 CMES, vol.92, no.1, pp.1-31, 2013

Compact Local IRBF and Domain Decomposition Method for solving PDEs using a Distributed termination detection based parallel algorithm

N. Pham-Sy[1], C.-D. Tran[1], T.-T. Hoang-Trieu[1], N. Mai-Duy[1], T. Tran-Cong[1]

Abstract: Compact Local Integrated Radial Basis Function (CLIRBF) methods based on Cartesian grids can be effective numerical methods for solving partial differential equations (PDEs) for fluid flow problems. The combination of the domain decomposition method and function approximation using CLIRBF methods yields an effective coarse-grained parallel processing approach. This approach has enabled not only each sub-domain in the original analysis domain to be discretised by a separate CLIRBF network but also compact local stencils to be independently treated. The present algorithm, namely parallel CLIRBF, achieves higher throughput in solving large scale problems by, firstly, parallel processing of sub-regions which constitute the original domain and, secondly, accelerating the convergence rate within each sub-region using groups of CLIRBF stencils in which function approximations are carried out by parallel processes. The procedure is illustrated with several numerical examples of PDEs and lid-driven cavity problem using Message Passing Interface supported by MATLAB.

Keywords: Integrated RBFs, Compact Local Stencils, Domain Decomposition Method, Parallel Algorithm, Distributed Termination Detection.

1 Introduction

Radial Basis Functions (RBFs) have traditionally been used to provide a continuous interpolation of scattered data sets (Franke, 1982; Kansa, 1990). The Differential RBF (DRBF) based methods have been successfully used to solve a wide variety of differential equations. For this approach, once the field variables are known, its derivatives can be calculated through differentiation (Kansa, 1990; Zerroukat, Power, and Chen, 1998; Tran-Canh and Tran-Cong, 2004). Another approach namely the Integrated RBF (IRBF) method, which was proposed by Mai-Duy and

[1] Computational Engineering and Science Research Centre, Faculty of Engineering and Surveying, The University of Southern Queensland, Toowoomba, QLD 4350, Australia.

Tran-Cong (2001), is based on the approximation of the highest-order derivatives of the ODE/PDE using RBF at the first step, and subsequently its lower-order derivatives and the dependent variable itself are obtained by integration. The IRBF based methods can outperform other approximation methods based on the DRBF owing to its ability to produce very accurate solutions using relatively small number of data nodes.

Although full-domain IRBF methods are highly flexible and exhibit high convergence rates in their basic implementation, the associated fully-populated system matrices can lead to poor numerical conditioning as the scale of a problem increases (Mai-Duy and Tran-Cong, 2008). The problem becomes critical with increasingly large data sets. Many techniques have been developed to reduce the effect of the problem, including domain decompositions (Ingber, Chen, and Tanski, 2004; Tran, Phillips, and Tran-Cong, 2009), adaptive selection of data centres (Ling, Opfer, and Schaback, 2006), RBF preconditioners (Brown, 2005) and RBF based compact local stencil methods (Mai-Duy and Tran-Cong, 2011). While a reliable method of controlling numerical ill-conditioning and particularly computational cost, as problem scale increases, can be based on domain decomposition method (DDM), the use of compact local approximations facilitates the solution of a differential equation without having to deal with large systems of global equations. In this work, a parallel algorithm based on Compact Local Integrated RBF (CLIRBF) and DDM is developed for the solution of Boundary Value Problems (BVP). A large problem is firstly decomposed into many smaller problems each of which is analysed in parallel, and secondly the acceleration of the convergence rate within each sub-region using groups of CLIRBF stencils is carried out by parallel processes. For ease of presentation, in this paper the terms node, subdomain, process are used interchangeably.

One common problem associated with distributed computing is global termination of nodes over the system. This problem occurs more frequently when dealing with asymmetric problems, in which one node may converge faster than the others. In fact, this situation was first noticed by Francez (1980) and independently by Dijkstra and Scholten (1980), and named Distributed Termination Detection (DTD). In general, the goal of DTD is to detect whether a system is in its quiescent state. Quiescence is defined as a state, in which no node is active and no message is in transmission. As the system is distributed, there is no shared clock or memory involved. Moreover, if the system is required to be synchronous, the existed problem becomes more complex. In our parallel approach, DDM is used to divide the computational work among computer nodes. These nodes are distributed and synchronous because the result obtained from each node must be consistent within the whole domain, i.e. all nodes must stop at the same step. In this paper, we

propose a bitmap DTD algorithm which possesses some important advantages (i) it allows any node to detect termination (symmetry); (ii) it does not require any central control agent (decentralisation); and (iii) the message complexity of the proposed algorithm is nearly optimal.

This paper is organised as follows. In section 2, a brief review of a CLIRBF method is presented. The domain decomposition method as well as DTD are described in section 3 and section 4 respectively. Numerical examples are then discussed in section 5 with a conclusion in section 6.

2 Review of the IRBF collocation method

Consider a second-order ODE with boundary conditions as follows.

$$\mathscr{L}u = f, \quad x \in \Omega \tag{1}$$

$$\mathscr{B}u = g, \quad x \in \partial\Omega \tag{2}$$

where $\mathscr{L}$ is a second order differential operator; $\mathscr{B}$ - an operator imposed as boundary conditions such as Dirichlet, Neumann or a mixture of both; u - an unknown function; f and g - given functions; Ω and $\partial\Omega$ - the domain under consideration and its boundary. For brevity, the 1D-IRBF scheme for discretisation of ODEs is presented.

2.1 1D-IRBF collocation method

The function u along an x-gridline is represented in the IRBF form (Mai-Duy and Tran-Cong, 2001) as

$$\frac{d^2u}{dx^2} = \sum_{i=1}^{N} w_i g_i = \sum_{i=1}^{N} w_i G_i^{[2]}, \tag{3}$$

where $\{w_i\}_{i=1}^N$ is the set of RBF weights; and $\{g_i(x)\}_{i=1}^N$ the set of RBFs. In this work, the Multiquadric (MQ) RBF is used and given by (Haykin, 1999)

$$G_i(x) = \left((x-c_i)^2 + a_i^2\right)^{1/2},$$

where $\{c_i\}_{i=1}^N$ is a set of centres and $\{a_i\}_{i=1}^N$ a set of MQ-RBF widths.

The corresponding first-order derivative and function are then determined through integration as follows.

$$\frac{du}{dx} = \sum_{i=1}^{N} w_i G_i^{[1]} + C_1, \tag{4}$$

$$u = \sum_{i=1}^{N} w_i G_i^{[0]} + C_1 x + C_2, \tag{5}$$

where $G_i^{[1]}(x) = \int G_i^{[2]}(x)dx$, $G_i^{[0]}(x) = \int G_i^{[1]}(x)dx$ and C_1 and C_2 are unknown constants of integration. The superscript [.] is used to indicate the associated derivative order.

Collocating equations (3) - (5) at a set of grid points $\{x_i\}_{i=1}^{N}$ yields the following set of algebraic equations

$$\frac{d^2\widetilde{\mathbf{u}}}{dx^2} = \widetilde{G}^{[2]}\widetilde{\mathbf{w}}, \tag{6}$$

$$\frac{d\widetilde{\mathbf{u}}}{dx} = \widetilde{G}^{[1]}\widetilde{\mathbf{w}}, \tag{7}$$

$$\widetilde{\mathbf{u}} = \widetilde{G}^{[0]}\widetilde{\mathbf{w}}, \tag{8}$$

where

$$\widetilde{\mathbf{G}}^{[2]} = \begin{bmatrix} G_1^{[2]}(x_1) & G_2^{[2]}(x_1) & \cdots & G_N^{[2]}(x_1) & 0 & 0 \\ G_1^{[2]}(x_2) & G_2^{[2]}(x_2) & \cdots & G_N^{[2]}(x_2) & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ G_1^{[2]}(x_N) & G_2^{[2]}(x_N) & \cdots & G_N^{[2]}(x_N) & 0 & 0 \end{bmatrix},$$

$$\widetilde{\mathbf{G}}^{[1]} = \begin{bmatrix} G_1^{[1]}(x_1) & G_2^{[1]}(x_1) & \cdots & G_N^{[1]}(x_1) & 1 & 0 \\ G_1^{[1]}(x_2) & G_2^{[1]}(x_2) & \cdots & G_N^{[1]}(x_2) & 1 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ G_1^{[1]}(x_N) & G_2^{[1]}(x_N) & \cdots & G_N^{[1]}(x_N) & 1 & 0 \end{bmatrix},$$

$$\widetilde{\mathbf{G}}^{[0]} = \begin{bmatrix} G_1^{[0]}(x_1) & G_2^{[0]}(x_1) & \cdots & G_N^{[0]}(x_1) & x_1 & 1 \\ G_1^{[0]}(x_2) & G_2^{[0]}(x_2) & \cdots & G_N^{[0]}(x_2) & x_2 & 1 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ G_1^{[0]}(x_N) & G_2^{[0]}(x_N) & \cdots & G_N^{[0]}(x_N) & x_N & 1 \end{bmatrix},$$

$$\begin{aligned} \widetilde{\mathbf{w}} &= (w_1, w_2, \cdots, w_N, C_1, C_2)^T, \\ \widetilde{\mathbf{u}} &= (u_1, u_2, \cdots, u_N)^T, \\ \frac{d^k\widetilde{\mathbf{u}}}{dx^k} &= \left(\frac{d^k u_1}{dx^k}, \frac{d^k u_2}{dx^k}, \cdots, \frac{d^k u_N}{dx^k}\right)^T, \end{aligned}$$

where $u_i = u(x_i)$ with $i = 1, 2, \cdots, N$.

