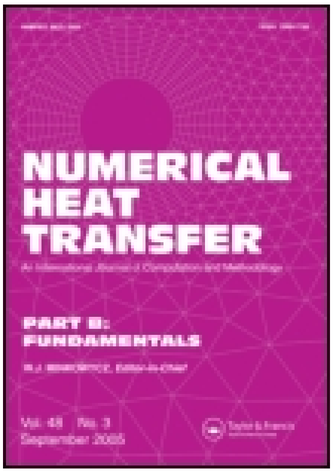


CrossMark
Click for updates

Numerical Heat Transfer, Part B: Fundamentals: An International Journal of Computation and Methodology
Publication details, including instructions for authors and subscription information:
http://www.tandfonline.com/loi/unhb20

A Parallel Multigrid Finite-Volume Solver on a Collocated Grid for Incompressible Navier-Stokes Equations

Pratanu Roy [a] , N. K. Anand [a] & Diego Donzis [a]
[a] Texas A&M University , College Station , Texas , USA

Published online: 26 Feb 2015.

To cite this article: Pratanu Roy , N. K. Anand & Diego Donzis (2015) A Parallel Multigrid Finite-Volume Solver on a Collocated Grid for Incompressible Navier-Stokes Equations, Numerical Heat Transfer, Part B: Fundamentals: An International Journal of Computation and Methodology, 67:5, 376-409, DOI: 10.1080/10407790.2014.985980

To link to this article: http://dx.doi.org/10.1080/10407790.2014.985980

Numerical Heat Transfer, Part B, 67: 376–409, 2015
Copyright © Taylor & Francis Group, LLC
ISSN: 1040-7790 print/1521-0626 online
DOI: 10.1080/10407790.2014.985980

A PARALLEL MULTIGRID FINITE-VOLUME SOLVER ON A COLLOCATED GRID FOR INCOMPRESSIBLE NAVIER-STOKES EQUATIONS

Pratanu Roy, N. K. Anand, and Diego Donzis
Texas A&M University, College Station, Texas, USA

Multigrid techniques are widely used to accelerate the convergence of iterative solvers. Serial multigrid solvers have been efficiently applied to a broad class of problems, including fluid flows governed by incompressible Navier-Stokes equations. With the recent advances in high-performance computing (HPC), there is an ever-increasing need for using multiple processors to solve computationally demanding problems. Thus, it is imperative that new algorithms be developed to run the multigrid solvers on parallel machines. In this work, we have developed a parallel finite-volume multigrid solver to simulate incompressible viscous flows in a collocated grid. The coarse-grid equations are derived from a pressure-based algorithm (SIMPLE). A domain decomposition technique is applied to parallelize the solver using a Message Passing Interface (MPI) library. The multigrid performance of the parallel solver has been tested on a lid-driven cavity flow. The scalability of the parallel code on both single- and multigrid solvers was tested and the characteristics were analyzed. A high-fidelity benchmark solution for lid-driven cavity flow problem in a 1,024 $\times$ 1,024 grid is presented for a range of Reynolds numbers. Parallel multigrid speedup as high as three orders of magnitude is achieved for low-Reynolds-number flows. The optimal multigrid efficiency is validated, i.e., the computational cost is shown to increase proportionally with the problem size.

1. INTRODUCTION

Multigrid methods are increasingly being used as an acceleration technique for the solution of Navier-Stokes equations. The basic idea of multigrid techniques lies in the principle of error smoothing of iterative solvers. A simple analogy can be the propagation of sound in air. The high-frequency component of sound, commonly known as "treble", dissipates quickly due to the

Received 11 July 2014; accepted 10 September 2014.
The authors are grateful to Professor Arun Srinivasa for his insightful lecture on multigrid techniques. The Texas A&M Supercomputing Facility (http://sc.tamu.edu) is gratefully acknowledged for providing computing resources useful in conducting the research reported in this article.
Address correspondence to N. K. Anand, Department of Mechanical Engineering, Texas A&M University, College Station, TX 77843-3123, USA. E-mail: nkanand@tamu.edu
Color versions of one or more of the figures in the article can be found online at www.tandfonline.com/unhb.

NOMENCLATURE

a	coefficient for discretized equation	α	underrelaxation factor
b	source term including contribution from transient formulation	Γ	diffusion coefficient
CDS	central difference scheme	δx, δy	diffusion length in x and y directions
C_p	Specific heat of the fluid, J/kg K	Δx, Δy	control-volume length in x and y directions
CV	Control volume		
f	Geometric interpolation factor	μ	dynamic viscosity of the fluid, Pa s
F	glow strength	ρ	density of the fluid, kg/m^3
FAS	Full approximation scheme	ϕ	generic flux variable
FMG	Full multigrid	Ω	convergence factor
H	$\sum_{\text{CVs}} a_{\text{nb}}\varphi_{\text{nb}}$		
I_{2h}^{h}	prolongation operator	**Subscripts**	
I_{h}^{2h}	restriction operator	h, $2h$	fine and coarse grid, respectively
k	thermal conductivity of the fluid W/m K	i, j	index variable
		nb	neighbor node
MG	Multigrid	u, v, p, mf	u velocity, v velocity, pressure, and mass flux, respectively
N	size of the problem		
Nx, Ny	number of control volumes in x and y directions	W,E, S,N	west, east, south, and north nodal points, respectively
p	pressure, Pa	w, e, s, n	west, east, south, and north faces, respectively
p'	pressure-correction variable		
P	Peclet number		
PLS	Power law scheme	**Superscripts**	
R	residual	$'$	correction to the fine-grid variable
Re	reynolds number	$''$	correction to the coarse-grid variable
S	source term		
SG	single grid	$*$	approximate solution
T	temperature, °C	$\wedge$	coarse-grid variable
u, v	u and v velocities in x and y directions, respectively, m/s	$\sim$	restricted quantities in coarse grid
		h, $2h$	fine and coarse grid-respectively
V	velocity vector, m/s	1	value from previous iteration

friction of sound waves with the air. However, the low-frequency component, known as "bass", is not easily damped by the air medium and can propagate through a long distance. A few animals, e.g., elephants, can detect these low-frequency components, which helps them to communicate with each other over a long distance. The computational grid of iterative solvers acts like the air medium and the solution error resembles the sound wave in this example. Similar to the sound wave, the solution errors of iterative solvers are composed of high-frequency components and low-frequency components. During the initial phase of iterations, the iterative solvers tend to smooth the high-frequency components (treble) of the solution errors quickly, resulting in a high convergence rate. However, the low-frequency parts of the solution error (bass) are not easily damped by the solver and propagate a long way before dying down completely. For this reason, the convergence rate of most iterative solvers decreases significantly after exhibiting fast convergence during the initial stage of iterations.

As most of the standard iterative methods tend to smooth out the high-frequency or oscillatory components of the error, they are commonly known as smoothers. After a few iterations of these smoothers, the high-frequency errors are quickly eliminated while the low-frequency errors remain unchanged. These low-frequency errors can be adequately represented on a coarse grid due to the inherent smoothness. However, the low-frequency errors in a fine grid are treated as high-frequency errors in the coarse grid, and hence iterations in a coarse grid can remove the low-frequency errors quickly. The basic idea of multigrid methods is to accelerate the convergence of iterative solvers by eliminating high- and low-frequency errors through repeated application of fine- and coarse-grid iterations.

As described by Achi Brandt [1], the golden rule of multigrid methods is the following: "*The amount of computational work should be proportional to the amount of real physical changes in the computed system.*" In a single grid, the computational work increases as $O(N^d)$, where N is the number of grid points and d is the dimension of the system. Thus, for a two-dimensional system, the computational work increases quadratically, whereas for a three-dimensional system, the increase is cubic in nature. The golden rule of Brandt predicts that an efficient multigrid method should be linearly dependent on the number of grid points N, i.e., it should exhibit an $O(N)$ increase in the computational work. One of the objectives of this study is to achieve this $O(N)$ increase in computation time for pressure-based incompressible Navier-Stokes solvers in multiple processors.

The pressure-based solvers of the incompressible Navier-Stokes equations rely on a predictor-corrector technique to reach the final solution. As the velocity and pressure variables are not known *a priori*, guessed values are used to solve the momentum equations to predict new velocities. Since the predicted velocities are not divergence-free as required by the incompressibility condition, a pressure-correction equation is solved to correct them. A number of researchers have investigated the application of multigrid methods in pressure-based incompressible Navier-Stokes solvers. Hortmann et al. [2] presented a finite-volume multigrid solution of the laminar natural-convection problem in a square cavity for different Rayleigh numbers. A high fidelity benchmark solution on grids as fine as 640×640 was provided. As desired, the computation time for multigrid solution in the collocated grid increased linearly as the grid was refined, whereas the single-grid computing time showed a quadratic increase. Lien and Leschziner [3] incorporated multigrid acceleration techniques into a nonorthogonal collocated finite-volume solver for laminar and turbulent flows. Compared to a single-grid solution, the computation speed-up for the multigrid solution was as high as 70 times for 2-D flows, and 10 times for 3-D flows. They concluded that the effectiveness of the multigrid scheme can be affected by the type of flow (i.e., laminar or turbulent, two-dimensional or three-dimensional), boundary conditions, turbulence-model equations, and Reynolds number. A multigrid solution for lid-driven cavity flow was presented in recent work by Kumar et al. [4]. Multigrid acceleration into the SIMPLEC algorithm was applied in a collocated grid to produce a benchmark solution on a uniform 513×513 grid. Pressure-based multigrid methods in staggered grids for recirculating and complex fluid flows were developed by Shyy and Sun [5] and Thakur et al. [6]. CPU time speed-up ratio for the multigrid method was as high as 25 for a second-order upwind convection scheme on a 81×81 grid. Multigrid procedures for three-dimensional laminar incompressible flows on nonorthogonal collocated grids were presented by Smith et al. [7]. They emphasized the consistent evaluation of coarse-grid mass fluxes after solving the momentum equations. CPU time speed-up of 4 to 5 was achieved for three-dimensional curved pipe flow. Yan and Thiele [8] developed a modified full multigrid method and applied it to the SIMPLE algorithm on collocated grids. They demonstrated a maximum speed-up of 25 for lid driven square cavity flow.

In all of the above-mentioned studies, segregated pressure-based solvers were used to solve the incompressible flow equations. Multigrid methods were also successfully applied to another type of pressure-based solvers, namely, coupled solvers, primarily developed by Vanka [9]. The initial attempts were limited to the staggered-grid approach [9, 10]. However, as the complexity of the problem and the computational power of the CPUs increased, multigrid procedures for coupled solvers in collocated grids were developed [11, 12].

With the rapid progress of computers in the past few decades, parallel multigrid methods have been developed to solve large-scale computational problems. At present, researchers are conducting simulations on massively parallel computers and are looking forward to the next-generation peta-scale supercomputers, which will enable them to run simulations on millions of processors. The question of whether parallel processors should be used in conjunction with multigrid methods is justified in the review on parallel multigrid methods by Jones and McCormick [13]. Durst and Schäfer [14] presented a parallel block-structured multigrid finite-volume solver for incompressible flows in complex geometries. Their results indicated that the combination of parallel computers and multigrid acceleration technique can produce improved computational performance. A parallel multigrid finite-volume solver for three-dimensional thermal convective flows was studied by Wang and Ferraro [15]. Good parallel scaling up to 128 processors was achieved for a 128^3 grid size. However, the effect of parallel processing elements on multigrid speed-up was not reported. In a recent survey on parallel multigrid solvers, Chow et al. [16] provided a brief account of the treatments of computationally efficient parallel multigrid methods. They discussed the parallel computation issues of standard multigrid algorithms such as spatial partitioning, parallel coarsening, and complexity, and coarse-grid solution strategy. In spite of the above-mentioned studies, a comprehensive investigation into the influence of parallel processing elements on a finite-volume multigrid solver in collocated grids has not been performed, which is the main objective of our work. In particular, we:

1. Describe a nonlinear multigrid algorithm with modified coarse-grid equations using the momentum interpolation method to solve incompressible Navier-Stokes equations in collocated grids, and
2. develop a parallel algorithm and investigate the effect of parallel processing elements on multigrid efficiency.

The article is organized as follows. Section 2 describes the discretization of governing equations using a finite-volume method. Section 3 presents the multigrid method and the derivation of the coarse-grid equations using the SIMPLE algorithm [17]. The code validation is presented in Section 4. The results are summarized and discussed in Section 5, and finally the conclusions are given in Section 6.

