}\right|_{\text{wall}} - \left.\frac{\partial v}{\partial x}\right|_{\text{wall}} \tag{28}$$

Like the governing equations, this boundary condition can also be discretized by the local MQ-DQ method and the vorticity value on the boundary can be updated by the following equation:

$$\omega_i = \sum_{k=1}^{N} \overline{w}_{i,k}^{(1)} u_i^k - \sum_{k=1}^{N} w_{i,k}^{(1)} v_i^k \tag{29}$$

3.2.2. Results and discussion.

3.2.2.1. Definition of Nusselt numbers. The local heat transfer coefficient h is expressed as

$$h = -k\frac{\partial T}{\partial n} \tag{30}$$

where k is the thermal conductivity. The average heat transfer coefficient $\overline{h}$ can be computed as

$$\overline{h} = \frac{1}{2\pi}\int_0^{2\pi} h\,\mathrm{d}\xi \tag{31}$$

The average Nusselt numbers for the inner and the outer boundaries are, respectively, determined by

$$\overline{Nu_\mathrm{i}} = \frac{\overline{h_\mathrm{i}}S_\mathrm{i}}{k}, \quad \overline{Nu_\mathrm{o}} = \frac{\overline{h_\mathrm{o}}S_\mathrm{o}}{k} \tag{32}$$

where S_i and S_o are defined in the same way as in the work of Moukalled and Acharya [20]. In their work, the computational domain is taken as half of the physical domain due to symmetry, so S_i and S_o are taken as half of the circumferential lengths of the inner and outer cylinder surfaces, respectively. As at steady state, the Nusselt numbers along the inner and outer walls are the same, there is no need to pay separate attention to $\overline{Nu_\mathrm{i}}$ and $\overline{Nu_\mathrm{o}}$. Thus in this study, we only show the value of $\overline{Nu_\mathrm{i}}$, which is also noted as $\overline{Nu}$.

3.2.2.2. Validation of numerical results. In the present study, Rayleigh number is fixed at 10^5 in a steady laminar boundary-layer regime, and Prandtl number is set to be 0.71. For the natural convection problem, the temperature T is considered as a very important flow parameter. Therefore,the temperature is chosen as the parameter of interest in this case. The upper and lower thresholds are set to 0.5 and 0.1, and the finest adaptive level is set to 3. During the computation, the adaptation checking procedure is activated for every 1000 iterations, to determine whether to insert or delete points. All the differential operators are discretized by the local MQ-DQ method and the resultant algebraic equations are solved by the successive over-relaxation (SOR) iterative method.

To the knowledge of the present authors, very few work has been done on the simulation of natural convection in a concentric annulus between a square outer cylinder and a circular inner cylinder. In the work of Moukalled and Acharya [20], three different aspect ratios and four different Rayleigh numbers have been considered. Their numerical data were validated by comparison with some experimental data and a good agreement was found. In the present study, the results of Moukalled and Acharya [20] will be used to validate the present numerical results. The maximum stream function value $\varphi_{\max}$ and the average Nusselt number $\overline{Nu}$ between the present work and work of Moukalled and Acharya [20] are compared in Table II for an aspect ratio of 2.5. It should be noted that due to different ways of non-dimensionalization between the work of Moukalled and Acharya [20] and the present work, the equivalent $\varphi_{\max}$ in Table II is the one given from Moukalled and Acharya [20] multiplying by the Prandtl number. From Table II, we can see that the results with an initial grid of 80×36 show significantly larger values for all the three parameters than and Acharya [20]. However, when the points are refined to resolution level 1, those by Moukalled the results agree much better with the reference results. It indicates that with the refinement of the grid, the accuracy of the solutions has been improved. With regard to efficiency, we can compare

Table II. Comparison of $\varphi_{\max}$ and $\overline{Nu}$ for aspect ratio $= 2.5$, $Ra = 10^5$, $Pr = 0.71$.

Different choices	Number of points used	$\varphi_{\max}$	$\varphi_{\min}$	$\overline{Nu}$	Time used (s)
80×36	$80 \times 36 = 2880$	8.481	−8.481	5.102	230
Level 1	4614	8.350	−8.351	5.016	464
160×71	$160 \times 71 = 11\,360$	8.347	−8.347	5.016	4017
Moukalled's	N.A.	8.38	N.A.	5.080	N.A.