Because of the presence of integration constants of the IRBF based approximation, the system of equations (8) will become under-determined. In this case, one can beneficially introduce in the algebraic equation system additional constraints such as nodal or derivative values. Thus, the algebraic equation system (8) can be reformulated as follows.

$$\begin{pmatrix} \widetilde{\mathbf{u}} \\ \widetilde{\mathbf{h}} \end{pmatrix} = \begin{bmatrix} \widetilde{G}^{[0]} \\ \widetilde{L} \end{bmatrix} \widetilde{\mathbf{w}} = \widetilde{\mathscr{C}} \widetilde{\mathbf{w}}, \tag{9}$$

where $\widetilde{\mathbf{h}} = \widetilde{L}\widetilde{\mathbf{w}}$ are additional constraints. The conversion of the network-weight space into the physical space yields

$$\widetilde{\mathbf{w}} = \widetilde{\mathscr{C}}^{-1} \begin{pmatrix} \widetilde{\mathbf{u}} \\ \widetilde{\mathbf{h}} \end{pmatrix}, \tag{10}$$

where $\widetilde{\mathscr{C}}^{-1}$ is the conversion matrix. By substituting equation (10) into equations (6) and (7), the second and first-order derivatives of u are expressed in terms of nodal variable values as follows.

$$\begin{aligned} \frac{d^2\widetilde{\mathbf{u}}}{d^2x} &= \widetilde{\mathscr{D}}_2\widetilde{\mathbf{u}} + \widetilde{\mathbf{k}}_2, \\ \frac{d\widetilde{\mathbf{u}}}{dx} &= \widetilde{\mathscr{D}}_1\widetilde{\mathbf{u}} + \widetilde{\mathbf{k}}_1, \\ \widetilde{\mathbf{u}} &= \widetilde{\mathscr{D}}_0\widetilde{\mathbf{u}} + \widetilde{\mathbf{k}}_0, \end{aligned} \tag{11}$$

where $\widetilde{\mathscr{D}}_2$, $\widetilde{\mathscr{D}}_1$ and $\widetilde{\mathscr{D}}_0$ are known matrices of dimension $N \times N$; and $\widetilde{\mathbf{k}}_2$, $\widetilde{\mathbf{k}}_1$ and $\widetilde{\mathbf{k}}_0$ are known vectors of length N.

2.2 Compact local IRBF methods

The 1D IRBF approximation, presented in section 2.1, is now developed into compact local 3–point stencils to discretise differential equations, following Mai-Duy and Tran-Cong (2011).

Consider local stencils $LS_i = [x_{i-1}, x_i, x_{i+1}]$ associated with grid point x_i ($2 \leq i \leq n-1$) in a typical global 1D Cartesian grid line. Equations (3) - (5) will be collocated at a stencil with N = 3 grid points. In the context of the present 3-point local stencils, we choose additional constraints to be the imposition of the governing equation at certain nodes. Thus, Eq. (9) includes (i) a set of three equations representing nodal values of u over the LS_i and (ii) a set of two algebraic equations obtained by evaluating the governing differential equation (1) at x_{i-1} and x_{i+1}. As a result, values of function u and its derivatives at an arbitrary point x on the stencil are calculated in the physical space as Eq. (11) where $\widetilde{\mathbf{u}} = (u_{i-1}, u_i, u_{i+1})^T$ $\widetilde{\mathbf{h}} = (f_{i-1}, f_{i+1})^T$.

3 Overlapping Domain Decomposition Method

Although the compact local IRBF methods are expected to (i) enhance the computational accuracy and convergence rate, and (ii) reduce the computational time, they could suffer from the problem of numerical ill-condition for large scale problems. In order to deal with this issue, a combination of the present CLIRBF approach and a DDM is used to solve differential equations.

DDM aims to split the domain under consideration into smaller subdomains while guaranteeing the continuity at the splitting boundary. DDM algorithms can be grouped into two classes: (i) non-overlapping methods and (ii) overlapping methods (Smith, Bjorstad, and Gropp, 1996).

For the Multiplicative Schwarz overlapping method, since the artificial boundary values of each subdomain are updated from the most recent results of its neighbouring subdomains (NS), the subdomains have to be solved sequentially. Thus, it has little potential for parallel programming. One way to make it parallelisable is by grouping subdomains through black-white colouring technique (Quarteroni, 1999), so that all the NSs have different colours. By this way, in the first half-step only one group, for example the white one, executes simultaneously because they do not share any common boundary. In the next half-step, the other group (the black one) executes in the same manner.

For the Additive Schwarz overlapping (ASO) method, the values of artificial boundary in each subdomain at step $n+1$ are updated from the results of its NS's at step n. Since the data of each subdomain are independent of others during the computational process, the approach is naturally parallel-capable and its implementation is quite straightforward. In this paper, the additive overlapping method is implemented on the domain under consideration. We will present this method in more detail below.

3.1 Additive Schwarz overlapping method

For illustrative purposes, the domain Ω is divided into two subdomains Ω_1 and Ω_2. Let $\partial\Omega_1$ and $\partial\Omega_2$ be boundaries and Γ_1, Γ_2 the artificial boundaries (ABs) of Ω_1 and Ω_2 respectively (Fig. 1). The boundary condition imposed on artificial boundaries can be Dirichlet-Dirichlet, Dirichlet-Neumann or otherwise. The so-called interface is the overlapping area between two subdomains. For example, in Fig. 1 the interface consists of three points two points u_1, u_2 are the artificial boundaries and the middle point would be the interface between the two subdomains in a non-overlapping DD. The width of the overlapping area is one of several factors that affect the convergence rate of DDM. According to Smith, Bjorstad, and Gropp (1996) this width must lie in a range from 10 to 20 percent, of the width of

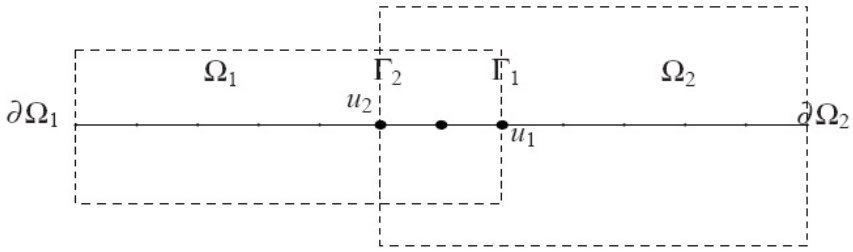


Figure 1: 1D Domain Decomposition with two subdomains Ω_1 and Ω_2

a subdomain. In our work, we choose the lower limit of this range, which is 10 percent.

In this work, the boundary condition imposed on Γ_1, Γ_2 is chosen to be Dirichlet-Dirichlet type because of its simple implementation. As the algorithm is iterative, equation (1) is written for the subdomains Ω_1 and Ω_2 respectively at a step n as follows.

$$\begin{aligned} \mathscr{L}u_1^n &= f, \quad x \in \Omega_1 \\ \mathscr{B}u_1^n &= g, \quad x \in \partial\Omega_1 \backslash \Gamma_1 \\ u_1^n &= u_{1\Omega_2}^{n-1}, x \in \Gamma_1 \end{aligned} \tag{12}$$

and

$$\begin{aligned} \mathscr{L}u_2^n &= f, \quad x \in \Omega_2 \\ \mathscr{B}u_2^n &= g, \quad x \in \partial\Omega_2 \backslash \Gamma_2 \\ u_2^n &= u_{2\Omega_1}^{n-1}, x \in \Gamma_2 \end{aligned} \tag{13}$$

where $u_{i\Omega_j}^{n-1}$ $(i, j \in \{1,2\}, i \neq j)$ is the value of artificial boundary u in Ω_i obtained from the solution in Ω_j at the step $n-1$ (see Fig. 1).

One important character of parallel DDMs is shared memory. In such systems all nodes have access to a shared memory and each node has to update its own record in that memory. Because of the nature of shared memory, as the number of nodes increases, memory access generates a bottleneck of the whole system. Another drawback of this approach is its high message complexity. Each node has to access the share memory at least once per iteration to update its state, which causes at least $(i \times n)$ (i - the number of iteration and n - the number of nodes) control messages to be communicated over the network. In order to overcome

above disadvantages, a shared bitmap that describes the termination status of the whole system is introduced. This idea will be discussed in section 4.

3.2 Algorithm of the present procedure

The present method can now be described in a more detailed algorithm whose flowchart is shown in Fig. 2 and consists of four main steps

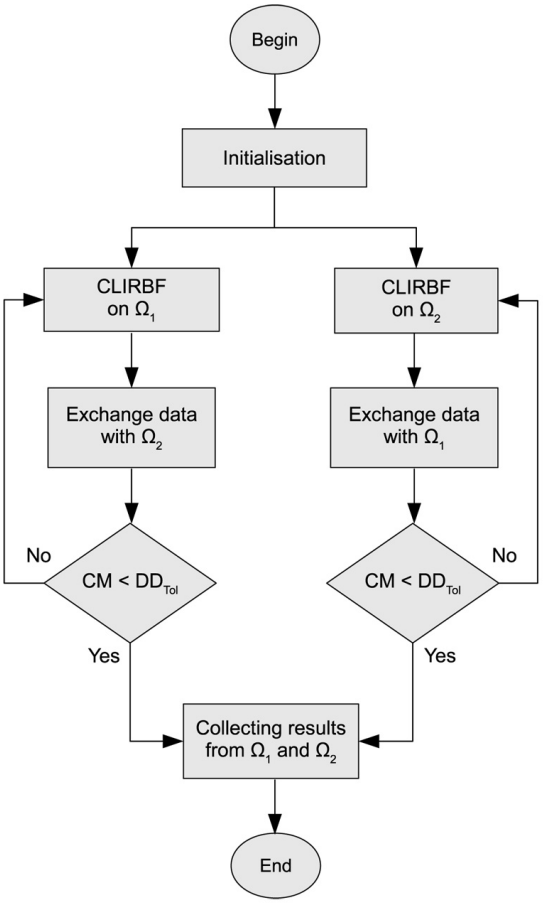


Figure 2: Parallel domain decomposition with 2 subdomains: CM - convergence measure; DD_{Tol} - predefined tolerance.

1. Divide the analysis domain into a number of subdomains. Guess initial boundary condition at artificial boundaries (ABs);

2. Solve the boundary value problem in each and every subdomain using Compact Local IRBF method;

3. Exchange the field variables or derivatives across ABs;

4. Check for the convergence on the interface.

For the case of two subdomains, it is quite easy to determine subdomains' neighbour and corresponding artificial boundary. However, with more than two subdomains, more effort is required as follows.

1. Enumerate subdomains;

2. Convert flat-index (Lab-index) into subdomain index (matrix index);

3. Determine the neighbours;

4. Determine real and artificial boundaries.

An example of this procedure is provided in Fig. 3

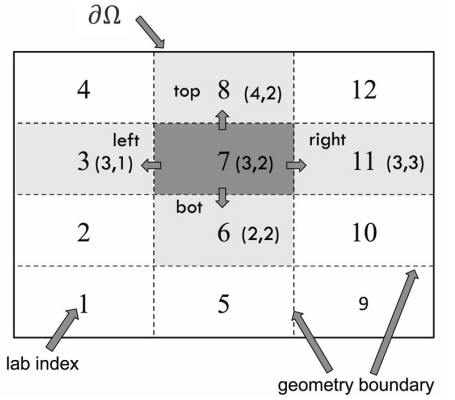


Figure 3: Enumeration in a system within $N_x = 4 \times N_y = 3$ subdomains: subdomains are enumerated from 1 to 12 and are geometrically indexed as in a matrix from bottom to top, left to right. In general, lab (i, j) has lab $(i, j-1)$ as its left neighbour if $(j-1) > 0$, lab $(i, j+1)$ as its right neighbour if $(j+1) \leq N_y$, lab $(i+1, j)$ as its top neighbour if $(i+1) \leq N_x$, and lab$(i-1, j)$ as its bottom neighbour if $(i-1) > 0$.