2. NUMERICAL METHODOLOGY

2.1. Governing Equations

The conservation equations of mass, momentum, and energy for an unsteady, incompressible flow can be expressed in the following vector form:

$$\nabla.\mathbf{V} = 0 \tag{1}$$

$$\frac{\partial \mathbf{V}}{\partial t} + \mathbf{V}.(\nabla \mathbf{V}) = -\frac{1}{\rho}\nabla p + \frac{1}{\rho}\nabla.(\mu \nabla \mathbf{V}) + S \tag{2}$$

$$\frac{\partial T}{\partial t} + \mathbf{V}.(\nabla T) = \frac{1}{\rho c_p}\nabla.(-k\nabla T) \tag{3}$$

Equation (1) is the continuity equation for an incompressible flow which ensures a divergence-free velocity field. Equation (2) is the so-called Navier-Stokes equations, which contain an unsteady term and a convective term on the left-hand side and a pressure gradient term and viscous diffusion term on the right-hand side. For constant properties fluid and laminar flow, the convective term is the only nonlinear term in the Navier-Stokes equation. Equation (3) is the thermal energy equation for constant thermal properties fluid, which is unsteady and linear in nature.

If we closely examine the momentum and energy equations, we can find that both of them share some common features, e.g., unsteadiness, convective terms, viscous terms, and source terms. As a result, we can express them in the following generic conservation form [17]:

$$\frac{\partial}{\partial t}(\rho\phi) + \nabla.(\rho \mathbf{V}\phi) = \nabla.(\Gamma\nabla\phi) + S \tag{4}$$

Here, Γ is the diffusion coefficient and S is the source or sink term. The generic variable ϕ can represent velocities (e.g., u and v in 2D) or temperature (T). The advantage of expressing the governing equations in generic conservation form is that, once we formulate a discretization method to solve this generic transport equation numerically, the method is readily applicable to all the governing equations.

Before describing the discretization, we note that the pressure variable appears only in the momentum equations. For a compressible flow, the equation of continuity carries a density term and the pressure is related to density by the equation of state. Consequently, the pressure can be calculated explicitly from the equation of state at each time step, and the numerical solution can be marched in time. However, in the case of incompressible flow, density is not present in the equation of continuity. As a result, there is no way that pressure can be calculated explicitly. This poses a serious difficulty in solving the incompressible Navier-Stokes equations. Fortunately, this problem was first overcome by Harlow and Welch [18] and later by Spalding and Patankar [19], so that at each time step, the momentum equations are solved iteratively by guessing the pressure and correcting the other primitive variables accordingly. This is the so-called semi-implicit pressure-linked equation

(SIMPLE) method, which is the basis of current pressure-based solvers for incompressible flow.

2.2. Finite-Volume Discretization

The finite-volume method has been widely applied to a variety of flow problems [20–27]. In this subsection, we will describe the finite-volume discretization scheme of the generic transport equation. While discretizing the momentum equations, a straightforward application of the central difference scheme to the pressure gradient term can result in a checkerboard-type pressure field, which might be an incorrect depiction of the reality. Similarly, the discretization of the first derivative of velocity terms in the continuity equation can lead us to a nonphysical wavy velocity field as a solution. One remedy for this problem is to discretize the pressure and velocity variables in a staggered grid, which was first proposed by Harlow and Welch [18]. However, the bookkeeping of all the variables in displaced locations of a staggered grid is cumbersome and difficult, especially for parallel algorithms, multigrid techniques, or curvilinear coordinates. The main reason for the failure of early attempts to use the collocated-grid approach was the delinking of the pressure and velocities at the node of a given control-volume when discretizing the pressure gradient terms in the momentum equations. Rhie and Chow [28] developed a special interpolation technique to calculate velocities at the control-volume faces of a collocated grid. The special interpolation technique is known as "momentum interpolation". Instead of linear interpolation at the control-volume interfaces, it uses discretized momentum equations of two adjacent control volumes to calculate the

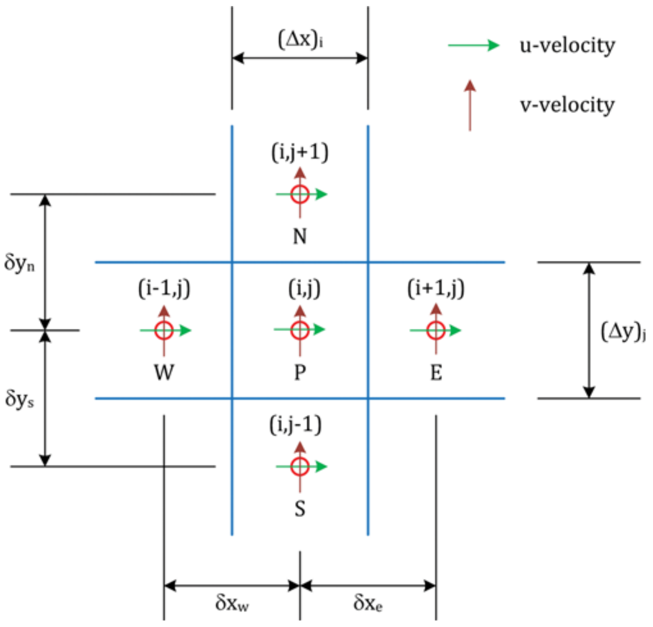


Figure 1. A typical control volume (CV) in a collocated grid approach.

face velocities. Figure 1 shows a typical two-dimensional control volume for the collocated-grid approach. The two-dimensional velocity vector **V** contains the x component u and the y component v, i.e., $\mathbf{V} = (u, v, 0)$. It can be seen from this figure that both pressure p and velocities u and v are computed at the same node.

Equation (4) can be rearranged in the following way:

$$\frac{\partial}{\partial t}(\rho\phi) = -\nabla.(\rho\mathbf{V}\phi) + \nabla.(\Gamma\nabla\phi) + S \tag{5}$$

The above equation can be integrated over the control volume:

$$\int_{\mathrm{CV}} \frac{\partial}{\partial t}(\rho\phi)\, d\mathcal{V} = -\int_{\mathrm{CV}} \nabla.(\rho\mathbf{V}\phi)\, d\mathcal{V} + \int_{\mathrm{CV}} \nabla.(\Gamma\nabla\phi)\, d\mathcal{V} + \int_{\mathrm{CV}} S\, d\mathcal{V} \tag{6}$$

If a backward Euler scheme is applied to discretize the transient term, then the two-dimensional discretized equation can be written in the following form [17]:

$$a_P\phi_P = a_E\phi_E + a_W\phi_W + a_N\phi_N + a_S\phi_S + b \tag{7}$$

where

$$a_E = D_e A(|P_e|) + \max(-F_e, 0) \tag{8a}$$

$$a_W = D_w A(|P_w|) + \max(F_w, 0) \tag{8b}$$

$$a_N = D_n A(|P_n|) + \max(-F_n, 0) \tag{8c}$$

$$a_S = D_s A(|P_s|) + \max(F_s, 0) \tag{8d}$$

$$b = S_c\, \Delta x\, \Delta y + \frac{\rho\, \Delta x\, \Delta y}{\Delta t} \tag{8e}$$

$$a_P = a_E + a_W + a_N + a_S - S_P\, \Delta x\, \Delta y + \frac{\rho\, \Delta x\, \Delta y}{\Delta t} \tag{8f}$$

Here, F_e, F_w, F_n, and F_s are the mass flow rates and D_e, D_w, D_n, and D_s are the diffusion conductances through the east, west, north, and south faces of the control volume, respectively.

If ρu is assumed to be uniform over the whole interface, we can write:

$$F_e = (\rho u)_e\, \Delta y, \quad F_w = (\rho u)_w\, \Delta y, \quad F_n = (\rho v)_n\, \Delta x, \quad F_s = (\rho v)_s\, \Delta y \tag{9}$$

The corresponding conductances are defined by

$$D_e = \frac{\Gamma_e\, \Delta y}{\delta x_e}, \quad D_w = \frac{\Gamma_w\, \Delta y}{\delta x_w}, \quad D_n = \frac{\Gamma_n\, \Delta x}{\delta y_n}, \quad D_s = \frac{\Gamma_s\, \Delta x}{\delta y_s} \tag{10}$$

P is the Peclet number, which is defined by the ratio of the strengths of convection (F) and diffusion (D). Thus, at the control-volume faces the corresponding

Peclet numbers are

$$P_e = \frac{F_e}{D_e} \quad P_w = \frac{F_w}{D_w} \quad P_n = \frac{F_n}{D_n} \quad P_s = \frac{F_s}{D_s} \tag{11}$$

The function $A(|P|)$ is calculated by using a power law scheme (PLS) or a central difference scheme (CDS) by the following formula:

$$PLS: \quad A(|P|) = \max\left(0, (1 - 0.1|P|)^{0.5}\right) \tag{12}$$

$$CDS: \quad A(|P|) = \max(0, (1 - 0.5|P|)) \tag{13}$$

Equation (7) is the discretized form of the Navier-Stokes equations where the value of the variable ϕ at point P (refer to Figure 1) is calculated from the neighboring ϕ variables at points E, W, N, and S. The coefficients a_E, a_W, a_N, and a_S carry the information of convection and diffusion from the neighboring points of P. The nonlinearity of the Navier-Stokes equations is also buried in these coefficients. Although the pressure gradient term is considered as a source term and included in the expression of b, it needs special treatment, which will be discussed in the subsequent sections.

Due to the strong nonlinear and coupled nature of the momentum equations, the above discretization method may not result in a converged solution. To avoid divergence, it is a common practice to use an underrelaxation factor in the discretized equation, which slows down the actual convergence rate but in the meantime ensures a stable solution [17, 29]. Incorporating an underrelaxation factor in Eq. (7) for a node (i, j) and rearranging, we get:

$$\phi_{i,j} = (1 - \alpha)\phi^{l}_{i,j} + \frac{\alpha}{a_{i,j}}[H_{i,j} + b_{i,j}] \tag{14}$$

where

$$H_{i,j} = a_{i+1,j}\phi_{i+1,j} + a_{i-1,j}\phi_{i-1,j} + a_{i,j+1}\phi_{i,j+1} + a_{i,j-1}\phi_{i,j-1} \tag{15}$$

Here, $\phi^{l}_{i,j}$ is the value of ϕ at node (i, j) from the previous iteration. Also, note that the neighboring points E, W, N, and S are replaced by the corresponding grid locations.

2.2.1. Calculation of CV face velocities. As stated earlier, in order to avoid the checkerboard pressure or velocity, a momentum interpolation method first proposed by Rhie and Chow [28] has been used to calculate the face velocities. In the case of the east face of cell (i, j), the interpolated face variable has the following form [30]:

$$\phi_{e_{i,j}} = (1 - \alpha)\phi^{l}_{e_{i,j}} + \frac{\alpha}{\bar{a}_{e_{i,j}}}[\bar{H}_{e_{i,j}} + \bar{b}_{e_{i,j}}] \tag{16}$$

The terms with overbars represent interpolated values at the east face of the control volume. These interpolated values can be calculated by the following linear interpolation formulas:

$$\frac{1}{\bar{a}_{e_{i,j}}} = \frac{f_e}{a_{i,j}} + \frac{(1-f_e)}{a_{i+1,j}} \tag{17a}$$

$$\bar{H}_{e_{i,j}} = f_e H_{i,j} + (1-f_e) H_{i+1,j} \tag{17b}$$

$$\bar{b}_{e_{i,j}} = f_e b_{i,j} + (1-f_e) b_{i+1,j} \tag{17c}$$

where the interpolation factor f_e is calculated by

$$f_e = \frac{\Delta x_{i+1}}{\Delta x_i + \Delta x_{i+1}} \tag{17d}$$

2.2.2. Correction of velocity and pressure fields by enforcing incompressibility condition. The calculated interface velocities in the previous subsection will not, in general, satisfy the continuity equation (1). Thus, the velocity and pressure fields have to be corrected in each iteration. In order to accomplish this, a pressure-correction equation is solved, which can be derived by expressing the velocities as a summation of a guessed value (u^*) and a correction term (u'), i.e., $u = u^* + u'$, and substituting them into the continuity equation. After rearrangement, the equation for the pressure correction p' takes the following form:

$$a^{pc}_{i,j} p' = a^{pc}_{i+1,j} p'_{i+1,j} + a^{pc}_{i-1,j} p'_{i-1,j} + a^{pc}_{i,j+1} p'_{i,j+1} + a^{pc}_{i,j-1} p'_{i,j-1} + b^{pc}_{i,j} \tag{18}$$

where

$$a^{pc}_{i+1,j} = \rho \frac{(\Delta y)_j}{\bar{a}_{e_{i,j}}} (\Delta y)_j \tag{19a}$$

$$a^{pc}_{i-1,j} = \rho \frac{(\Delta y)_j}{\bar{a}_{w_{i,j}}} (\Delta y)_j \tag{19b}$$

$$a^{pc}_{i,j+1} = \rho \frac{(\Delta x)_i}{\bar{a}_{n_{i,j}}} (\Delta x)_i \tag{19c}$$

$$a^{pc}_{i,j-1} = \rho \frac{(\Delta x)_i}{\bar{a}_{s_{i,j}}} (\Delta x)_i \tag{19d}$$

$$b^{pc}_{i,j} = \rho(u^*_w - u^*_e)(\Delta y)_j + \rho(v^*_s - v^*_n)(\Delta x)_i \tag{19e}$$

The a- coefficients with overbars are calculated from Eq. (17a). The source term for pressure [Eq. (19e)] represents the mass imbalance of the CV at node (i, j). Once the solution of the pressure-correction equation (18) is obtained, the interface velocities are corrected with the following expressions:

$$u_{e_{i,j}} = u^*_{e_{i,j}} + \frac{(\Delta y)_j}{\bar{a}_{e_{i,j}}} (p'_{i,j} - p'_{i+1,j}) \tag{20a}$$

$$u_{w_{i,j}} = u^*_{w_{i,j}} + \frac{(\Delta y)_j}{\bar{a}_{w_{i,j}}}(p'_{i-1,j} - p'_{i,j}) \tag{20b}$$

$$v_{n_{i,j}} = v^*_{n_{i,j}} + \frac{(\Delta x)_i}{\bar{a}_{n_{i,j}}}(p'_{i,j} - p'_{i,j+1}) \tag{20c}$$

$$v_{s_{i,j}} = v^*_{s_{i,j}} + \frac{(\Delta x)_i}{\bar{a}_{s_{i,j}}}(p'_{i,j-1} - p'_{i,j}) \tag{20d}$$

Similarly, the nodal velocities are corrected:

$$u_{i,j} = u^*_{i,j} + \frac{(\Delta y)_j}{a_{i,j}}(p'_w - p'_e) \tag{21a}$$

$$v_{i,j} = v^*_{i,j} + \frac{(\Delta x)_i}{a_{i,j}}(p'_s - p'_n) \tag{21b}$$

The nodal pressures are corrected by the following equation:

$$p_{i,j} = p^*_{i,j} + \alpha_p p'_{i,j} \tag{22}$$

where α_p is the underrelaxation factor for pressure.