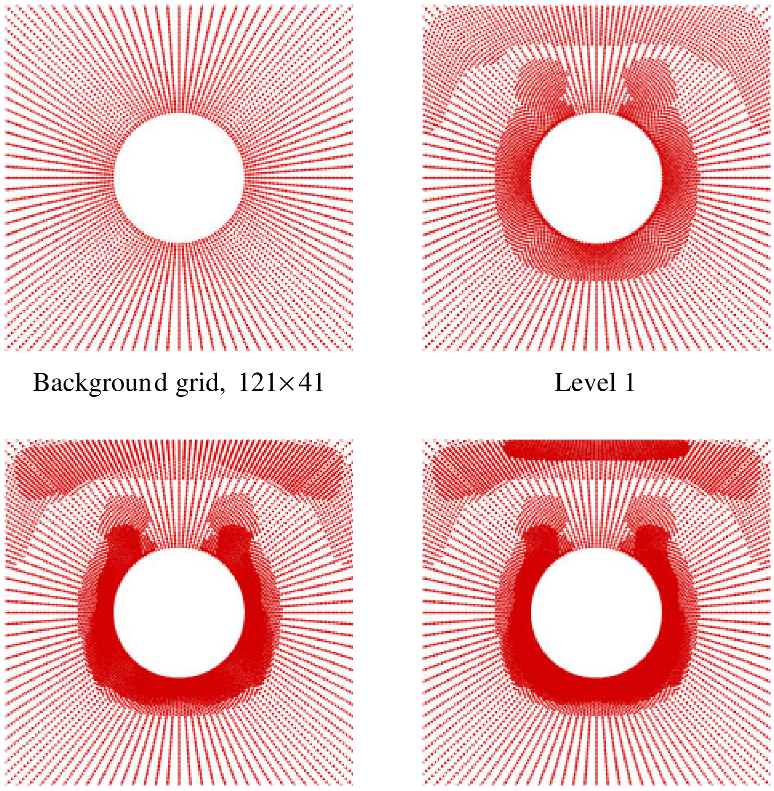

Figure 15. Final node distributions with different highest resolution levels.

the results of two cases. One is using the resolution level 1 grid whose initial grid is set to 80×36 and the other is using the fixed grid 160×71. From Table II, it can be seen that as compared with the fixed grid method on a 160×71 grid (11 360 grid points), our adaptive algorithm with one resolution level (4614 grid points) requires much less grid points and running time to achieve a solution of similar accuracy. The node distributions of using different resolution levels are shown in Figure 15 for an aspect ratio of 2.6. The isotherms and streamlines are displayed in Figures 16

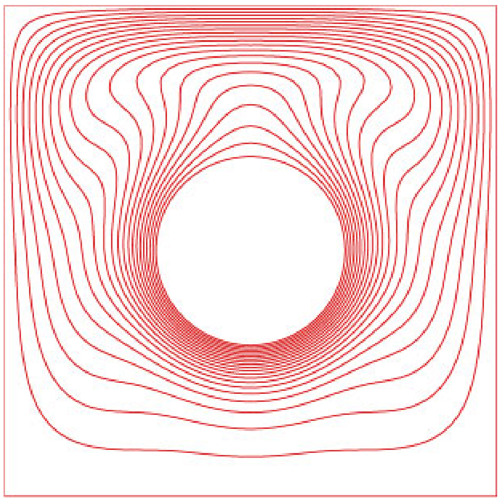

Figure 16. Isotherms for $Pr=0.71$, $Ra=10^5$ and $rr=2.6$.

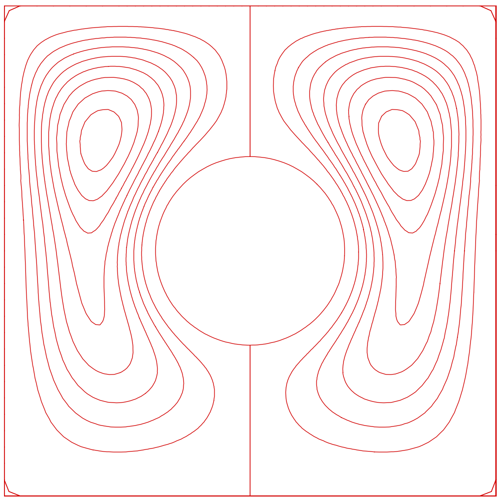

Figure 17. Streamlines for $Pr=0.71$, $Ra=10^5$ and $rr=2.6$.

and 17. Comparing Figures 15 and 16, we can see that with increasing resolution level, more and more points are inserted into the regions near the inner circular cylinder and the regions near the top boundary of the outer square cylinder, where the variation of temperature is relatively large. A clear correlation between the refined regions in Figure 15 and the temperature gradients in Figure 16 can be observed.