4 Parallel algorithm based on Distributed Termination Detection (DTD)

In a general distributed system, any node will terminate after finishing its job and the system is fully terminated only when all nodes have terminated. However, when solving a PDE using DDM based parallel algorithm, as the result obtained from all subdomains must be consistent, all nodes are required to terminate strictly at the same step. Clearly, if one node terminates while some other nodes are still active, those active nodes cannot exchange information with their terminated neighbours. Consequently, the system will end up with communication errors. A bitmap based DTD method is employed in the current parallel algorithm to synchronise the activity of the system and is described in this section.

During the DTD process, a shared data called bitmap is spread throughout the system and reflects the current state of all nodes. The use of bitmap makes the present algorithm symmetric because any node can detect the termination at any time. Furthermore, the algorithm requires neither a central node nor a spanning tree/graph to monitor the state of the system.

4.1 Terminology

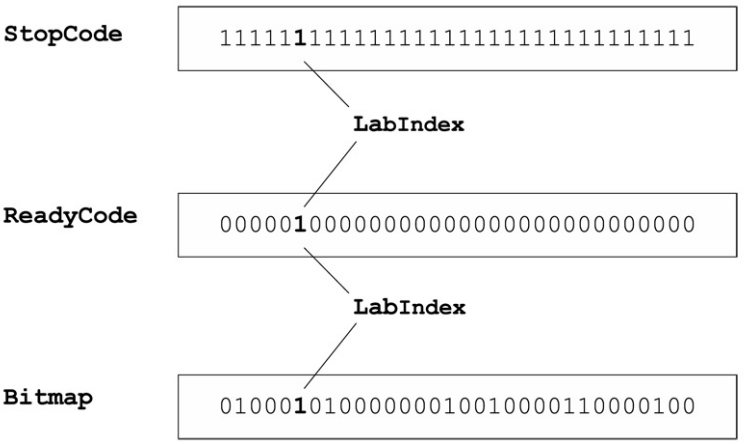


Figure 4: Terminology of bitmap DTD: StopCode, ReadyCode and a general bitmap for the sixth subdomain in a system of 32 subdomains

Bitmap is a map of binary bits (see Fig. 4). Each bit has two states **true** or **false**, which are correspondingly equivalent to two states of a node **ready** or **not-ready** to terminate. The length of a bitmap is equal to the number of nodes in the parallel work. For example, in order to encode states of 32 nodes, it is sufficient to use a

bitmap of 32 bits of data. In this example, the bitmap could have any value in the range from 0 to $(2^{32}-1)$. For ease of presentation, we define some technical terms as follows.

- **LabIndex**: The index of a node in system. This index is also corresponding to the position of a bit, reflecting the state of that node, in the bitmap;
- **StopCode**: A bitmap with all bit 1, i.e. all nodes are ready. When **StopCode** is achieved the termination is detected;
- **ReadyCode**: A node-specific bitmap with only one bit 1. This bitmap is used when that specific node becomes READY;
- **Step-To-End (STE)**: is the number of iterations left for each node to execute before termination.

4.2 Mechanism of bitmap DTD

The present DTD algorithm consists of two components: (i) Termination Detection and (ii) Synchronised Termination. The Termination Detection's role is to detect whether the system is in its quiescent state, i.e., all the nodes are ready to terminate. When that state is achieved, the Synchronised Termination is activated to get all nodes terminated simultaneously at the same step.

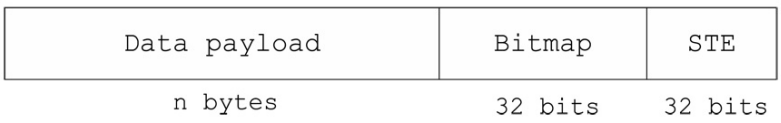


Figure 5: Message format in a system of 32 subdomains

The termination detection process runs along with the main computation as the data (including bitmap, see Fig. 5) are communicated between nodes. Each node keeps a record of its current bitmap which is initially empty. When a node receives a bitmap from one of its neighbours it updates its own bitmap by doing a binary union between its bitmap and the received bitmap. At a stage, a node detects termination if its bitmap is equal to the **StopCode**. Otherwise, this node continues to exchange its data and bitmap with neighbours. The Termination Detection algorithm is expressed by a flowchart in Fig. 6. When the termination condition is detected, the phase of Synchronous Termination is started. In this phase, the STE is

introduced to keep track of whether a node can actually terminate. The rule is simple, all STEs are initially set to (-1). Once the termination condition is detected by a node, its STE is set to a value $n+m-2$ (m,n: dimension of the matrix index, see section 3.2) which is decreased by one after each iteration or message exchange. The STE and other data (Fig. 5) are then transferred to its neighbours. The node's job will be terminated when its STE reaches zero.

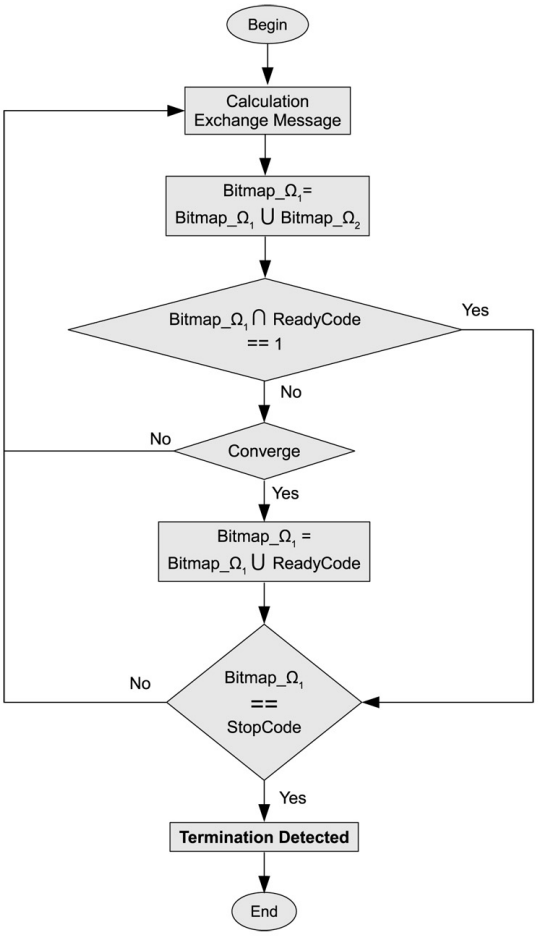


Figure 6: Bitmap Distributed Termination Detection algorithm on Ω_1

5 Numerical results

The proposed parallel method based on the combination of a CLIRBF method and parallel DDM to solve PDEs is verified using 1D and 2D problems with different boundary conditions. The capability and efficiency of the present method are then demonstrated with the simulation of the lid driven cavity (LDC) flow of a viscous fluid.

5.1 One dimensional problem

Consider the following second-order ODE.

$$\frac{d^2u}{dx^2}+\frac{du}{dx}+u=-\exp(-5x)\left[9979\sin(100x)+900\cos(100x)\right],\quad 0\le x\le 1, \quad (14)$$

with an analytic solution $u=\exp(-5x)\sin(100x)$.

This problem is solved using the present method with two different types of boundary condition. The domain is partitioned into 2 subdomains and a wide range of grids $(201,303,\ldots,501)$ is considered.

Dirichlet boundary condition

The Dirichlet conditions are $u(0)=0$ and $u(1)=\sin(100)\exp(-5)$.

The results show that the present method achieves the same accuracy level as the CLIRBF method. In fact, the relative L_2 error of the present method is $O(h^{\alpha})$, showing a convergence rate $\alpha=4.12$ while that for the CLIRBF is 4.26 (see Fig. 7 (bottom figure)). Figure 7 (top figure) depicts a comparison of the results obtained by the present method, the CLIRBF and the analytic one.

Dirichlet and Neumann boundary conditions

The Dirichlet condition is imposed on the left end $u(0)=0$ and the Neumann conditions on the right end $\frac{du(1)}{dx}=5\exp(-5)\left[20\cos(100)-\sin(100)\right]=0.598$.

While the result described in Fig. 8 (top) by the present method is in very good agreement with the analytic solution, the convergence rate displayed in Fig. 8 (bottom) shows that the present parallel method yields a higher accuracy in comparison with the CLIRBF method. Generally, the 1D example shows that the present parallel scheme based on CLIRBF and DD method can attain the numerical accuracy of corresponding schemes using single domain.

5.2 Two dimensional problem

Consider the following 2D problem

$$\frac{\partial^2u}{\partial x^2}+\frac{\partial^2u}{\partial y^2}=4(1-\pi^2)\sin(2\pi x)\sinh(2y)+16(1-\pi^2)\cosh(4x)\cos(4\pi y). \quad (15)$$

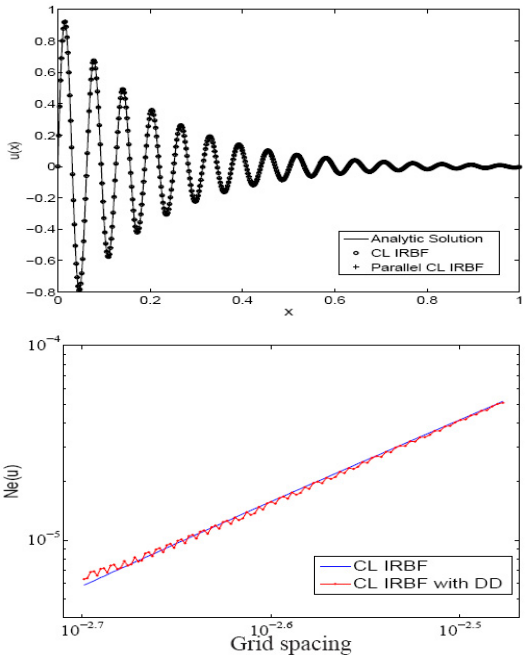


Figure 7: Second order problem with Dirichlet boundary condition: Solutions obtained by the 3-node CLIRBF method, the present parallel method and and the analytic solution (top figure); Relative L_2 errors of the solution u against the grid density by the 3-node CLIRBF method and the present method (bottom figure).