The interface pressures p'_w, p'_e, p'_s, and p'_n are calculated by an interpolation of the nodal values. The interpolation technique is similar to Eq. (17*c*).

2.2.3. Convergence criteria. Once the pressure and velocities are corrected, the relative changes of these variables over two consecutive iterations are monitored to check convergence. The details of the calculation for convergence criteria can be found in [31]. In this article, the relative residuals for the generic variables (u, v, and T) are calculated by the following formula:

$$\begin{aligned} R_\phi = &\frac{1}{\sum\limits_{\text{CVs}} |a_{i,j}\phi_{i,j}|} \sum_{\text{CVs}} |a_{i,j}\phi_{i,j} - (a_{i+1,j}\phi_{i+1,j} + a_{i-1,j}\phi_{i-1,j} + \\ &a_{i,j+1}\phi_{i,j+1} + a_{i,j-1}\phi_{i,j-1} + b_{i,j})| \end{aligned} \tag{23}$$

For a converged pressure field, the only solution to the pressure-correction equation is a trivial solution. Thus, instead of checking the relative change in the values of the pressure-correction variable, we check the change in residual for mass imbalance b_{pc} [refer to Eq. (19e)]. The relative residual for mass conservation is:

$$R_{mf} = \frac{\sum\limits_{\text{CVs}} |\rho(u_w - u_e)(\Delta y)_j + \rho(v_s - v_n)(\Delta x)_i|}{\rho u_{\text{ref}} L_{\text{ref}}} \tag{24}$$

where u_{ref} is a characteristic or reference velocity and L_{ref} is a reference length, the values of which depend on the geometry and boundary conditions.

3. MULTIGRID ALGORITHM

The basic principle of the multigrid algorithm is to eliminate the approximate solution error efficiently by alternatively iterating on fine and coarse grids. The first step of this algorithm is to derive a set of coarse-grid equations from the existing fine-grid discretized equations. It is important to note that, in the coarse grid, the original discretized equations for the SIMPLE algorithm are not solved, rather correction equations are solved. Due to the nonlinear nature of the governing equations, the correction equations are derived from the discretized equations for the SIMPLE algorithm. This process of obtaining the correction equations is known as the full approximation scheme (FAS). After deriving the coarse-grid correction equations with the FAS, suitable grid transfer operators are chosen to interchange the coarse-and fine-grid values without losing essential information. Transferring the fine-grid information to the coarse-grid is known as *restriction*. On the other hand, transferring the coarse-grid information into the fine grid is called *prolongation*. In the next subsections, these essential multigrid elements will be described.

3.1. Coarse-Grid Equations

The system of linear equations [Eq. (7)] can be expressed in the following form:

$$A\phi = b \tag{25}$$

Here, ϕ is the exact solution of the problem, A is the coefficient matrix, and b is the source term. A is composed of a linear part, $L(\phi)$ (pressure difference term) and a nonlinear part, $N(\phi)$ (convection-diffusion term). Thus, we can write:

$$N\phi + L\phi = b \tag{26}$$

After a few outer iterations (ν_1), the approximate solution ϕ^* satisfies the following equation:

$$N^*\phi^* + L\phi^* = b^* + R \tag{27}$$

where R is the residual. By subtracting (27) from (26), we get:

$$N\phi - N^*\phi^* + L\phi' = b - b^* - R \tag{28}$$

Here, ϕ' is the difference between the exact and approximate solution, i.e., $\phi' = \phi - \phi^*$. The coarse-grid equation is assembled from Eq. (28) by restricting the fine-grid variables and using them as a fixed source term on the coarse grid:

$$\widehat{N}\widehat{\phi} = \widehat{b} - \widehat{L}(\phi') + \underline{\widetilde{N}\widetilde{\phi} - \widetilde{b} - \widetilde{R}} \tag{29}$$

The coarse-grid variables are denoted by a caret '^', and the restricted quantities are denoted by tilde '~'. The underlined terms are kept frozen during the coarse-grid iterations. Thus, Eq. (29) estimates the difference $(\widehat{\phi} - \widetilde{\phi})$ that is driven by residual R. This difference $(\widehat{\phi} - \widetilde{\phi})$ is known as the coarse grid correction. In view of Eq. (29), the coarse-grid correction equations for u velocity and v velocity are as follows:

$$\widehat{a}_P\widehat{u}_P = \sum_{\text{nb}} \widehat{a}_{\text{nb}}\widehat{u}_{\text{nb}} + \Delta y(p'_w - p'_e) + \widehat{S}_u + \underline{\widetilde{a}_P\widetilde{u}_P - \sum_{\text{nb}} \widetilde{a}_{\text{nb}}\widetilde{u}_{\text{nb}} - \widetilde{S}_u - \widetilde{R}_u} \tag{30}$$

$$\widehat{a}_P\widehat{v}_P = \sum_{\text{nb}} \widehat{a}_{\text{nb}}\widehat{v}_{\text{nb}} + \Delta x(p'_s - p'_n) + \widehat{S}_v + \underline{\widetilde{a}_P\widetilde{v}_P - \sum_{\text{nb}} \widetilde{a}_{\text{nb}}\widetilde{v}_{\text{nb}} - \widetilde{S}_v - \widetilde{R}_v} \tag{31}$$

Denoting the underlined terms with $\tilde{f}_{\text{u}}$ and $\tilde{f}_{\text{v}}$, we can compute the face velocities similarly:

$$(\widehat{a}_P)_e\widehat{u}_e = \left(\sum_{\text{nb}} \widehat{a}_{\text{nb}}\widehat{u}_{\text{nb}}\right)_e + (\Delta y)_e(p'_P - p'_E) + (\widehat{S}_u)_e + (\widetilde{f}_u)_e \tag{32}$$

$$(\widehat{a}_P)_n\widehat{v}_n = \left(\sum_{\text{nb}} \widehat{a}_{\text{nb}}\widehat{v}_{\text{nb}}\right)_n + (\Delta x)_n(p'_P - p'_N) + (\widehat{S}_v)_n + (\widetilde{f}_v)_n \tag{33}$$

The coarse-grid steps for the SIMPLE algorithm can be summarized as follows.

1. Start with the guess:

$$\widehat{u} = \widetilde{u}, \quad \widehat{v} = \widetilde{v}, \quad p' = 0 \tag{34}$$

2. A number of relaxation sweeps on the coarse-grid momentum equations are performed to get approximate solutions $\widehat{u}^*$ and $\widehat{v}^*$.
3. Face velocities are calculated with the modified momentum interpolation method:

$$\widehat{u}_e^* = (1 - \alpha_u)\widehat{u}_e^l + \frac{\alpha_u}{(\widehat{a}_P)_e}\left[\left(\sum_{\text{nb}} \widehat{a}_{\text{nb}}\widehat{u}_{\text{nb}}\right)_e + (\Delta y)_e(p'_P - p'_E) + (\widehat{S}_u)_e + (\widetilde{f}_u)_e\right] \tag{35}$$

$$\widehat{v}_n^* = (1 - \alpha_v)\widehat{v}_n^l + \frac{\alpha_v}{(\widehat{a}_P)_n}\left[\left(\sum_{\text{nb}} \widehat{a}_{\text{nb}}\widehat{v}_{\text{nb}}\right)_n + (\Delta x)_n(p'_P - p'_N) + (\widehat{S}_v)_n + (\widetilde{f}_v)_n\right] \tag{36}$$

4. An equation for correction to pressure correction is solved:

$$a_P p''_P = \sum_{\text{nb}} a_{\text{nb}} p_{\text{nb}}'' + b''_p \tag{37}$$

$$b''_p = \rho[(\widehat{u}_w^* - \widetilde{u}_w) - (\widehat{u}_e^* - \widetilde{u})](\Delta y) + \rho[(\widehat{v}_s^* - \widetilde{v}_s) - (\widehat{v}_n^* - \widetilde{v}_n)](\Delta x) \tag{38}$$

5. Nodal and face velocities are corrected by using p'':

$$\widehat{u}_e = \widehat{u}_e^* + \frac{(\Delta y)}{(\widehat{a}_P)_e}(p''_P - p''_E) \tag{39}$$

$$\widehat{v}_n = \widehat{v}_n^* + \frac{(\Delta x)}{(\widehat{a}_P)_n}(p''_P - p''_N) \quad (40)$$

$$\widehat{u}_P = \widehat{u}_P^* + \frac{(\Delta y)}{(\widehat{a}_P)}(p''_w - p''_e) \quad (41)$$

$$\widehat{v}_P = \widehat{v}_P^* + \frac{(\Delta x)}{(\widehat{a}_P)}(p''_s - p''_n) \quad (42)$$

Similarly, the pressure-correction variables are updated through

$$p'_P = p'_P + \alpha_p p''_P \quad (43)$$

6. Return to step 2 to complete one outer relaxation sweep.

The coarse-grid sequence may look similar to the ones described by Lien and Leschziner [3]. However, there is a subtle difference which lies in the expression of coarse-grid face velocities. In their coarse-grid formulations, the cell-face velocities become equal to the average of the node velocities when the pressure-correction terms are zero. This is not the case in the present coarse-grid formulation.

3.2. Grid Transfer Operators

In order to achieve favorable $O(N)$ convergence behavior, as predicted by Brandt [1], a multigrid method should satisfy the *smoothing property* and the *approximation property* [32]. The smoothing property is responsible for the rapid elimination of the oscillatory or high-frequency modes of the error, leaving the smooth modes [33]. Most iterative solvers, including the one used here (line-by-line method), exhibit the smoothing property. It should be noted that, although in single-grid methods this smoothing property can be seen as a serious drawback, for multigrid methods this smoothing property is desirable. The approximation property requires that $m_P + m_R > M$, where m_P and m_R are the order of accuracy plus one for the prolongation and restriction schemes, respectively, and M is the highest derivative of the variable under consideration in the governing equations. In the incompressible Navier-Stokes equations, $M = 1$ for pressure, and $M = 2$ for velocity components. Thus, first-order-accurate interpolation schemes (e.g., bilinear interpolation for 2D and trilinear interpolation for 3D) should be sufficient to satisfy the approximation property.