4. CONCLUSIONS

In this paper, an efficient local stencil adaptive algorithm has been presented for two-dimensional fluid flow problems with *curved boundaries*. This algorithm is able to automatically adjust the local stencils to reflect the gradient of the solution. The local MQ-DQ method has been used to discretize the governing equations because of its meshless nature. Two numerical experiments have been carried out to examine the performance of this method. Numerical results show that this method can effectively solve problems with *curved boundaries*. Furthermore, it can solve problems as accurately as the local MQ-DQ method does on a regular grid, but with less grid points and running time. As a result, this local MQ-DQ-based stencil adaptive method offers a promising approach to solve engineering problems with *curved boundaries*.

REFERENCES

1. Miller K, Miller RN. Moving finite elements. I. *SIAM Journal on Numerical Analysis* 1981; **18**:1019–1032.
2. Eiseman PR. Adaptive grid generation. *Computer Methods in Applied Mechanics and Engineering* 1987; **64**: 321–376.
3. Budd CJ, Carretero-Gonzalez R, Russell RD. Precise computations of chemotactic collapse using moving mesh methods. *Journal of Computational Physics* 2005; **202**:463–487.
4. Basebi T, Thomas RM. A study of moving mesh methods applied to a thin flame propagating in a detonator delay element. *Computers and Mathematics with Applications* 2003; **45**:131–163.
5. Beckett G, Mackenzie JA, Ramage A, Sloan DM. Computational solution of two-dimensional unsteady PDEs using moving mesh methods. *Journal of Computational Physics* 2002; **182**:478–495.
6. Berger MJ, Oliger J. Adaptive mesh refinement for shock hydrodynamics. *Journal of Computational Physics* 1984; **53**:484–512.
7. Berger MJ, LeVeque R. Adaptive mesh refinement for two-dimensional hyperbolic systems and the AMRCLAW software. *SIAM Journal on Numerical Analysis* 1998; **35**:2298–2316.
8. Zienkiewicz OC, Zhu JZ. The three R's engineering analysis and error estimation and adaptivity. *Computer Methods in Applied Mechanics and Engineering* 1990; **82**:95–113.
9. Ding H, Shu C. A stencil adaptive algorithm for finite difference solution of viscous incompressible flows. *Journal of Computational Physics* 2006; **214**:397–420.
10. Howell LH, Greenough JA. Radiation diffusion for multi-fluid Eulerian hydrodynamics with adaptive mesh refinement. *Journal of Computational Physics* 2003; **184**:53–78.
11. Hornung RD, Trangenstein JA. Adaptive mesh refinement and multilevel iteration for flow in porous media. *Journal of Computational Physics* 1997; **136**:522–545.
12. Anderson RW, Elliott NS, Pember RB. An arbitrary Lagrangian-Eulerian method with adaptive mesh refinement for the solution of the Euler equations. *Journal of Computational Physics* 2004; **199**:598–617.
13. Lucy LB. A numerical approach to the testing of the fission hypothesis. *Astronomical Journal* 1977; **8**:1013–1024.
14. Belytschko T, Lu YY, Gu L. Element-free Galerkin methods. *International Journal for Numerical Methods in Engineering* 1994; **37**:229–256.
15. Liu W, Jun S, Zhang Y. Reproducing kernel particle methods. *International Journal for Numerical Methods in Fluids* 1995; **20**:1081–1106.
16. Babuska I, Melenk J. The partition of unity method. *International Journal for Numerical Methods in Engineering* 1997; **40**:727–758.
17. Duarte CA, Oden JT. Hp clouds—a meshless method to solve boundary-value problems. *TICAM Report 95-05*.
18. Atluri SN, Zhu T. New meshless local Petrov–Galerkin (MLPG) approach in computational mechanics. *Computational Mechanics* 1998; **22**:117–127.
19. 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**:941–954.
20. Moukalled F, Acharya S. Natural convection in the annulus between concentric horizontal circular and square cylinders. *Journal of Thermophysics and Heat Transfer* 1996; **10**:524–531.