This problem is solved in the analysis domain and with the Dirichlet and Dirichlet/Neumann boundary conditions given in Fig. 9. The analytic solution is given by $u(x,y) = \sin(2\pi x)\sinh(2y) + \cosh(4x)\cos(4\pi y)$ and presented in Fig. 10(a).

A range of uniform grids (77×77, 113×113, 149×149, 185×185, 237×237) and subdomains (2×2, 3×3, 4×4, 5×5) are considered in the simulation. The results show that the numerical solution by the present parallel method whose computational parameters are given in Tables 1 and 2 is in very good agreement with the analytic solution (see Fig. 10). Indeed, Figure 10 presents the numerical solution of field variable u by the present parallel method with Dirichlet-Dirichlet boundary condition (Fig. 10(b)) and Dirichlet-Neumann boundary condition (Fig. 10(c)) using a grid of 77×77 collocation points and 2×2 subdomains in comparison with the analytic solution.

The results by the present parallel method presented in Tables 1 and 2 show that when grid density increases the present parallel domain decomposition method sig-

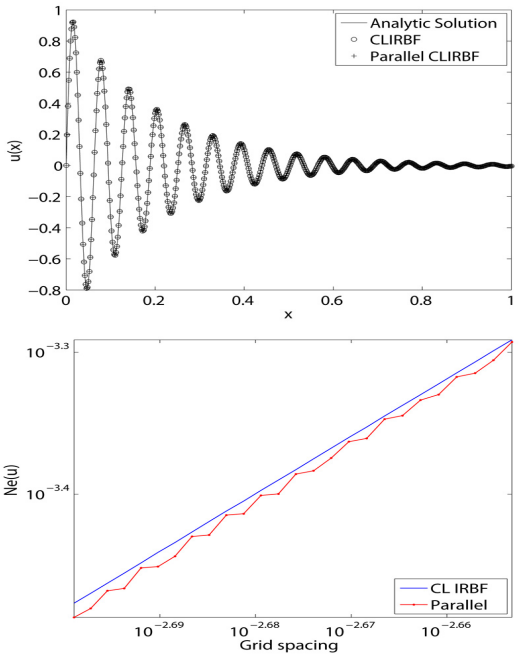


Figure 8: Second order problem with Dirichlet-Neumann boundary condition: Solutions obtained by the 3-node CLIRBF method, the present parallel method and and the analytic solution (top figure); Relative L_2 errors of the solution u against the grid density by the 3-node CLIRBF method and the present method (bottom figure).

nificantly increases the throughput. Indeed, the computation time is reduced significantly in comparison with non-parallel computation when increasing the number of collocation points to 185×185 as shown in Tables 1-2. Furthermore, the parallel algorithm is really efficient for solving large scale problems which require a large number of calculations as described in the next example.

5.3 Lid driven cavity problem

The lid-driven cavity (LDC) flow is commonly used as a typical example to benchmark numerical methods and therefore is also employed here to investigate the accuracy as well as the efficiency of the present parallel scheme. The problem is

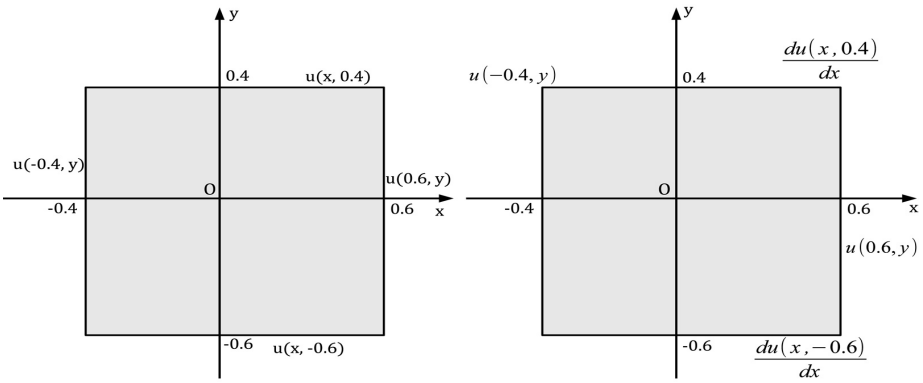


Figure 9: 2-D problem: Geometry of the analysis domain with 1) Dirichlet boundary conditions (left figure) and 2) Dirichlet-Neumann boundary conditions (right figure)

Table 1: 2D problem with Dirichlet Boundary conditions. $n_x \times n_y$: number of collocation points in the analysis domain; $CPUs$: number of CPUs; CM_{Tol}: the tolerance of convergence measure on the artificial interfaces; N_i: number of iterations; N_{es}: error norm for the single domain CLIRBF method; N_{ep}: error norm for the parallel CLIRBF-DD method; t_s: computation time for the single domain CLIRBF method; t_p: computation time for the parallel CLIRBF-DD method.

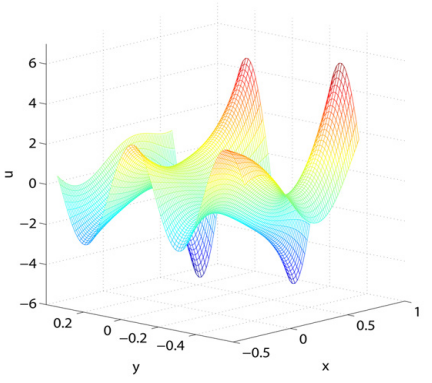
	Single Domain		Multi Domains - Parallel computation				
$n_x \times n_y$	N_{es}	t_s (sec)	$CPUs$	CM_{Tol}	N_i	N_{ep}	t_p (sec)
77×77	$2.963E-6$	25	4	$1.E-6$	124	$3.174E-6$	428
113×113	$1.882E-6$	104	9	$1.E-6$	217	$1.831E-6$	744
149×149	$1.443E-6$	383	16	$1.E-6$	384	$1.413E-6$	1311
185×185	$1.268E-6$	11234	25	$1.E-6$	515	$1.408E-6$	1739
237×237	n/a	n/a	25	$1.E-7$	577	$2.877E-6$	3873

defined in the stream function formulation as follows.

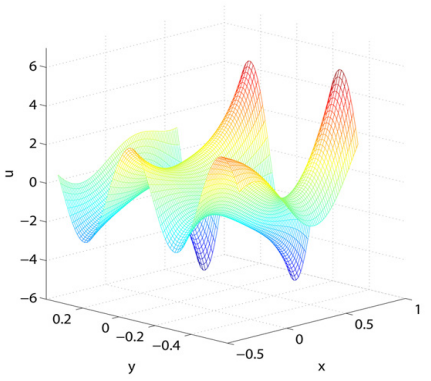
$$\frac{\partial \omega}{\partial t} + \left(\frac{\partial \psi}{\partial y}\frac{\partial \omega}{\partial x} - \frac{\partial \psi}{\partial x}\frac{\partial \omega}{\partial y}\right) = \frac{1}{\mathrm{Re}}\left(\frac{\partial^2 \omega}{\partial x^2} + \frac{\partial^2 \omega}{\partial y^2}\right), \tag{16}$$

$$-\omega = \frac{\partial^2 \psi}{\partial x^2} + \frac{\partial^2 \psi}{\partial y^2}, \tag{17}$$

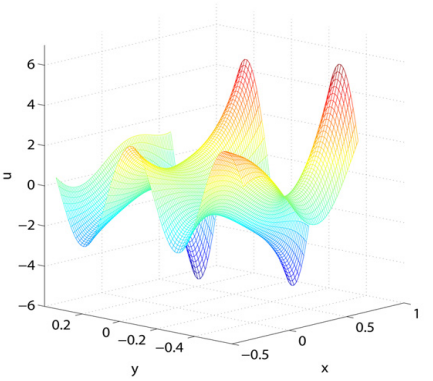
where Re is the Reynolds number, ψ the stream function, ω the vorticity and t the time. The geometry of the analysis domain with the chosen coordinate system is shown in Fig. 11. The $x-$ and $y-$ velocity components (u_x and u_y) are given by



(a)



(b)



(c)

Figure 10: 2D problem: (a) - Analytical solution; (b) - Present method with Dirichlet-Dirichlet boundary condition; (c) - Present method with Dirichlet-Neumann boundary condition.

$u_x = \frac{\partial \psi}{\partial y}$ and $u_y = -\frac{\partial \psi}{\partial x}$.

Table 2: 2D problem with Neumann and Dirichlet Boundary conditions. $n_x \times n_y$: number of collocation points in the analysis domain; $CPUs$: number of CPUs; CM_{Tol}: the tolerance of convergence measure on the artificial interfaces; N_i: number of iterations; N_{es}: error norm for the single domain CLIRBF method; N_{ep}: error norm for the parallel CLIRBF-DD method; t_s: computation time for the single domain CLIRBF method; t_p: computation time of the parallel CLIRBF-DD method.

	Single Domain		Multi Domains - Parallel computation				
$n_x \times n_y$	N_{es}	t_s (sec)	cpu	CM_{Tol}	N_i	N_{ep}	t_p (sec)
77×77	$1.523E-6$	19	4	$1.E-5$	132	$1.356E-6$	469
113×113	$5.898E-6$	54	9	$1.E-6$	660	$3.949E-6$	2294
149×149	$2.708E-6$	145	16	$5.E-7$	1150	$1.945E-6$	4039
185×185	$1.443E-6$	16560	25	$5.E-7$	921	$1.129E-6$	3193
237×237	n/a	n/a	25	$1.E-7$	661	$1.307E-6$	4379

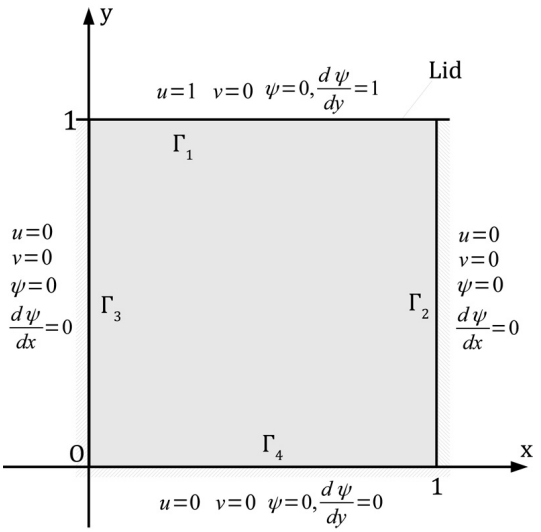


Figure 11: The lid driven square cavity problem: geometry of the analysis domain and boundary conditions in terms of the stream function.