In the collocated grid arrangements, the variables are located at the center of the control volumes (CVs). In this type of cell-centered discretization, the coarse-grid points do not coincide with the fine-grid points (as shown in Figure 2). As a result, special care is needed for grid transfer operations, i.e., for restriction and prolongation. A four-point average was used for restriction and a bilinear interpolation was used for prolongation [3, 34]. During the restriction, the coarse-grid variables are calculated as an area-weighted average of the surrounding fine-grid variables. Mathematically, the fine to coarse-grid operation can be expressed as:

$$\phi^{2h} = I_h^{2h} \begin{bmatrix} \phi_1 \\ \phi_2 \\ \phi_3 \\ \phi_4 \end{bmatrix}^h \quad (44)$$

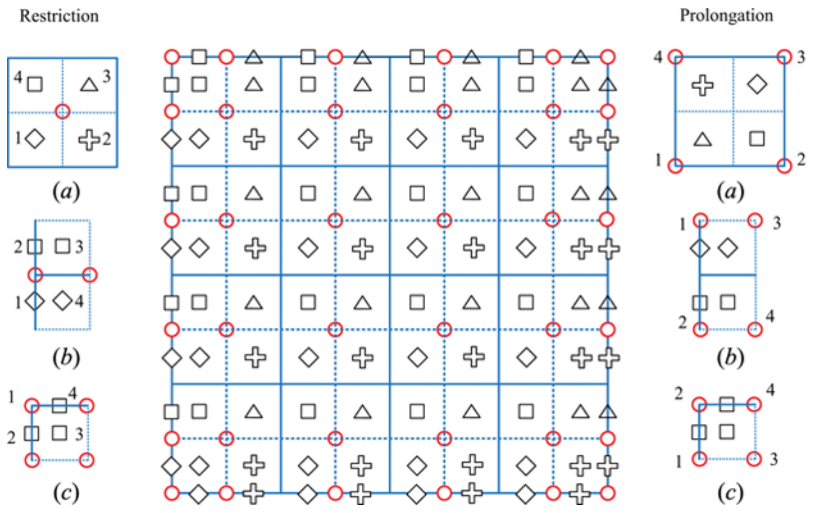


Figure 2. Finite-volume collocated grid with symbols explaining the grid transfer operators. Red circles denote the coarse-grid points and all other symbols denote the fine-grid points. For restriction and prolongation; (a) interior cells; (b) west boundary cells; (c) north-west cells.

For inner regions (excluding the boundary cells), the restriction operator is:

$$I_h^{2h} = \frac{1}{4}[1 \quad 1 \quad 1 \quad 1] \tag{45}$$

For west boundary cells,

$$I_h^{2h} = \frac{1}{4}[2 \quad 2 \quad 0 \quad 0] \tag{46}$$

For north-east corner cells,

$$I_h^{2h} = \frac{1}{4}[4 \quad 0 \quad 0 \quad 0] \tag{47}$$

The residuals are restricted differently from the variables. In the finite-volume formulations, the residuals represent flux imbalances through the CV faces. Therefore, the coarse-grid residual must be computed as the sum of the surrounding fine-grid residuals. The resulting prolongation operator for residuals is

$$I_h^{2h} = [1 \quad 1 \quad 1 \quad 1] \tag{48}$$

The fine-grid variables are corrected in the following way:

$$\phi_h^{\text{new}} = \phi_h^{\text{old}} + \alpha_{MG}\delta\phi_h = \phi_h^{\text{old}} + \alpha_{MG}I_{2h}^{h}(\hat{\phi}_{2h} - \widetilde{\phi}_{2h}) \tag{49}$$

where α_{MG} is an underrelaxation factor. The coarse-grid to fine-grid operations, i.e., prolongation, can be expressed as

$$\begin{bmatrix} \delta\phi_1 \\ \delta\phi_2 \\ \delta\phi_3 \\ \delta\phi_4 \end{bmatrix}^h = I_{2h}^h \begin{bmatrix} \delta\phi_1 \\ \delta\phi_2 \\ \delta\phi_3 \\ \delta\phi_4 \end{bmatrix}^{2h} \tag{50}$$

The prolongation operators are as follows:
For the interior region:

$$I_{2h}^h = \frac{1}{16} \begin{bmatrix} 9 & 3 & 3 & 1 \\ 3 & 9 & 1 & 3 \\ 3 & 1 & 9 & 3 \\ & 3 & 3 & 9 \end{bmatrix} \tag{51}$$

For west boundary cells:

$$I_{2h}^h = \frac{1}{8} \begin{bmatrix} 0 & 0 & 6 & 2 \\ 0 & 0 & 2 & 6 \\ 3 & 1 & 3 & 1 \\ 1 & 3 & 1 & 3 \end{bmatrix} \tag{52}$$

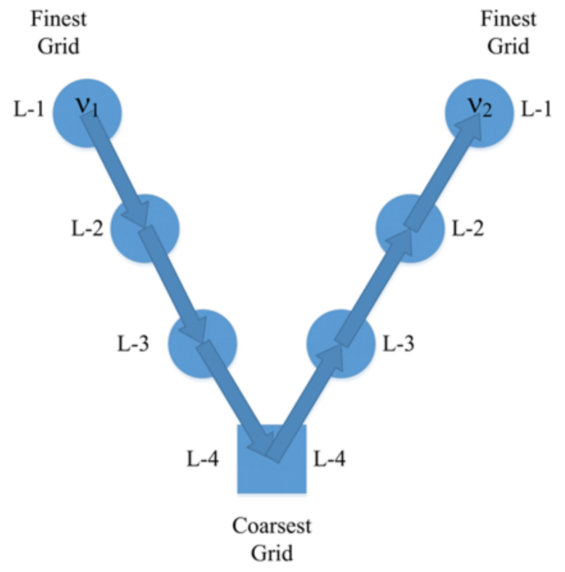
For north-west corner cells:

$$I_{2h}^h = \frac{1}{4} \begin{bmatrix} 2 & 2 & 0 & 0 \\ 0 & 0 & 0 & 4 \\ 1 & 1 & 1 & 1 \\ 0 & 2 & 0 & 2 \end{bmatrix} \tag{53}$$

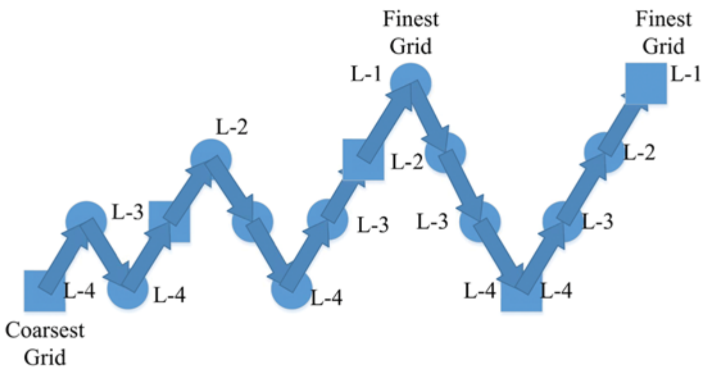
3.3. Multigrid Cycles

Once the coarse-grid correction equations are derived and suitable grid transfer operators are chosen, the multigrid algorithm can be applied by a recursive operation of smoothing and intergrid transfer. This recursive process is commonly known as a multigrid cycle (MGC). The most basic cycle is called a V-cycle. In a V-cycle, the fine-grid errors are restricted to the coarse grid. After a few iterations, namely, pre-smoothing operations, the coarse-grid errors are restricted to the next coarser level. This process is continued until the coarsest grid level is reached, where the computational work for solving the residual equation is so cheap that a converged solution to the discretized equation can easily be calculated, by using either a direct solver or an iterative solver. Once the converged solution is found in the coarsest grid, the error is prolongated into the next fine grid and the old error on the fine grid is corrected using this prolongated value. After a few post-smoothing operations on the corrected error, it is further prolongated to the next finer level, and the process is continued until the finest grid level is reached. A schematic of the V-cycle is shown in Figure 3a. The down arrow indicates a restriction, whereas the up arrow indicates a prolongation. The circles at each level implies smoothing operations to get an approximate solution. The square box at the coarsest level signifies a converged solution of the residual equation.

There are a number of variations to the V-cycle, of which the full multigrid (FMG) V-cycle is the most popular one [2, 3]. Instead of starting the MG V-cycle on a given fine grid, we can first apply it on a very coarse grid, where a few number of cycles would produce converged solution. This converged solution on the coarse grid then can be prolongated on the next fine-grid to provide a good initial guess to the solution of the next level. If this process is applied consecutively, then we will already have a very good initial guess when the finest grid level is reached. Thus, a faster convergence rate than a V-cycle can be achieved by using an FMG V-cycle.



(*a*) Multigrid V cycle with ν_1 pre-smoothing and ν_2 post-smoothing iterations



(*b*) Full multigrid (FMG) V-Cycle with 4 levels

Figure 3. Multigrid cycles. The circles denote approximate solutions and the squares denote converged solutions.

Figure 3b presents a schematic of the FMG V-cycle. In terms of computational complexity, the MG V-cycle has a cost of $O(N)$, where N is the number of unknown variables in the fine-grid system, as described earlier. However, the initial guess for the pressure/velocity variables introduce an error of $O(1)$, and it takes $O(N \log N)$ V-cycle operations to reduce the error below the convergence criteria [13]. Although this is a large reduction in the computational work from $O(N^d)$ operations in a single grid, it does not exactly satisfy the "golden rule" of multigrid [1, 35]. Fortunately, this limitation is overcome by the FMG V-cycle, which can effectively reduce the number of operations to $O(N)$ by successively introducing a good initial guess on each fine grid.

As discussed above, during the FMG V-cycle, the converged solution in the coarse grid is prolongated to the next fine-grid. The prolongation can be performed in several ways, of which the bilinear interpolation described in the previous subsection and the upwind-based interpolation are very common. In this work, a bilinear interpolation is used for node velocities and pressures. A momentum interpolation is applied to obtain the face velocities. It is observed that the momentum interpolation method results in better initial guesses for the fine-grid iterations than the upwind-based interpolation scheme.

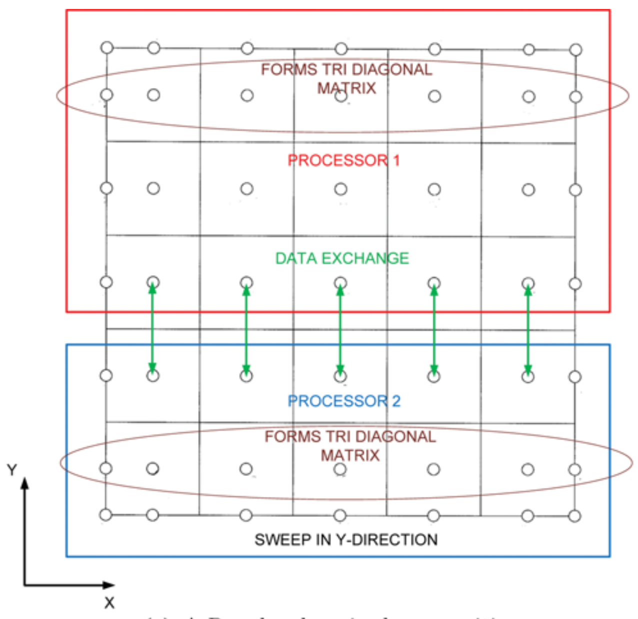
3.4. Parallel Implementation

In order to parallelize the finite volume algorithm described in the previous subsections, two options have been considered:

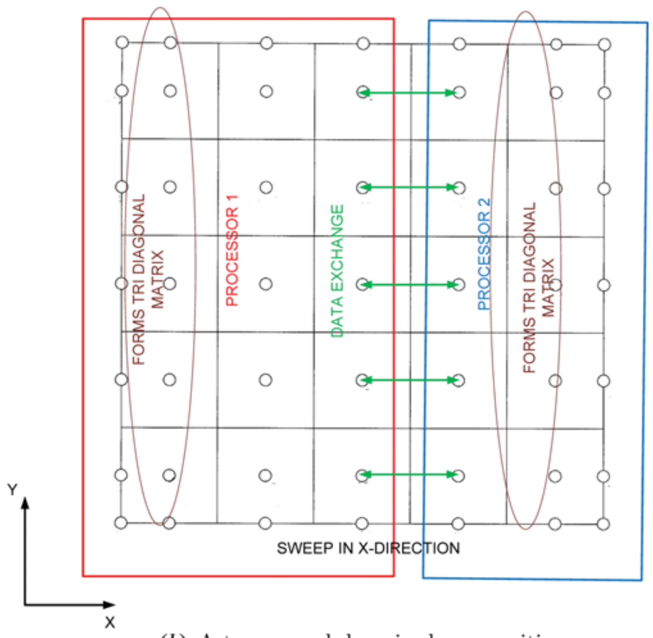
1. Shared-memory parallelization
2. Distributed-memory parallelization

In a shared-memory multiprocessor, variables are stored in a shared address space and each processor can access the same variables with relatively low overheads. As a result, parallel extension of a serial code in a shared-memory multiprocessor is straightforward. OpenMP (OpenMultiProcessing) is a standard Application Programming Interface (API) for shared memory parallelization. Saldana et al. [36] used OpenMP to parallelize a SIMPLE-based finite-volume algorithm which is similar to the algorithm in a staggered grid. Although shared-memory parallelization is easy to implement, it becomes increasingly difficult and expensive to increase the number of processors arbitrarily in a shared-memory system.

On the other hand, in a distributed-memory multiprocessor, each processor has its own memory module to store the variables, and the modules are connected through a high-speed communication network. There is no concept of global address space, and if one processor needs to access data from another processor, they have to communicate explicitly. This makes the programmer responsible for maintaining all the details of the communication in a distributed-memory architecture. Message Passing Interface (MPI) is a standard API for interprocess communication in distributed-memory multiprocessors. Although great care is needed to perform parallelization with MPI, very efficient and scalable programs can be written using this message-passing paradigm in distributed-memory multiprocessors [37, 38]. In this article, we will describe the parallelization of a finite-volume algorithm in a structured grid using MPI. For an analysis of the parallel finite-volume method in unstructured grid, readers are referred to the recent work of Zhao and Zhang [39].



(*a*) A Regular domain decomposition



(*b*) A transposed domain decomposition

Figure 4. Domain decomposition of a 5×5 CV grid.

3.4.1. Domain decomposition. The first step toward parallelization is to decompose the computational domain into a number of subdomains and assign them to different processors. Figure 4*a* presents how the domain is decomposed in two processors. Each processor exchanges data at the corresponding subdomain boundaries. To accommodate this exchange of data at the boundaries, each boundary node has an extra ghost node for each subdomain, and they overlap each other.

In the serial FVM code, the discretized algebraic equations are solved by a line-by-line method, which is a combination of the iterative Gauss-Seidel method and the tri-diagonal matrix algorithm (TDMA). The use of the underrelaxation factor makes the Gauss-Seidel method effectively a weighted Jacobi iteration method. In a two-dimensional line-by-line method, a tri-diagonal matrix is formed along a chosen line (say in the x direction), and it is solved directly by the TDMA. Then a sweeping is performed along the other direction (say, in the y direction) for each line. By solving the tri-diagonal matrix system along a line, the boundary conditions are transmitted into the interior points. After one whole sweep in a fixed direction, we can alternate the direction in which TDMA sweeping is employed. In this process, we can quickly bring the information from all the boundaries into the interior of the domain [17]. The switching of directions in a line-by-line method can accelerate the numerical solution, especially for elliptic problems with Dirichlet boundary conditions. For a parallel code, this is difficult to employ, since the subdomain has to be transposed alternatively in order to employ the line-by-line TDMA in both directions (Figure 4b). MPI provides a useful collective communication operation (*MPI_Alltoall*) to perform a domain transpose. To properly conduct the domain transpose in different processors, three steps are needed: (1) a local in-core transpose, (2) an *MPI_Alltoall* operation, and (3) another local in-core transpose. However, the iterative solver for momentum equations will call the *MPI_Alltoall* function every time the domain needs to be transposed. Since it may take thousands of iterations to get a converged solution at each time step, transposing the domain can greatly increase the communication overhead, which may adversely affect the parallel performance of the code. Another limitation of domain transpose is that the number of grid points must be a multiple of the number of processors. This problem can be overcome by using an *MPI_Alltoallv* function which allows different processors to send a different amount of data and provide displacements for the input and output messages, increasing signifcantly the complexity of the code. Taking all these issues into consideration, domain transpose was avoided and sweeping along only one direction was performed.

3.4.2. Parallel algorithm. The following sequence of instructions is carried out by the processes:

1. Read the input data with the root processor (processor 0).
2. Broadcast the input data from the root processor to all other processors.
3. Perform domain decomposition, i.e., map the domain into different processors.
4. Calculate all the relevant thermo-fluid properties, diffusion lengths, and control-volume lengths.
5. Initialize the data in all processors. Store guessed values for pressure and interface velocities.

6. Impose boundary conditions.
7. At each multigrid level, execute the following steps:

   a. Compute the momentum coefficients for u and v velocities using Eqs. (8) through (13).
   b. Solve the tri-diagonal matrix system of equations for each line in the x direction and traverse in the y direction. Exchange the updated values of u and v at the subdomain boundaries.
   c. Exchange the values of a_P, H_u, and H_v at the subdomain boundary nodes. These values will be used in the subsequent calculations for momentum interpolation coefficients.
   d. Calculate the face velocities at each control volume using the momentum interpolation technique in Eq. (16) in a fine grid or Eq. (35) and (36) in a coarse grid.
   e. Calculate the coefficients of the pressure-correction equation using Eq. (19) Use Eq. (38) for coarse-grid mass imbalance calculation.
   f. Solve the pressure-correction equation [(18) in a fine grid or (37) in a coarse grid] using an iterative solver (e.g., the line-by-line method).
   g. Correct the interface velocities using nodal pressure-correction values by applying Eqs. (20*a*)–(20*d*).
   h. Correct the pressure variables at each node using Eq. (22).
   i. Correct the nodal velocities using Eq. (21*a*) and (21*b*). To evaluate these equations we will need to interpolate the pressure corrections at interfaces, which is similar to Eq. (17*c*).
   j. Solve for other scalar variables such as temperature, kinetic energy, and the like.
   k. Calculate the relative residuals using Eq. (23) and (24). Perform a collective reduction operation using *MPI_Allreduce*[1] to get the summation of the relative residuals.

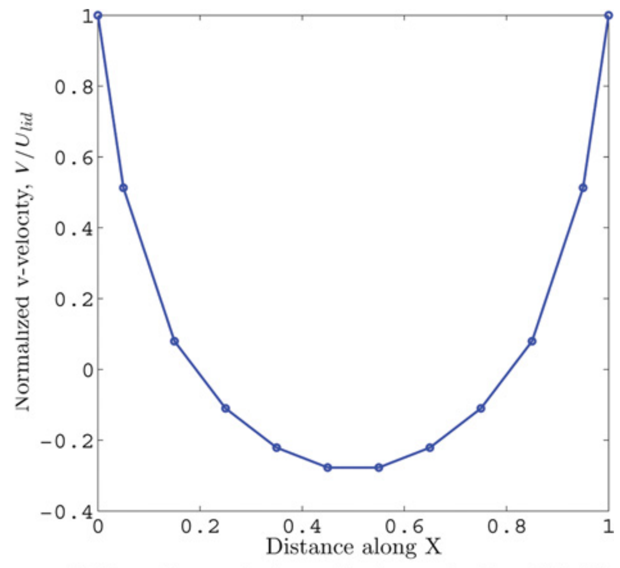
8. Check for convergence. If the solution is converged, prolongate the velocities and pressures into the next fine-level and go to step 7. If the solution is converged at the finest multigrid level, go to step 9.
9. Store the velocities from the previous time step. If instructed, write the output into a file. Proceed to the next time step.

4. CODE VALIDATION

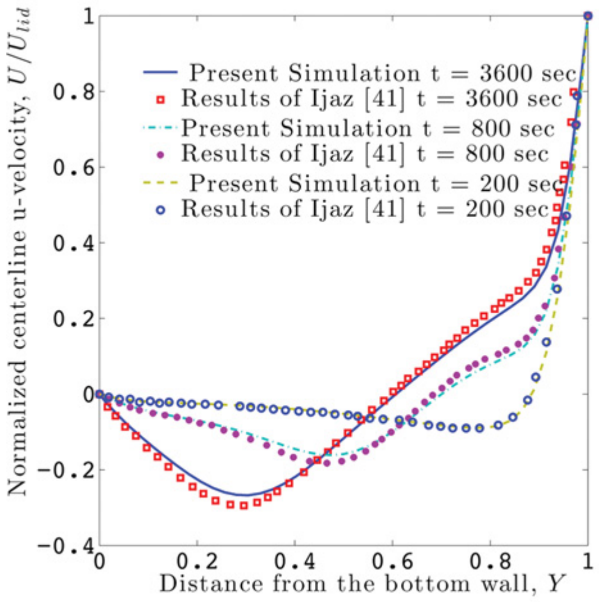
Lid-driven cavity flow has been widely used as a benchmark problem to validate and verify incompressible Navier-Stokes solvers. The simplicity in geometry coupled with rich flow physics and the ease of applying boundary conditions make it a suitable candidate for benchmarking the code. The parallel finite-volume code has been tested and validated in several steps. First, the lid-driven cavity flow

[1]*MPI_Allreduce* combines data from all processes and distributes the reduced data back to all processes [40].

problem was solved using 10×10 control volumes. Both left and right plates were maintained at the same velocity, and no-slip boundary conditions were applied to all boundaries. The symmetry of the steady-state solution in the u and v velocities were checked. Figure 5*a* presents the normalized centerline v velocity along the x



(*a*) Centerline v-velocity profile along x for Re = 100 with both side plates moving at the same velocity



(*b*) Centerline u-velocity profile along y at different time instants with the upper plate moving at Re = 400

Figure 5. Code validation for 2-D lid-driven cavity flows.

axis. The symmetry of the solution was confirmed up to six decimal places. Then the number of processors were increased to check that the results are processor-independent.

Additionally, the unsteady solution of a lid-driven cavity flow with the upper plate moving at a Reynolds number of 400 has been validated against the simulation results of Ijaz [41]. From Figure 5*b*, it can be observed that the present simulation results are in good agreement with the results of Ijaz [41].

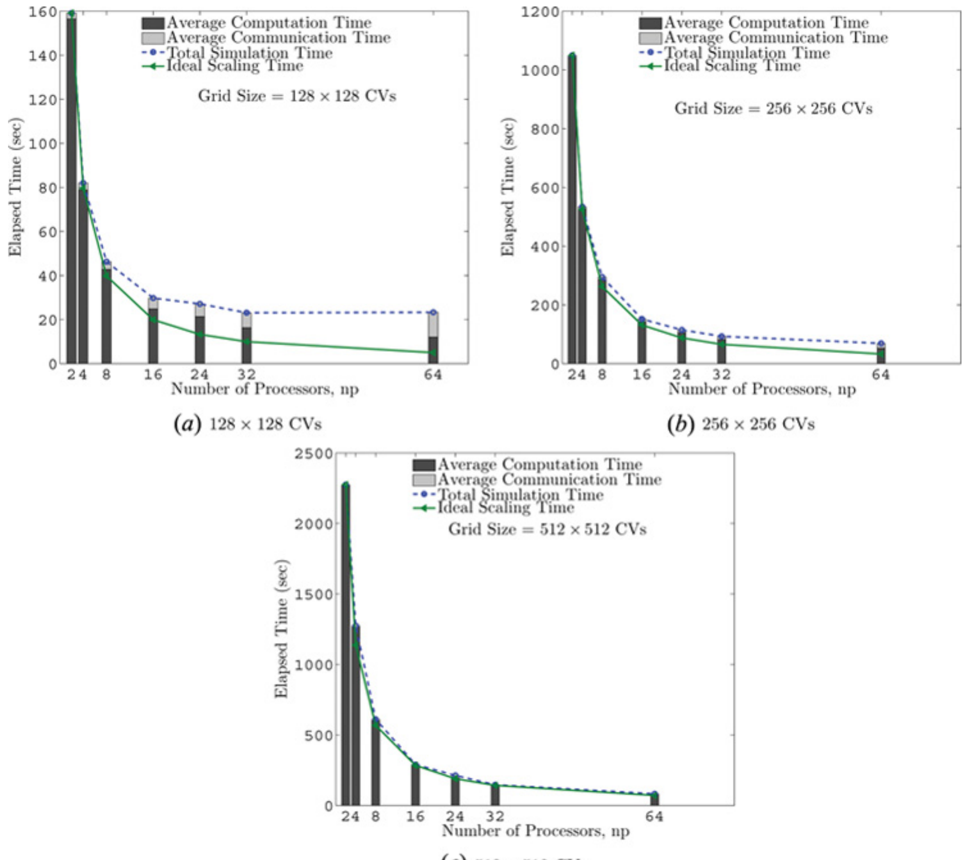
5. RESULTS AND DISCUSSION

5.1. Parallel Performance on a Single Grid

Before studying the performance of the parallel multigrid solver, a performance analysis on a single grid was conducted. Parallel scalability was assessed by performing strong scaling studies, where the number of processors is increased for a fixed problem size. In ideal scaling, the simulation time is inversely proportional to the number of processors. Departures from this ideal scaling indicate overheads associated with parallelism, such as interprocessor communication, which leads to a degradation in the efficiency of the code. In order to test the strong scaling, three grid sizes were chosen: 128×128 CVs, 256×256 CVs, and 512×512 CVs. For each case, the total CPU time, the average computation time, and the average communication time were calculated. The simulations were conducted using up to 64 processors on the Texas A&M EOS supercomputer, which is an IBM Linux cluster with 8-core (Nehalem) nodes connected through a high-speed infiniband fabric.

Figure 6a shows the comparison of communication and computation time for a grid size of 128×128 CVs. For this coarse grid size, good scalability is found only up to 8 processors. As the number of processors increases, the communication time increases relative to the computation time, which decreases in a per-core basis, resulting in a reduction of the parallel performance. Figure 6b shows the comparison of simulation time for a grid size of 256×256 CVs. In this case, better scalability can be observed than in the previous case, as the percentage computation time is much higher than the percentage communication time. The scalability increases up to 16 processors, and performance degrades after the choke point as the communication overhead becomes dominant. From Figure 6c, it can be observed that strong scalability can be achieved up to 64 processors for a grid size of 512×512 CVs.