The boundary conditions are given in terms of the stream function as

$$\psi = 0, \quad \frac{\partial \psi}{\partial x} = 0 \quad \forall (x,y) \in \Gamma_2 \cup \Gamma_3; \tag{18}$$

$$\psi = 0, \quad \frac{\partial \psi}{\partial y} = 0 \quad \forall (x,y) \in \Gamma_4; \tag{19}$$

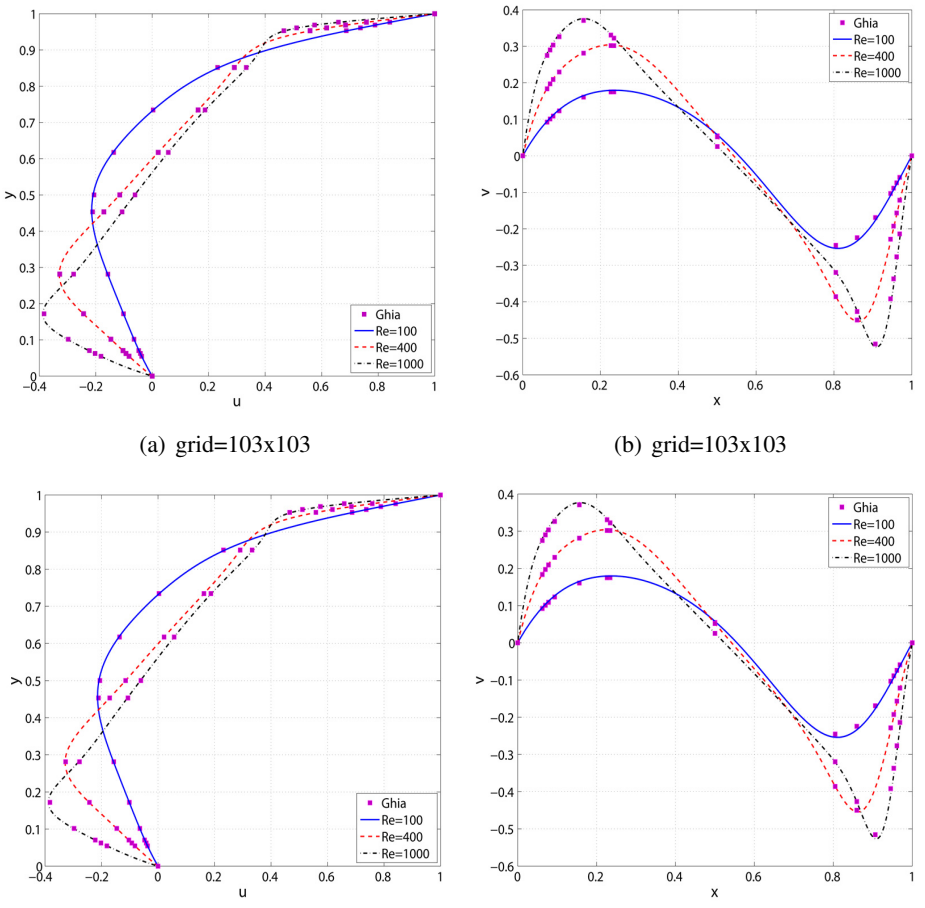
$$\psi = 0, \quad \frac{\partial \psi}{\partial y} = 1 \quad \forall (x,y) \in \Gamma_1. \tag{20}$$

The problem is solved by the present parallel algorithm using the 3×3 nodes CLIRBF - DD method. Let N ($N = N_x \times N_y$) be the number of subdomains and $(n_x \times n_y)$ the grid for the whole domain. Since each subdomain is governed by a separate CPU, N is also the number of CPUs used for the present parallel method. The algorithm for solving LDC problem can be described as follows.

1. Divide the analysis domain into a number of subdomains. Guess initial boundary condition at artificial boundaries (ABs);
2. Guess the initial values of ω;
3. Solve the LDC in each and every subdomain using CLIRBF method;
   (a) Solve Eq. (17) for ψ;
   (b) Approximate the values of ω on boundaries and the convective terms inside subdomain;
   (c) Solve Eq. (16) for ω;
4. Exchange the values of ψ and ω at interfaces with neighbours;
5. Calculate convergence measure on all interfaces;
6. Check for termination condition on all interfaces;
   If the procedure is not yet converged, replace values of ψ and ω at ABs with those received from neighbours and go back to step 3;
7. Stop the procedure.

A range of Re $(100, 400, 1000)$ and uniform grids are considered in the simulation. The time step (Δt) is chosen in the range from 10^{-4} to 10^{-6} and based on Re value and spatial grid size (smaller time steps are for finer grids and/or higher Re). The results by the present parallel method using a range of different subdomains (see Tables 3 - 5) are in very good agreement with the benchmark solution by Ghia, Ghia, and Shin (1982) using a multi-grid method.

Indeed, Fig. 12 depicts profiles of the velocities u and v along the vertical and horizontal centre-lines, respectively at several Reynolds numbers (100, 400 and 1000) with grids of 103×103 collocation points (Figs 12(a) - 12(b)) and 295×295 collocation points (Figs 12(c) - 12(d)) using the present parallel method with 5×5 subdomains. These results are in very good agreement with the benchmark solution by Ghia, Ghia, and Shin (1982) using a grid of 129×129 points. Although the present method can produce results in excellent agreement with Ghia's using only 103×103 grid points, a larger grid of 295×295 points is used to demonstrate a



(a) grid=103x103 (b) grid=103x103

(c) grid=295x295 (d) grid=295x295

Figure 12: The LDC fluid flow. Profiles of the u velocity along the vertical centre lines (a, c) and the v velocity along the horizontal centre lines (b, d) by the present parallel method at several Reynolds numbers ($Re = 100$, 400, and 1000) and grids (103×103, 295×295) in comparison with the corresponding Ghia's results. Although the present method can produce results in excellent agreement with Ghia's using only 103×103 grid points, a larger grid of 295×295 points is used to demonstrate a large scale solution.

large scale solution. Meanwhile the corresponding streamlines and vorticity contours in Fig. 13 by the method show a very good agreement with benchmark solution by Ghia, Ghia, and Shin (1982) (Figs. 3 and 4) at Reynolds numbers Re = 100, 400 and 1000 and from Botella and Peyret (1998) at Reynolds numbers Re

= 100, 1000. Furthermore, secondary vortices at the bottom corners (Figs. 13(a), 13(c) and 13(e) using grid 103×103 and Figs. 14(a), 14(c) and 14(e) using grid 295×295) are well captured by the present method.

The efficiency of the present parallel method is considered. The general goal is (i) to assign each processor an equal amount of work to be completed as fast as possible and (ii) to minimise the amount of communication between processors by essentially minimising the surface area of the subdomains. Results described in Tables 3 - 5 show that the computation time of the present parallel method for the solution of non-linear time-dependent problems using an iterative method decreases massively with respect to the number of CPUs used, for example, the throughput is $346.66, 100.48, 49.44, \ldots, 2.61$ minutes, respectively using the number of CPUs $2, 4, 6, \ldots, 36$ for the grid 101×101 (Table 3). An interpretation on this significant improvement of throughput will be presented below (Eq. 21).

Fig. 15 shows the efficiency of the present parallel algorithm with respect to the number of CPUs for the solution of LDC problem for several Reynolds numbers ($Re = 100$, 400 and 1000) using a range of grid densities ($n_x \times n_y$ with $n_x = n_y \in \{101, 155, 209, 253\}$). The results show that the computation time of the present parallel algorithm decreases tremendously as the number of subdomains/CPUs increases. This correctly reflects the scalability nature of parallel computing. However there are always some thresholds (called $cpus_{opt}$) over which the increase of number of CPUs influences insignificantly on the computation time. For example, the improvement of efficiency of computation is not significant anymore as the number of CPUs is more than 10 for a grid of 101×101 (Fig. 15(a)), 25 for a grid of 155×151 (Fig. 15(b)) and $32 - 35$ for a grid of 209×209 (Fig. 15(c)) and 253×253 (Fig. 15(d)). It shows that the choice of the scale for subdomains plays an important role in the high performance computation using parallel schemes. The detailed results are provided in Tables 3 - 5. Furthermore, the method removes the ill-conditioning of the global system matrix evidenced by a significant decrease of the condition number of the system matrix associated with the IRBF method (see column 3 of Table 6 and Fig. 17).

Fig.16 depicts the influence of the grid density on the efficiency of the present parallel algorithm with respect to the number of CPUs for the solution of the LDC problem for several Reynolds numbers ($Re = 100, 400$ and 1000). Indeed, the gradient (slope) of time curves for larger numbers of collocation points is steeper for any Reynolds number, which indicates that the efficiency of the present parallel method will be higher for larger scale problems (i.e. larger numbers of collocation points).

It is worth noting that the efficiency of present parallel method for iterative fluid flow problem is very high and increases gradually with respect to the number of