Figure 7 presents the comparison of parallel speed-up for all three cases. The parallel speed-up indicates how much faster the code runs using multiple processors as opposed to the single processor. It is evident from this graph that superior speedup can be achieved for finer grid size. An ideal speed-up curve is a starlight line with a unit slope, passing through the origin. For the grid size of 512×512 CVs, almost linear speed-up can be observed. The reason is that for finer grid size, the ratio of computation time to communication time is very high. Although the communication time increases with increase of the number of processors, the portion of communication overhead in comparison to the computation time is low. Therefore, as long as the communication time is a small percentage of the total execution time, good speed-up is more likely to be achieved. In Figure 7, it can also be seen that in the case of 24 processors, the speed-up deviated a bit from the regular trend of the



(*a*) 128 × 128 CVs (*b*) 256 × 256 CVs (*c*) 512 × 512 CVs

Figure 6. Strong scaling for lid-driven cavity flow with top plate moving (a) 128 × 128 CVs; (b) 256 × 256 CVs; (c) 512 × 512 CVs (Re = 10,000).

curves. This might be due to the load imbalance among the processors. In the finite-volume formulation, the total number of grid points is two more than the number of control volumes. So, all the grid points may not be equally distributed among the processors, which may result in a load imbalance situation.

5.2. Multigrid Performance on Parallel Machines

In order to assess the multigrid performance of the parallel code, steady two-dimensional lid-driven cavity flow simulations were conducted for a range of Reynolds number (Re = 100–7,500). The simulation parameters, including the underrelaxation factors and the number of coarse-grid iterations, are tabulated in Table 1. It should be noted that, for high Reynolds numbers (Re > 6, 000), the flow becomes inherently unsteady, three-dimensional, and turbulent [42]. Thus, a steady state two-dimensional laminar solution of high-Reynolds-number lid-driven cavity flow might not be physically accurate, but has been reported in the literature for benchmarking purposes [43, 44]. Since in the coarsest level the number of grid points

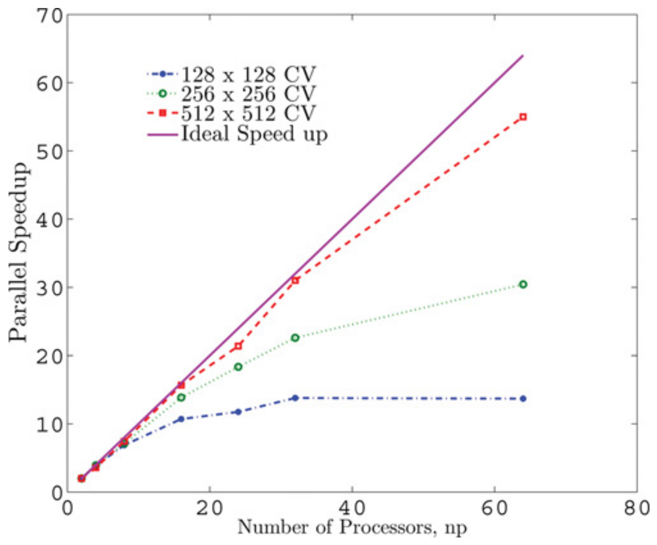


Figure 7. Comparison of parallel speed-up with different grid sizes for lid-driven cavity flow (Re = 10,000).

is very low, it is not possible to increase the number of processors arbitrarily. Otherwise, a degradation of parallel performance would happen and the number of processors might well exceed the number of coarse grid points. Therefore, in the present study, the maximum number of processors for parallel multigrid code has been limited to 8. However, there are ways to overcome this limitation. Interested readers are directed to Chow et al. [16] for a survey of specialized parallel multigrid methods.

From this point on, the single grid will be referred to by SG and the multigrid by MG. Two parameters are commonly used as MG performance indicators: CPU time and work unit (WU). CPU time is defined as the total execution time of the problem. While CPU time is a machine-dependent measurement, WU is machine-independent. As shown in Figure 3*a*, in one V-cycle, ν_1 iterations are performed during pre-smoothing and ν_2 iterations are performed during post-smoothing in the finest grid. One WU is defined as the computation work needed to complete one MG cycle, which in this case is $\nu_1 + \nu_2$ iterations during one outer relaxation sweep over the finest grid. Table 2 compares the multigrid performance of a lid-driven cavity flow simulation with the top plate moving for different Reynolds numbers and grid

Table 1. Specifications of simulation parameters

Reynolds number	α_u	α_v	α_p	α_{pc}	α_{MG}	Coarse-grid iterations
100	0.5	0.5	0.8	1.0	0.8	45
400	0.5	0.5	0.8	1.0	0.8	50
1,000	0.5	0.5	0.8	1.0	0.8	50
3,200	0.4	0.4	0.7	1.0	0.8	120
5,000	0.4	0.4	0.6	1.0	0.7	150
7,500	0.4	0.4	0.6	1.0	0.7	180

Table 2. MG performance for different Reynolds numbers (number of processors = 4)

	Re = 100		Re = 400		Re = 1,000	
Grid	CPU time SG/MG	WU SG/MG	CPU time SG/MG	WU SG/MG	CPU time SG/MG	WU SG/MG
128 × 128	13.9	75.2	11.41	49.54	4.5	13.71
256 × 256	89.3	283.66	64.9	190.6	19.95	45.92
512 × 512	404.1	1,040.11	307.8	643.75	110.99	264.77
1,024 × 1,024	1,371.6	2,940.7	1,182.7	2,297.92	509.04	1,100.93

sizes. The simulations were conducted using 4 processors. It can be observed that significant speed-up over single-grid performance can be achieved both in CPU time and WU with the multigrid method. As the number of grid points increases, the speed-up increases. For 512 × 512 CVs, the CPU time in the single-grid is about 400 times higher than that in the multigrid. The MG performance declines with increase of Reynolds number, which is in accordance with the findings of other researchers [3, 5]. It is surprising that the SG/MG ratio of WU is much higher than the ratio of CPU time in all the cases. Since WU is a machine-independent measure of the work done in each MG cycle, its ratio should be close to the speed-up in CPU time, as reported in the literature [3]. This seemingly contradictory result can be explained by looking at the communication time of the parallel processors in Table 3. The communication time has been presented as a percentage of total execution time for all the SG and MG cases. As shown in Table 3, the percentage of communication time for MG is considerably higher than that for SG due to the very few number of nodes on each processor in the coarsest MG level. Thus, for the grids under consideration, the CPU time of multigrid simulations is dominated by the communication overhead, resulting in low parallel efficiency. Consequently, a higher number of computational work is done on a single grid than is predicted by the CPU time ratio.

5.3. Convergence Rate and Convergence Factor

Figure 8 demonstrates the convergence of residuals as a function of WU for single-grid and multigrid simulations with Re = 400. A steep decrease of multigrid

Table 3. Percentage of communication time (number of processors = 4)

	Re = 100		Re = 400		Re = 1,000	
Grid	%Comm time (SG)	%Comm time (MG)	%Comm time (SG)	%Comm time (MG)	%Comm time (SG)	%Comm time (MG)
128 × 128	15.83	67.80	10.68	79.13	10.44	58.31
256 × 256	3.82	51.54	3.58	41.11	3.88	35.28
512 × 512	1.93	21.23	2.08	22.62	2.11	20.25
1,024 × 1,024	1.39	7.74	1.25	6.18	1.51	9.50

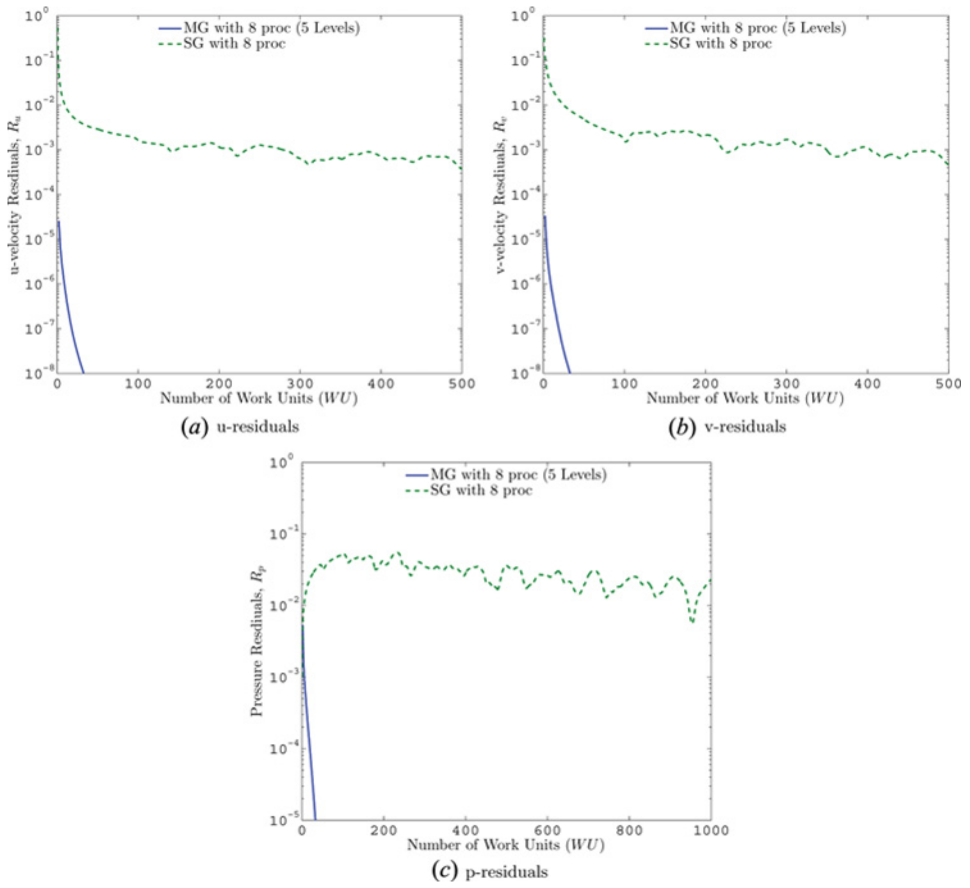


Figure 8. Residuals on a 512×512 CV grid as a function of work units (WU) for $\text{Re} = 400$.

residuals is observed from these graphs, whereas the single-grid exhibits very slow convergence. To quantify the convergence rate, we can introduce a convergence factor which is roughly the maximum factor by which the error is reduced by each work unit [33]. In other words, the lower the convergence factor, the greater the convergence rate. Since in practice the exact solution is not known, the error cannot be computed during the iterations. Thus, we define a convergence factor Ω based on the residuals which indicates the number of iterations required to reduce the residual by a factor of $10^{-\sigma}$. Let K be the smallest integer that satisfies

$$\frac{\| R^{(K)} \|}{\| R^{(0)} \|} \leq 10^{-\sigma} \tag{54}$$

The condition is approximately satisfied if

$$\Omega^{K} \leq 10^{-\sigma} \tag{55}$$

Thus, we can approximate Ω from the following equation:

$$-log_{10}(\Omega) \leq \frac{\sigma}{K} \tag{56}$$

Figure 9 shows the asymptotic value of the convergence factors as a function of the number of grid points. For relatively low Reynolds number cases (Re = 100–1,000), the convergence factor is lower, and it decreases with the increase of grid points, resulting in a fast convergence rate. However, for a higher Reynolds number case (Re = 3,200), the convergence factor remains very close to 0.98, indicating a degradation of FMG performance. The reason is that the prolongation of converged coarse-grid values to the next finer grid may not be very accurate due to the effect of large convection at a higher Reynolds number. Thus, although in theory we are expecting a "good" initial guess for the next finer level, in practice this may not be achievable at a higher Reynolds number. For the same reason, degradation of multigrid performance has been observed for turbulent flows. This observation is in complete accordance with the previous work of Lien and Leschziner [3] and Muzaferija [45].

5.4. Computational Complexity

In order to compare the computational work between single-grid (SG) and multigrid (MG) methods, we plotted the CPU time against the number of grid points in Figure 10. For single-grid simulations with 8 processors, the CPU time increases almost quadratically [$O(N^2)$], as expected. On the other hand, for multigrid simulations, the increase in CPU time follows an $O(N)$ trend for both 4 and 8 processors.

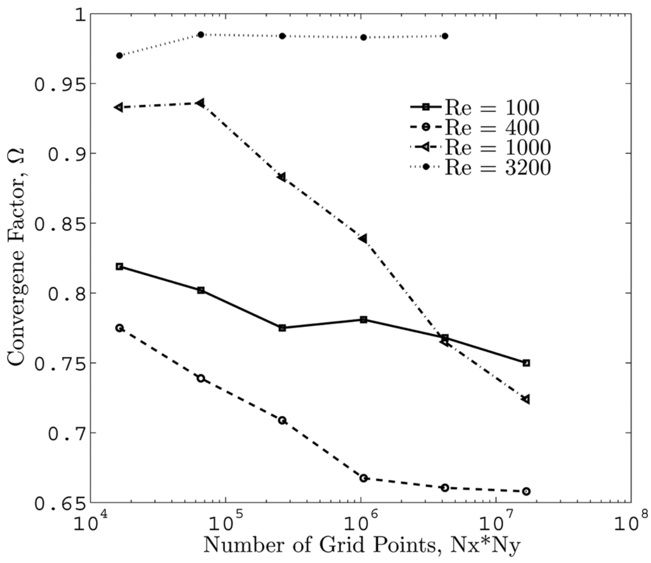


Figure 9. Asymptotic values of the convergence factors for different Reynolds numbers (number of processors = 4).

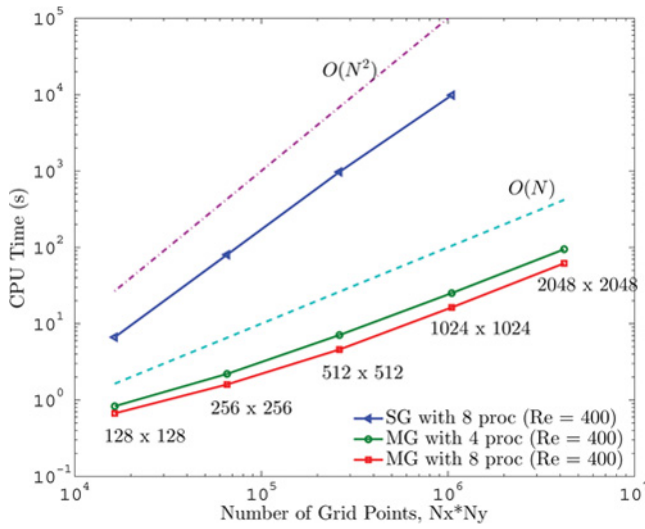


Figure 10. Comparison of parallel MG performance with SG for Re = 400. The dashed line has a slope 1, indicating MG effciency. The dashed-dotted line has a slope 2, indicating SG effciency.

The slight deviation of the curves from their ideal behavior may be attributed to the communication overhead due to the use of parallel machines.

Figure 11 demonstrates the effect of Reynolds number on the computational complexity. Although the CPU time increases with the increase of Reynolds number, the increase of computational work follows an $O(N)$ trend for a fixed Reynolds

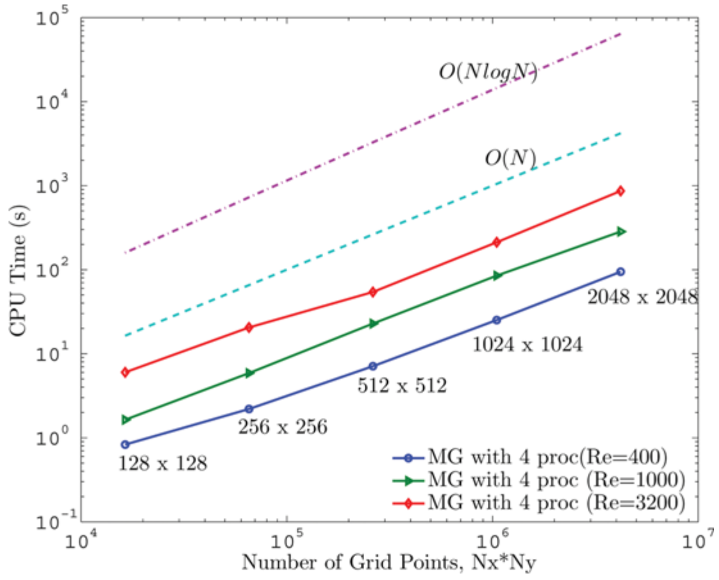


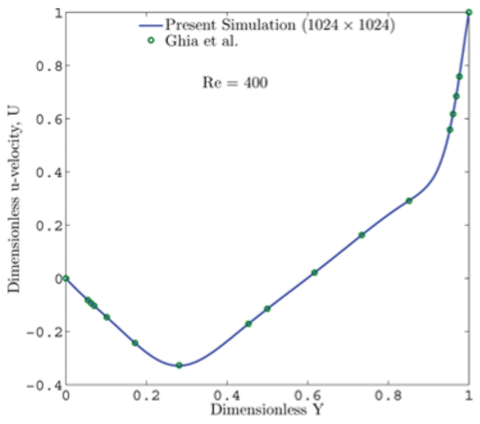
Figure 11. Effect of Reynolds number on parallel MG performance.

Table 4. Nondimensional u-velocity along the y-centerline for different Reynolds numbers. The top plate is moving with a uniform velocity. No-slip boundary conditions at the walls

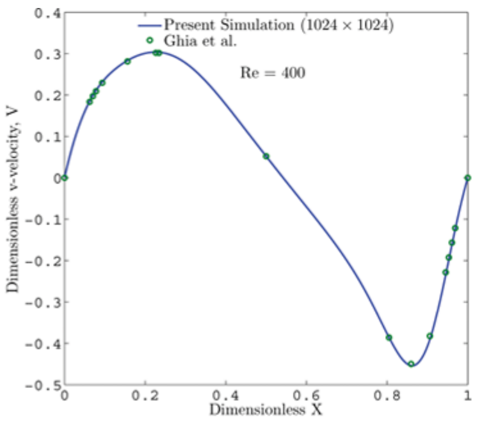
Y	Re = 400	Re = 1,000	Re = 3,200	Re = 5,000	Re = 7,500
0.0000	0.00000	0.00000	0.00000	0.00000	0.00000
0.0200	−0.03177	−0.07603	−0.17777	−0.22495	−0.27754
0.0405	−0.06189	−0.14120	−0.29262	−0.35426	−0.39631
0.0601	−0.08923	−0.19577	−0.37658	−0.43066	−0.43509
0.0806	−0.11732	−0.24927	−0.42746	−0.44392	−0.41603
0.1001	−0.14410	−0.29689	−0.43339	−0.41856	−0.38573
0.1206	−0.17257	−0.33989	−0.41101	−0.38853	−0.36219
0.1401	−0.19996	−0.36975	−0.38285	−0.36682	−0.34424
0.1606	−0.22849	−0.38617	−0.35711	−0.34766	−0.32580
0.1802	−0.25458	−0.38717	−0.33688	−0.32978	−0.30805
0.2007	−0.27956	−0.37527	−0.31763	−0.31067	−0.28948
0.5005	−0.11446	−0.06151	−0.03633	−0.03149	−0.02628
0.9009	0.35427	0.38479	0.41898	0.41901	0.39897
0.9106	0.37502	0.39215	0.43189	0.43391	0.41373
0.9204	0.40187	0.40008	0.44321	0.44822	0.42854
0.9302	0.43679	0.41073	0.45216	0.46100	0.44270
0.9409	0.48695	0.43021	0.45838	0.47176	0.45607
0.9507	0.54512	0.46149	0.46082	0.47691	0.46438
0.9604	0.61626	0.51371	0.46447	0.47686	0.46695
0.9702	0.70001	0.59432	0.48617	0.47842	0.46387
0.9800	0.79419	0.70664	0.56396	0.51934	0.47856
0.9907	0.90472	0.85977	0.76162	0.70875	0.64723
1.000	1.00000	1.00000	1.00000	1.00000	1.00000

Table 5. Non-dimensional v-velocity along the x-centerline for different Reynolds numbers. The top plate is moving with a uniform velocity. No-slip boundary conditions at the walls

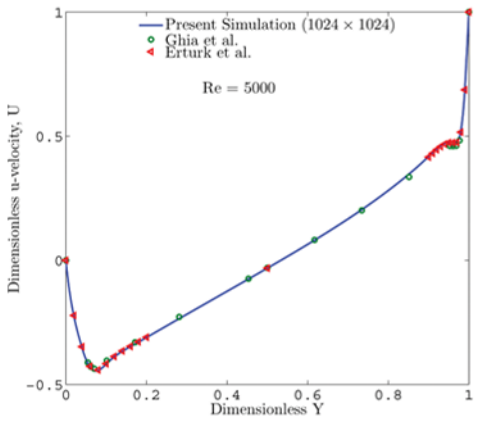
X	Re = 400	Re = 1,000	Re = 3,200	Re = 5,000	Re = 7,500
0.0000	0.00000	0.00000	0.00000	0.00000	0.00000
0.0151	0.05951	0.10292	0.18196	0.21956	0.24578
0.0308	0.11028	0.18283	0.29182	0.33292	0.35341
0.0454	0.14906	0.23661	0.34983	0.39131	0.40595
0.0600	0.18047	0.27521	0.39019	0.42905	0.431754
0.0747	0.20578	0.30403	0.41760	0.44543	0.43417
0.0903	0.22746	0.32819	0.43135	0.44258	0.42059
0.1049	0.24397	0.34645	0.43078	0.42821	0.40201
0.1206	0.25854	0.36149	0.41957	0.40821	0.38208
0.1352	0.27001	0.37111	0.40330	0.38937	0.36494
0.1450	0.27667	0.37494	0.39111	0.37749	0.35419
0.5005	0.05146	0.02526	0.01373	0.01119	0.00988
0.8501	−0.44994	−0.40326	−0.36762	−0.36416	−0.34357
0.8647	−0.45381	−0.44028	−0.38430	−0.38208	−0.36106
0.8804	−0.44362	−0.48158	−0.40198	−0.40059	−0.37991
0.8950	−0.41888	−0.51369	−0.42160	−0.41680	−0.39706
0.9106	−0.37613	−0.52674	−0.45415	−0.43483	−0.41371
0.9253	−0.32251	−0.50508	−0.50317	−0.46228	−0.42972
0.9399	−0.25931	−0.44280	−0.55640	−0.51565	−0.46188
0.9546	−0.19118	−0.34384	−0.55560	−0.57221	−0.532543
0.9702	−0.11873	−0.21597	−0.42028	−0.50375	−0.54975
0.9849	−0.05590	−0.09848	−0.19571	−0.24939	−0.30553
1.000	0.00000	0.00000	0.00000	0.00000	0.00000



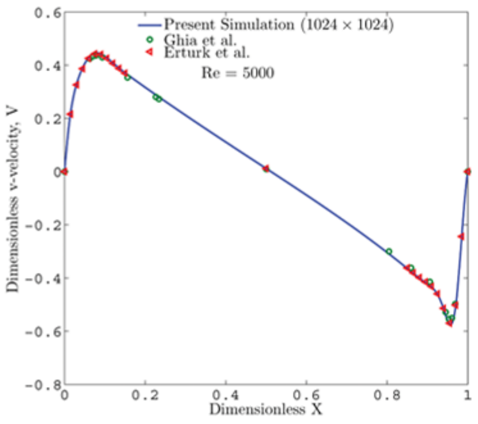
(*a*) u-velocity for Re = 400



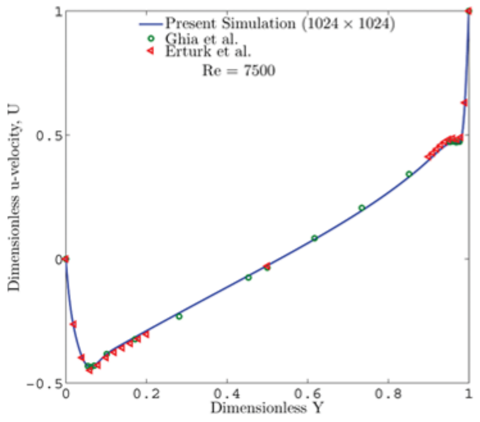
(*b*) v-velocity for Re = 400



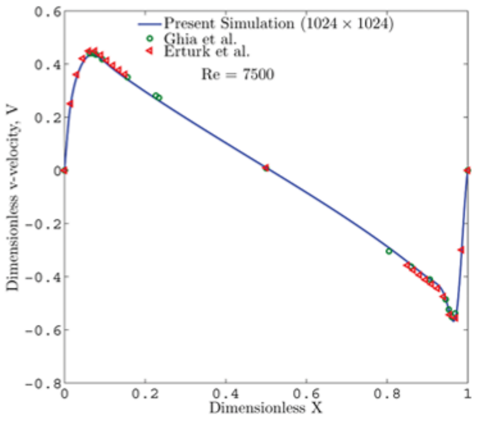
(*c*) u-velocity for Re = 5000



(*d*) v-velocity for Re = 5000



(*e*) u-velocity for Re = 7500



(*f*) v-velocity for Re = 7500

Figure 12. Centerline velocity profiles for Re = 400, 5,000, and 7,500.

number. The variation of trend for Re = 3,200 at lower grid sizes is again due to the very high communication overhead (74% and 45%, respectively) which occurs during a large number of coarse-grid iterations.