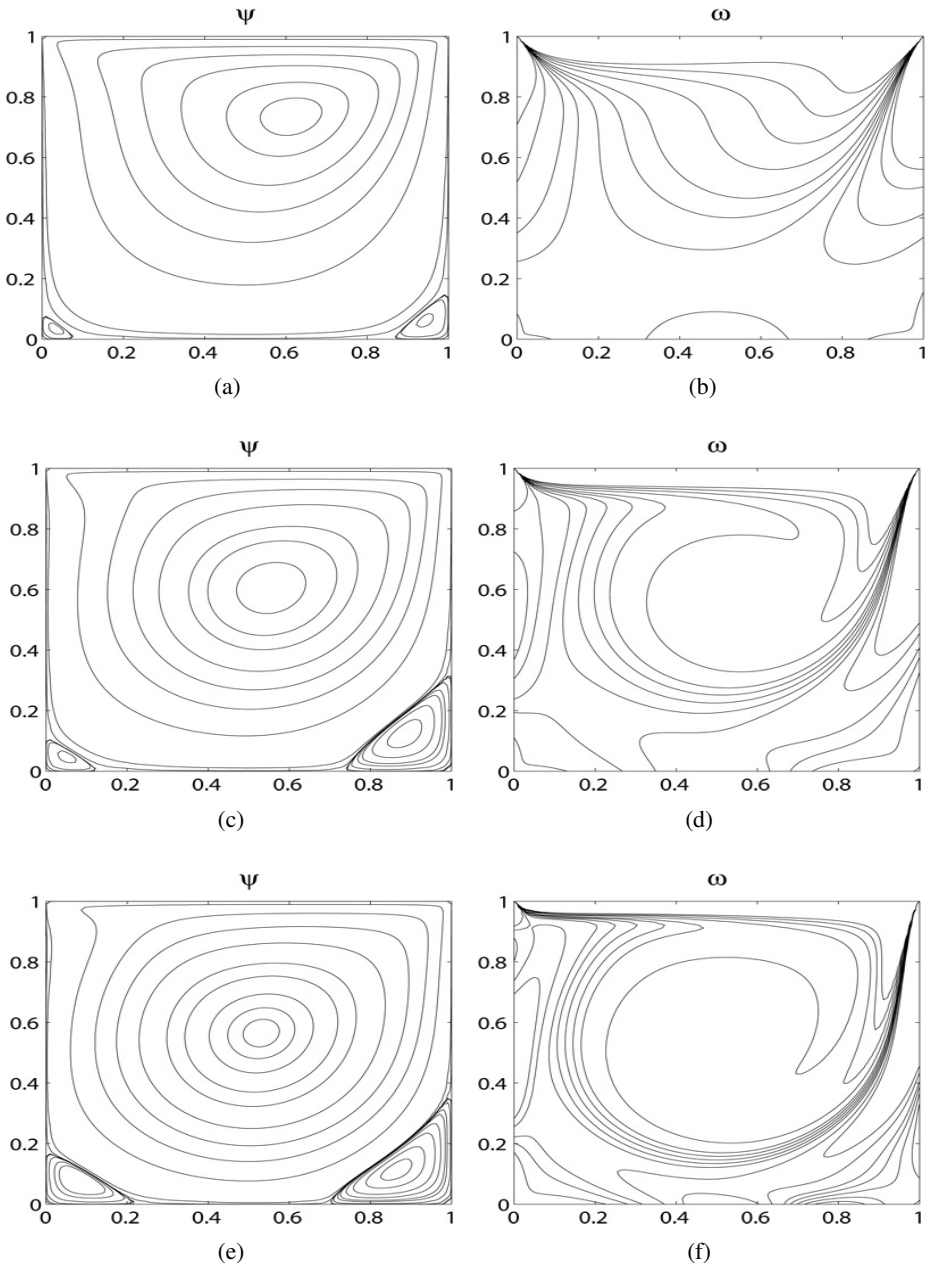


Figure 13: The LDC fluid flow. Streamlines and vorticity contours of the flow for several Reynolds numbers ($Re = 100$, 400, and 1000) by the present parallel method using a grid of 103×103 of points.

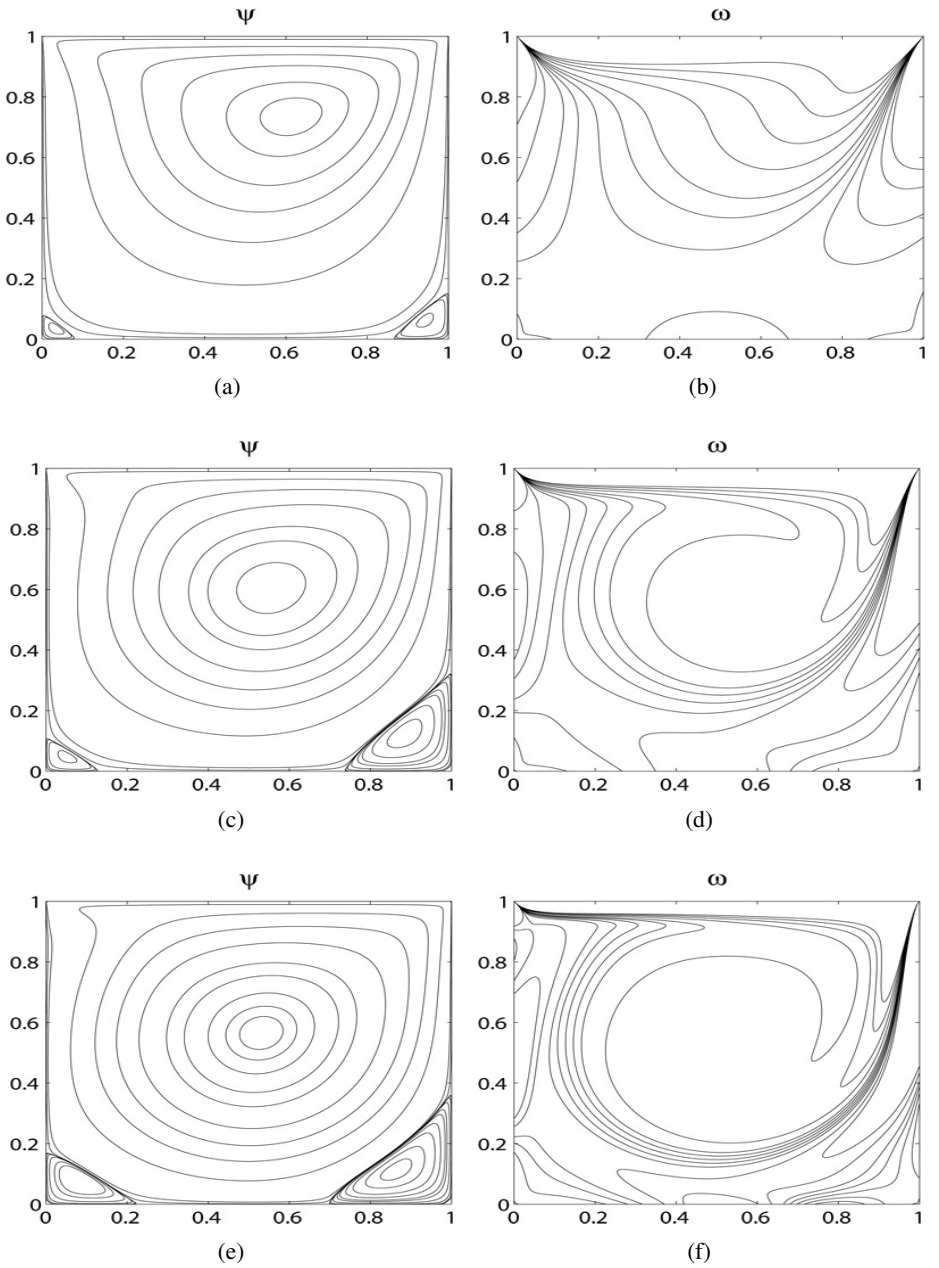


Figure 14: The LDC fluid flow. Streamlines and vorticity contours of the flow for several Reynolds numbers ($Re = 100$, 400, and 1000) using a grid of 295×295 points.

Table 3: Parallel computation of the LDC problem with $Re = 100$. $n_x \times n_y$: grid of collocation points; $CPUs$ ($N_x \times N_y$): number of CPUs (number of subdomains); Δt: time step; CM_{Tol}: tolerance of convergence measure at interfaces; N_i: number of iterations; $t(m)$: elapsed time (minutes); CM_p: average convergence measure of the whole analysis domain.

$n_x \times n_y$	$CPUs$	Δt	CM_{Tol}	N_i	$t(m)$	CM_p
101×101	2	$1.E-5$	$1.E-6$	2983	346.66	$3.3862E-07$
–	4	$1.E-5$	$1.E-6$	3039	100.48	$3.2612E-07$
–	6	$1.E-5$	$1.E-6$	3130	49.44	$3.0900E-07$
–	8	$1.E-5$	$1.E-6$	3127	26.32	$3.1188E-07$
–	9	$1.E-5$	$1.E-6$	3153	23.90	$2.6812E-07$
–	12	$1.E-5$	$1.E-6$	3141	13.69	$2.7585E-07$
–	15	$1.E-5$	$1.E-6$	3140	10.22	$2.5592E-07$
–	16	$1.E-5$	$1.E-6$	3142	7.96	$2.5326E-07$
–	20	$1.E-5$	$1.E-6$	3139	5.97	$2.3442E-07$
–	25	$1.E-5$	$1.E-6$	3128	4.54	$2.2448E-07$
–	28	$1.E-5$	$1.E-6$	3125	3.59	$2.1013E-07$
–	30	$1.E-5$	$1.E-6$	3123	3.10	$2.1792E-07$
–	35	$1.E-5$	$1.E-6$	3119	2.70	$2.0436E-07$
–	36	$1.E-5$	$1.E-6$	3118	2.61	$2.0544E-07$
155×155	9	$5.E-6$	$1.E-06$	5806	190.00	$2.8840E-07$
–	12	$5.E-6$	$1.E-6$	5822	110.39	$2.8021E-07$
–	15	$5.E-6$	$1.E-6$	5828	74.76	$2.7005E-07$
–	16	$5.E-6$	$1.E-6$	5837	63.17	$2.5711E-07$
–	20	$5.E-6$	$1.E-6$	5842	41.61	$2.4535E-07$
–	25	$5.E-6$	$1.E-6$	5837	29.23	$2.3962E-07$
–	28	$5.E-6$	$1.E-6$	5764	25.24	$2.2321E-07$
–	30	$5.E-6$	$1.E-6$	5820	23.00	$2.1753E-07$
–	35	$5.E-6$	$1.E-6$	5813	18.27	$2.0763E-07$
–	36	$5.E-6$	$1.E-6$	5732	17.94	$2.1531E-07$
209×209	16	$5.E-6$	$1.E-06$	5862	190.41	$3.4286E-07$
–	20	$5.E-6$	$1.E-6$	5968	134.30	$2.8124E-07$
–	25	$5.E-6$	$1.E-6$	5835	89.89	$3.3027E-07$
–	28	$5.E-6$	$1.E-6$	5823	74.26	$2.6927E-07$
–	30	$5.E-6$	$1.E-6$	5854	61.78	$3.0911E-07$
–	35	$5.E-6$	$1.E-6$	5820	51.69	$2.7399E-07$
–	36	$5.E-6$	$1.E-6$	5820	45.19	$2.2564E-07$
253×253	25	$5.E-6$	$1.E-6$	6715	227.97	$2.9365E-07$
–	28	$5.E-6$	$1.E-6$	6598	190.89	$2.8132E-07$
–	30	$5.E-6$	$1.E-6$	6673	149.42	$2.6219E-07$
–	35	$5.E-6$	$1.E-6$	6619	115.89	$2.4176E-07$
–	36	$5.E-6$	$1.E-6$	6537	101.33	$2.5564E-07$
295×295	36	$4.E-6$	$1.E-6$	8437	313.05	$2.4598E-07$
299×299	42	$4.E-6$	$1.E-6$	8485	207.59	$2.0221E-07$

Table 4: Parallel computation of the LDC problem with $Re = 400$. $n_x \times n_y$: grid of collocation points; $CPUs$ ($N_x \times N_y$): number of CPUs (number of subdomains); Δt: time step; CM_{Tol}: tolerance of convergence measure at interfaces; N_i: number of iterations; $t(m)$: elapsed time (minutes); CM_p: average convergence measure of the whole analysis domain.

$n_x \times n_y$	$CPUs$	Δt	CM_{Tol}	N_i	$t(m)$	CM_p
101×101	2	$1.E-5$	$1.E-6$	1769	199.67	$3.6469E-07$
–	4	$1.E-5$	$1.E-6$	1809	59.51	$3.4120E-07$
–	6	$1.E-5$	$1.E-6$	1907	30.38	$3.1543E-07$
–	8	$1.E-5$	$1.E-6$	1943	17.09	$3.0031E-07$
–	9	$1.E-5$	$1.E-6$	1949	15.34	$2.6732E-07$
–	12	$1.E-5$	$1.E-6$	1980	8.55	$2.6361E-07$
–	15	$1.E-5$	$1.E-6$	1988	6.55	$2.6674E-07$
–	16	$1.E-5$	$1.E-6$	2044	5.06	$2.5307E-07$
–	20	$1.E-5$	$1.E-6$	2092	3.89	$2.2655E-07$
–	25	$1.E-5$	$1.E-6$	2163	3.10	$2.0506E-07$
–	28	$1.E-5$	$1.E-6$	2184	2.51	$2.1807E-07$
–	30	$1.E-5$	$1.E-6$	2199	2.26	$1.9648E-07$
–	35	$1.E-5$	$1.E-6$	2330	2.09	$1.4480E-07$
–	36	$1.E-5$	$1.E-6$	2359	1.86	$1.3151E-07$
155×155	9	$1.E-5$	$1.E-6$	1957	117.43	$2.2188E-07$
–	12	$1.E-5$	$1.E-6$	1978	38.02	$2.0113E-07$
–	15	$1.E-5$	$1.E-6$	1995	25.64	$1.9335E-07$
–	16	$1.E-5$	$1.E-6$	1993	21.82	$2.1742E-07$
–	20	$1.E-5$	$1.E-6$	1996	14.64	$2.1968E-07$
–	25	$1.E-5$	$1.E-6$	2047	10.41	$1.8317E-07$
–	28	$1.E-5$	$1.E-6$	2303	10.34	$1.7704E-07$
–	30	$1.E-5$	$1.E-6$	2292	9.23	$1.7548E-07$
–	35	$1.E-5$	$1.E-6$	2398	7.54	$1.4143E-07$
–	36	$1.E-5$	$1.E-6$	2577	6.54	$1.0865E-07$
209×209	16	$1.E-5$	$1.E-6$	2088	68.21	$3.5483E-07$
–	20	$1.E-5$	$1.E-6$	2192	49.50	$3.5207E-07$
–	25	$1.E-5$	$1.E-6$	2305	35.96	$4.3792E-07$
–	28	$1.E-5$	$1.E-6$	2245	28.56	$3.1715E-07$
–	30	$1.E-5$	$1.E-6$	2293	25.80	$3.5021E-07$
–	35	$1.E-5$	$1.E-6$	2453	23.16	$2.6163E-07$
–	36	$1.E-5$	$1.E-6$	2393	18.43	$2.2997E-07$
253×253	25	$8.E-6$	$1.E-6$	2814	91.87	$3.4725E-07$
–	28	$8.E-6$	$1.E-6$	2980	75.47	$4.1144E-07$
–	30	$8.E-6$	$1.E-6$	3015	67.91	$3.6694E-07$
–	35	$8.E-6$	$1.E-6$	3195	58.14	$2.4023E-07$
–	36	$8.E-6$	$1.E-6$	3236	49.99	$2.7969E-07$
295×295	36	$4.E-6$	$1.E-6$	5164	171.74	$2.5456E-07$
299×299	42	$4.E-6$	$1.E-6$	5333	130.06	$1.9579E-07$

Table 5: Parallel computation of the LDC problem with $Re = 1000$. $n_x \times n_y$: grid of collocation points; $CPUs$ ($N_x \times N_y$): number of CPUs (number of subdomains); Δt: time step; CM_{Tol}: tolerance of convergence measure at interfaces; N_i: number of iterations; $t(m)$: elapsed time (minutes); CM_p: average convergence measure of the whole analysis domain.

$n_x \times n_y$	$CPUs$	Δt	CM_{Tol}	N_i	$t(m)$	CM_p
101×101	2	$1.E-6$	$1.E-6$	9292	972.00	$2.7711E-07$
–	4	$1.E-6$	$1.E-6$	9146	301.05	$2.9968E-07$
–	6	$1.E-6$	$1.E-6$	8958	246.58	$3.6250E-07$
–	8	$1.E-6$	$1.E-6$	8789	76.40	$3.9178E-07$
–	9	$1.E-6$	$1.E-6$	8964	68.32	$3.3912E-07$
–	12	$1.E-6$	$1.E-6$	8936	37.71	$3.1141E-07$
–	15	$1.E-6$	$1.E-6$	8901	29.06	$2.8634E-07$
–	16	$1.E-6$	$1.E-6$	8866	21.68	$2.4941E-07$
–	20	$1.E-6$	$1.E-6$	8862	16.27	$2.2520E-07$
–	25	$1.E-6$	$1.E-6$	8881	12.11	$2.1926E-07$
–	28	$1.E-6$	$1.E-6$	9062	9.54	$1.9764E-07$
–	30	$1.E-6$	$1.E-6$	8685	8.54	$2.4913E-07$
–	35	$1.E-6$	$1.E-6$	8764	7.27	$2.3634E-07$
–	36	$1.E-6$	$1.E-6$	9326	6.71	$1.5421E-07$
155×155	9	$1.E-6$	$1.E-6$	8887	290.22	$3.5905E-07$
–	12	$1.E-6$	$1.E-6$	8973	170.57	$3.2145E-07$
–	15	$1.E-6$	$1.E-6$	8979	137.84	$3.0020E-07$
–	16	$1.E-6$	$1.E-6$	9018	97.69	$2.6282E-07$
–	20	$1.E-6$	$1.E-6$	9072	79.81	$2.3382E-07$
–	25	$1.E-6$	$1.E-6$	9094	46.22	$2.1900E-07$
–	28	$1.E-6$	$1.E-6$	9565	43.57	$1.8332E-07$
–	30	$1.E-6$	$1.E-6$	9171	36.38	$2.2463E-07$
–	35	$1.E-6$	$1.E-6$	9901	29.89	$1.5324E-07$
–	36	$1.E-6$	$1.E-6$	11704	36.81	$1.2794E-07$
209×209	16	$5.E-6$	$1.E-06$	3435	203.43	$1.7919E-07$
–	20	$5.E-6$	$1.E-6$	3738	148.03	$1.6568E-07$
–	25	$5.E-6$	$1.E-6$	3944	107.23	$1.6116E-07$
–	28	$5.E-6$	$1.E-6$	3890	75.06	$1.6030E-07$
–	30	$5.E-6$	$1.E-6$	3988	69.16	$1.5540E-07$
–	35	$5.E-6$	$1.E-6$	4118	60.20	$1.3527E-07$
–	36	$5.E-6$	$1.E-6$	4268	54.84	$1.0258E-07$
253×253	25	$5.E-6$	$1.E-6$	3798	227.09	$1.5637E-07$
–	28	$5.E-6$	$1.E-6$	4043	160.56	$1.5123E-07$
–	30	$5.E-6$	$1.E-6$	4042	139.30	$1.4436E-07$
–	35	$5.E-6$	$1.E-6$	4131	124.96	$1.3336E-07$
–	36	$5.E-6$	$1.E-6$	4486	114.92	$8.8219E-08$
295×295	36	$5.E-6$	$1.E-6$	4550	271.60	$2.3908E-07$
299×295	42	$5.E-6$	$1.E-6$	4849	212.26	$2.0420E-07$

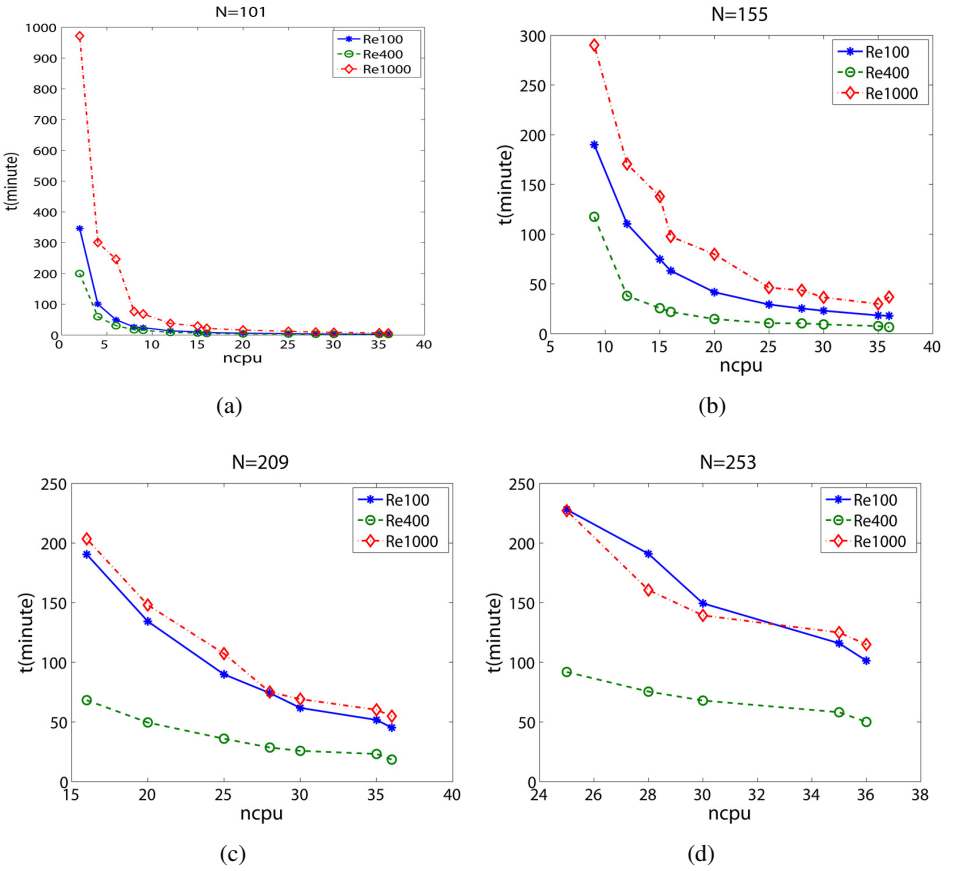


Figure 15: The LDC fluid flow. Computation time versus number of CPUs for the present parallel method with different number of collocation points $n_x \times n_y$ ($n_x = n_y = N$).