5.5. Benchmark Results for Lid-Driven Cavity Flow

One of the objectives of multigrid simulations is to conduct simulations in very fine grids and produce high-fidelity results. This is due to the fact that several orders of magnitude of speed-up in CPU time can be achieved with the multigrid method as compared to the single-grid method. In this study, lid-driven cavity flow simulations were performed using a grid of $1{,}024 \times 1{,}024$ CVs. Tables 4 and 5 tabulate the u- and v-velocity profiles along the vertical and horizontal centerlines of the cavity at different Reynolds numbers ranging from 400 to 7,500. Figure 12 presents the centerline u- and v-velocity profiles along the y and x directions, respectively, for selected Reynolds numbers. The results are compared with these of Ghia et al. [43] and Erturk et al. [44]. Very good agreement can be observed for the low-Reynolds-number case, Re = 400. As the Reynolds number increases (Re = 5,000 and 7,500), small deviations from the results of Ghia et al. [43] near the peak values of the velocity profiles were identified. It should be noted that the numerical scheme used by Ghia et al. is spatially second-order-accurate, and the benchmark results were presented for the 256×256 grid. Thus, it is quite possible that at high Reynolds number, the results may not be very accurate. Erturk et al. [44] also used a second-order-accurate spatial discretization scheme, but they presented the benchmark results for a relatively fine grid of 601×601. The velocity profiles from the present simulations match pretty well with the results of Erturk et al. [44] for Re = 5,000. For Re = 7,500, slight differences near the peak regions are seen. It has been tested in the present study (not shown, though) that $1{,}024 \times 1{,}024$ CVs form a suitable grid to sufficiently capture the large gradients at high Reynolds number. Since a second-order-accurate central difference scheme (CDS) has been applied to generate the simulation results, they can be used as benchmark values for future code validation.

6. CONCLUSIONS

A parallel multigrid finite-volume solver was developed to simulate two-dimensional incompressible Navier-Stokes equations. The code was tested and verified by conducting lid-driven cavity flow simulations. Good parallel scaling was observed up to 64 processors for a grid size of 512×512 CVs on single-grid simulations. The analysis showed that, as long as the communication time is small compared to the computation time, parallel speed-up can be achieved up to a large number of processors. However, in the multigrid framework, the maximum number of processors is limited by the coarse-grid size. The parallel multigrid code showed superior convergence over the single-grid counterpart. Multigrid speed-up as high as 1,370 was achieved for the finest grid size ($1{,}024 \times 1{,}024$). The ratio of work units between single grid and multigrid is not close to the ratio of CPU time. This is due to the large communication overhead occurring at the coarsest grid level during the multigrid iterations. It has been shown that the computational effort on multigrid increases proportionally to the number of grid points. Thus, the "golden rule" of the multigrid method, as proposed by Brandt [1], has been validated. The convergence factors for different Reynolds number were analyzed, and a good convergence rate has been shown for low-Reynolds-number flows. Using the parallel multigrid solver, very-high-fidelity solution for lid-driven cavity flow has been documented, which can be used for future benchmarking purposes.

REFERENCES

1. A. Brandt. Multi-level Adaptive Solutions to Boundary-Value Problems, *Math. Comput.*, vol. 31, pp. 333–390, 1977.
2. M. Hortmann, M. Perić, and G. Scheuerer, Finite Volume Multigrid Prediction of Laminar Natural Convection: Bench-Mark Solutions, *Int. J. Numer. Meth. Fluids*, vol. 11, pp. 189–207, 1990.
3. F. S. Lien, and M. A. Leschziner, Multigrid Acceleration for Recirculating Laminar, and Turbulent Flows Computed with a Non-orthogonal, Collocated Finite-Volume Scheme, *Comput. Meth. Appl. Mech. Eng.*, vol. 118, pp. 351–371, 1994.
4. D. S. Kumar, K. S. Kumar, and M. K. Das, A Fine Grid Solution for a Lid-Driven Cavity Flow Using Multigrid Method, *Eng. Appl. Comput. Fluid Mech.*, vol. 3, pp. 336–354, 2009.
5. W. Shyy, and C. S. Sun, Development of a Pressure-Correction/Staggered-Grid Based Multigrid Solver for Incompressible Recirculating Flows, *Comput. Fluids*, vol. 22, pp. 51–76, 1993.
6. S. Thakur, J. Wright, W. Shyy, J. Liu, H. Ouyang, and T. Vu, Development of Pressure-Based Composite Multigrid Methods for Complex Fluid Flows, *Prog. Aeros. Sci.*, vol. 32, pp. 313–375, 1996.
7. K. M. Smith, W. K. Cope, and S. P. Vanka, A Multigrid Procedure for Three-Dimensional Flows on Non-orthogonal Collocated Grids, *Int. J. Numer. Meth. Fluids*, vol. 17, pp. 887–904, 1993.
8. J. Yan, and F. Thiele, Performance, and Accuracy of a Modified Full Multigrid Algorithm for Fluid Flow, and Heat Transfer, *Numer. Heat Transfer, B*, vol. 34, pp. 323–338, 1998.
9. S. P. Vanka, Block-Implicit Multigrid Solution of Navier-Stokes Equations in Primitive Variables, *J. Comput. Phys.*, vol. 65, pp. 138–158, 1986.
10. S. P. Vanka, A Calculation Procedure for Three-Dimensional Steady Recirculating Flows Using Multigrid Methods, *Comput. Meth. Appl. Mech. Eng.*, vol. 55, pp. 321–338, 1986.
11. D. S. Joshi, and S. P. Vanka, Multigrid Calculation Procedure for Internal Flows in Complex Geometries, *Numer. Heat Transfer B*, vol. 20, pp. 61–80, 1991.
12. P. S. Sathyamurthy, and S. V. Patankar, Block-Correction-Based Multigrid Method for Fluid Flow Problems, *Numer. Heat Transfer B*, vol. 25, pp. 375–394, 1994.
13. J. E. Jones, and S. F. McCormick, Parallel Multigrid Methods, In *Parallel Numerical Algorithms*, pp. 203–224, Springer-Verlag, 1997.
14. F. Durst, and M. Schäfer, A Parallel Block-Structured Multigrid Method for the Prediction of Incompressible Flows, *Int. J. Numer. Meth. Fluids*, vol. 22, pp. 549–565, 1996.
15. P. Wang, and R. D. Ferraro, Parallel Multigrid Finite Volume Computation of Three-Dimensional Thermal Convection, *Comput. Math. Appl.*, vol. 37, no. 9, pp. 49–60, 1999.
16. E. Chow, R. D. Falgout, J. J. Hu, R. S. Tuminaro, and U. M. Yang, A Survey of Parallelization Techniques for Multigrid Solvers, *Parallel Process. Sci. Comput.*, vol. 20, pp. 179–201, 2006.

17. S. V. Patankar, *Numerical Heat Transfer, and Fluid Flow*, Hemisphere, Washington, DC, 1980.
18. F. H. Harlow, and J. E. Welch, Numerical Calculation of Time-Dependent Viscous Incompressible Flow of Fluid with Free Surface, *Phys. Fluids*, vol. 8, pp. 2182, 1965.
19. S. V. Patankar, and D. B. Spalding, A Calculation Procedure for Heat, Mass and Momentum Transfer in Three-Dimensional Parabolic Flows, *Int. J. Heat Mass Transfer*, vol. 15, pp. 1787–1806, 1972.
20. P. Ding, and D. L. Sun, A Pressure-Based Segregated Solver for Incompressible Flow on Unstructured Grids, *Numer. Heat Transfer B*, vol. 64, pp. 460–479, 2013.
21. D. Chatterjee, and B. Mondal, Forced Convection Heat Transfer from Tandem Square Cylinders for Various Spacing Ratios, *Numer. Heat Transfer A*, vol. 61, pp. 381–400, 2012.
22. D. Chatterjee, G. Biswas, and S. Amiroudine, Mixed Convection Heat Transfer from an In-line Row of Square Cylinders in Cross-flow at Low Reynolds Number, *Numer. Heat Transfer A*, vol. 61, pp. 891–911, 2012.
23. D. Chatterjee, and B. Mondal, Mixed Convection Heat Transfer from Tandem Square Cylinders for Various Gap to Size Ratios, *Numer. Heat Transfer A*, vol. 63, pp. 101–119, 2013.
24. H. R. Ebrahimi-Kebria, M. Darbandi, and S. F. Hosseinizadeh, Numerical Simulation of Low-Mach-Number Laminar Mixing and Reacting Flows Using a Dual-Purpose Pressure-Based Algorithm, *Numer. Heat Transfer B*, vol. 59, pp. 495–514, 2011.
25. G. Marck, M. Nemer, and J. L. Harion, Topology Optimization of Heat and Mass Transfer Problems: Laminar Flow, *Numer. Heat Transfer B*, vol. 63, pp. 508–539, 2013.
26. P. Ming, and W. Zhang, Numerical Simulation of Natural Convection, and Radiation Heat Transfer in Two-Dimensional Enclosure on Hybrid Grids, *Numer. Heat Transfer B*, vol. 61, pp. 505–520, 2012.
27. F. Zhou, and I. Catton, Numerical Evaluation of Flow, and Heat Transfer in Plate-Pin Fin Heat Sinks with Various Pin Cross-sections, *Numer. Heat Transfer A*, vol. 60, pp. 107–128, 2011.
28. C. M. Rhie, and W. L. Chow, Numerical Study of the Turbulent Flow Past an Airfoil with Trailing Edge Separation, *AIAA J.*, vol. 21, pp. 1525–1532, 1983.
29. J. H. Ferziger and M. Perić, *Computational Methods for Fluid Dynamics*, Vol. 3. Springer-Verlag, Berlin, 2002.
30. M. Ijaz, and N. K. Anand, Co-located Variables Approach Using Implicit Runge-Kutta Methods for Unsteady Incompressible Flow Simulation, *Numer. Heat Transfer B*, vol. 54, pp. 291–313, 2008.
31. N. Kim, N. K. Anand, and D. L. Rhode, A Study on Convergence Criteria for a SIMPLE-Based Finite-Volume Algorithm, *Numer. Heat Transfer B*, vol. 34, pp. 401–417, 1998.
32. W. Hackbusch, *Multi-grid Methods and Applications*, vol. 4, Springer-Verlag, Berlin, 1985.
33. W. L. Briggs and S. F. McCormick, *A Multigrid Tutorial*, vol. 72. Siam, 2000.
34. U. Trottenberg, C. W. Oosterlee, and A. Schuller, *Multigrid*, Academic Press, 2000.
35. A. Brandt, and O. E. Livne, *Multigrid Techniques: 1984 Guide with Applications to Fluid Dynamics*, vol. 67. SIAM, 2011.
36. J. G. B. Saldaña, V. Sarin, and N. K. Anand, Parallelization of a SIMPLE-Based Algorithm to Simulate Mixed Convective Flow over a Backward-Facing Step, *Numer. Heat Transfer B*, vol. 56, pp. 105–118, 2009.
37. D. A. Donzis, P. K. Yeung, and D. Pekurovsky, Turbulence Simulations on O (10^4) Processors, *Proc. TeraGrid*, 2008.
38. S. Jagannathan, and D. A. Donzis, Massively Parallel Direct Numerical Simulations of Forced Compressible Turbulence: A Hybrid MPI/OpenMP Approach, in *Proceedings*

of the 1st Conference of the Extreme Science, and Engineering Discovery Environment: Bridging from the eXtreme to the Campus, and Beyond, p. 23, ACM, 2012.

39. L. Zhao, and C. Zhang, A Parallel Unstructured Finite-Volume Method for All-Speed Flows, *Numer. Heat Transfer B*, vol. 65, pp. 336–358, 2014.
40. P. Balaji, W. Bland, W. Gropp, R. Latham, H. Lu, A. J. Pena, K. Raffenetti, R. Thakur, and J. Zhang, *MPICH User's Guide*. 2014.
41. M. Ijaz. Implicit Runge-Kutta Methods to Simulate Unsteady Incompressible Flows, Ph.D. thesis, Texas A&M University, 2007.
42. P. N. Shankar, and M. D. Deshpande, Fluid Mechanics in the Driven Cavity, *Ann. Rev. Fluid Mech.*, vol. 32, pp. 93–136, 2000.
43. U. K. N. G. Ghia, K. N. Ghia, and C. T. Shin, High-Re Solutions for Incompressible Flow Using the Navier-Stokes equations and a Multigrid Method, *J. Comput. Phys.*, vol. 48, pp. 387–411, 1982.
44. E. Erturk, T. C. Corke, and C. Gökçöl, Numerical Solutions of 2-D Steady Incompressible Driven Cavity Flow at High Reynolds Numbers, *Int. J. Numer. Meth. Fluids*, vol. 48, pp. 747–774, 2005.
45. S. Muzaferija, Adaptive Finite Volume Method for Flow Prediction Using Unstructured Meshes and Multigrid Approach, Ph.D. thesis, University of London, UK, 1994.