CPUs until it reaches some threshold ($cpus_{opt}$). Indeed, although the communication time among processors is generally orders of magnitude slower than calculation time within a single processor, due to a great number of iterations, the computation time for each subdomain plays a key role in improving the throughput of the present parallel method. One typical numerical example is given in Table 6 and the extensive improvement of throughput of the present parallel method can be explained as follows.

Let N_p be the number of iterations in the whole domain, t_{cal} the average computational time (second) in subdomains for each iteration; t_{com} the total communication

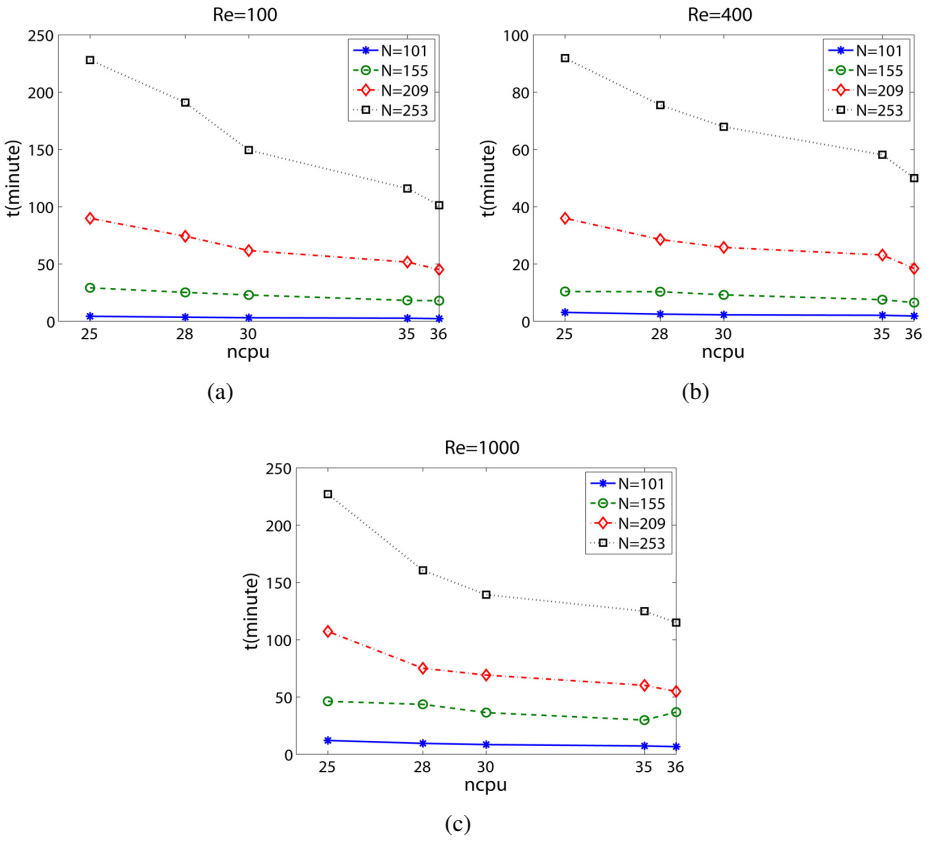


Figure 16: The LDC fluid flow. Computation time versus number of CPUs when solving LDC using the present parallel method with different Reynolds numbers: $Re = 100$ (Fig. 16(a)); $Re = 400$ (Fig. 16(b)); $Re = 1000$ (Fig. 16(c)).

time (minutes). The total execution time (elapsed time) T_{total} (minutes) is given by

$$T_{total} = \frac{N_p \times t_{cal}}{60} + t_{com} \tag{21}$$

Table 6 shows that while the number of iterations for different cases of parallel computation is similar (approximately 3000 for $CM_p = 3.E-7$), the average computation time t_{cal} for each iteration in subdomains decreases quickly with increasing number of CPUs or subdomains (scale of sub-problems) (Figure 17 (top)), resulting in a very high rate of throughput.

Table 6: Parallel computation using the present method for the LDC problem with $Re = 100$ and $\Delta t = 1.E-5$. $n_x \times n_y$: grid points; $CPUs$ $(N_x \times N_y)$: number of CPUs (number of subdomains); $Cond.\ Num$: condition number; CM_p: average convergence measure of the whole analysis domain; N_p: number of iterations on the whole domain; $t_{cal}(s)$: average computational time (second) in subdomains for each iteration; $t_{com}(m)$: total communication time (minutes); $T_{total}(m)$: elapsed time (minutes).

$n_x \times n_y$	$CPUs$	$Cond.\ Num$	CM_p	N_p	$t_{cal}(s)$	$t_{com}(m)$	$T_{total}(m)$
101×101	2	2190.99	$3.3862E-7$	2983	6.891	4.06	346.66
$--$	4	1389.97	$3.2612E-7$	3039	1.971	0.63	100.48
$--$	9	617.80	$2.6812E-7$	3153	0.350	5.48	23.90
$--$	16	322.27	$2.5326E-7$	3142	0.143	0.45	7.96
$--$	25	230.75	$2.2448E-7$	3128	0.065	1.14	4.54
$--$	36	154.49	$2.0544E-7$	3118	0.037	0.67	2.61

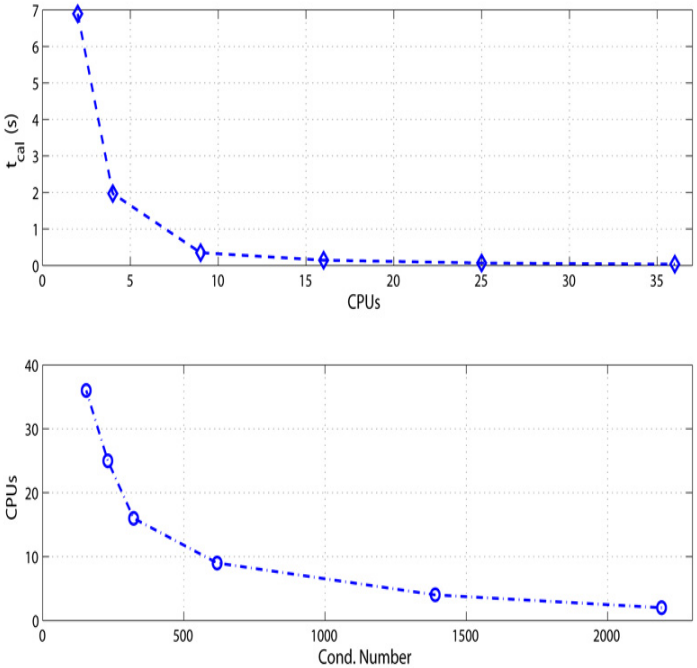


Figure 17: The LDC problem. Computation time (Top Figure) and Condition numbers (Bottom Figure) of the present parallel computation method using different number of CPUs with a grid of 101×101.

6 Conclusion

In this paper, we have developed a parallel algorithm based on the combination of the overlapping domain decomposition technique and CLIRBF methods. The proposed algorithm allows not only a large scale problem to be discretised by parallel CLIRBF processes but also compact local stencils to be independently treated in multiple-core CPUs. Advantages of the new approach include (i) to alleviate the ill-condition problem associated with IRBF methods; (ii) to avoid the reduction in convergence rate caused by differentiation and (iii) to achieve much higher throughput in solving large scale problems. The method is verified with several numerical examples using Matlab Distributed Computing Engine.

Acknowledgement

The first author would like to thank the CESRC, FoES and University of Southern Queensland for a Ph.D. scholarship. This research was supported by the Australian Research Council.

References

Botella, O.; Peyret, R. (1998): Benchmark spectral results on the Lid-driven cavity flow. *Computers & Fluids*, vol. 27, pp. 421–433.

Brown, D. (2005): On approximate cardinal preconditioning methods for solving pdes with radial basis functions. *Engineering Analysis with Boundary Elements*, vol. 29(4), pp. 343–353.

Dijkstra, E. W.; Scholten, C. (1980): Termination detection for diffusing computation. *Information Processing Letters*, vol. 11, pp. 1–4.

Francez, N. (1980): Distributed termination. *ACM Transaction on Programming Languages and Systems*, vol. 2, pp. 42–55.

Franke, R. (1982): Scattered data interpolation: Tests of some methods. *Mathematics of Computation*, vol. 38 (157).

Ghia, U.; Ghia, K.; Shin, C. (1982): High-Re solutions for Incompressible flow using the Navier-Stokes equations and a multigrid method. *Journal of Computational Physics*, vol. 48, pp. 387–411.

Haykin, S. (1999): *Neural Networks: A Comprehensive Foundation (second Edition)*, volume 842. Prentice Hall.

Ingber, M.; Chen, C.; Tanski, J. (2004): A mesh free approach using radial basis functions and parallel domain decomposition for solving three-dimensional

diffusion equations. *International Journal for Numerical Methods in Engineering*, vol. 60, pp. 2183–2201.

Kansa, E. (1990): Multiquadrics - A scattered data approximation scheme with applications to computational fluid-dynamics - I Surface approximations and partial derivative estimates. *Computers & Mathematics with Applications*, vol. 19(8-9), pp. 127–145.

Ling, L.; Opfer, R.; Schaback, R. (2006): Results on meshless collocation techniques. *Engineering Analysis with Boundary Elements*, vol. 30(4), pp. 247–253.

Mai-Duy, N.; Tran-Cong, T. (2001): Numerical solution of differential equations using multiquadric radial basis function networks. *Neural Networks*, vol. 14, pp. 185–199.

Mai-Duy, N.; Tran-Cong, T. (2008): A multidomain integrated-radial-basis-function collocation method for elliptic problems. *Numerical Methods for Partial Differential Equations*, vol. 24(5), pp. 1301–1320.

Mai-Duy, N.; Tran-Cong, T. (2011): Compact local integrated-RBF approximations for second-order elliptic differential problems. *Journal of Computational Physics*, vol. 230, pp. 4772–4794.

Smith, B. F.; Bjorstad, P. E.; Gropp, W. D. (1996): *Domain Decomposition Parallel Multilevel Methods for Elliptic Partial Differential Equations*. Cambridge University Press.

Tran, C.-D.; Phillips, D.; Tran-Cong, T. (2009): Computation of dilute polymer solution flows using bcf-rbfn based method and domain decomposition technique. *Korea-Australia Rheology Journal*, vol. 21, no. 1, pp. 1–12.

Tran-Canh, D.; Tran-Cong, T. (2004): Element-free simulation of dilute polymeric flows using brownian configuration fields. *Korea-Australia Rheology journal*, vol. 13(1), pp. 1–15.

Zerroukat, M.; Power, H.; Chen, C. (1998): A numerical method for heat transfer problems using collocation and radial basis functions. *Int. J. for Numer. Meth. in Engng.*, vol. 42, pp. 1263–1278.