A Parallel Semi-Coarsening Multigrid Algorithm for Solving the Reynolds-Averaged Navier-Stokes Equations

Kiattisak Ngiamsoongnirn
School of Mechanical Engineering, Institute of Engineering,
Suranaree University of Technology, Nakhon Ratchasima 30000, Thailand,
Phone: (+6644)224410-2, Email: Kiatt2000@hotmail.com

Ekachai Juntasaro
School of Mechanical Engineering, Institute of Engineering,
Suranaree University of Technology, Nakhon Ratchasima 30000, Thailand,
Phone: (+6644)224410-2, Email: junta@sut.ac.th

Varangrat Juntasaro
Department of Mechanical Engineering, Faculty of Engineering,
Kasetsart University, Bangkok 10900, Thailand,
Phone: (+662)9428555 ext 1829, Email: fengvrj@ku.ac.th

Putchong Uthayopas
Department of Computer Engineering, Faculty of Engineering,
Kasetsart University, Bangkok 10900, Thailand,
Phone: (+662)9428555 ext 1416, Email: pu@ku.ac.th

Abstract

The multigrid method adopted in conjunction with the parallel computing is presented. The standard multigrid doubling the mesh size in all directions, called full-coarsening technique, suffers from the partitioning of data for parallel computing. To remedy this problem, the semi-coarsening technique should be used instead. This paper is aimed to present an algorithm of the semi-coarsening multigrid technique combined with the parallel computing technique. The parallel computing technique used is the one based on the distributed memory machine. The MPI library is adopted in order to exchange the data among processors. The solver code is developed for three-dimensional turbulent flows and validated with the available experimental data.

1. Introduction

In recent years, Computational Fluid Dynamics (CFD) has been widely used as a design tool in industries due to, for example, the lack of instruments to measure some quantities in some dangerous zones. In addition, the advantages of CFD are the low cost to construct and the ability to immediately observe some phenomena through a monitor of personal computer (PC). However, when the flow becomes more complicated, such as turbulent flow, the number of data points required to capture the physics of flow has to be large enough. A single computer is limited to its memory and speed. In other words, it may compute one problem using a large number of iterations and hence longer computational time or it cannot sustain to a large number of data at all. The convergence rate of iterative methods can be greatly improved by using multigrid acceleration techniques. The multigrid methods eliminate

Proceedings of the Seventh International Conference on High Performance Computing and Grid in Asia Pacific Region (HPCAsia'04)
0-7695-2138-X/04 $ 20.00 IEEE

some errors on the coarser grid that cannot be eliminated by the fine grid and then correct the solutions up to the finer grid. The time consumption, moreover, can be decreased by using a large number of computers to simultaneously solve the same problem, which is the so-called parallel computing.

Over the past decade, the standard multigrid method has been successful for solving several problems, but the efficiency of the multigrid methods is reduced dramatically by the presence of anisotropies [1]. The anisotropies occur when the coefficients of the discretized equation vary throughout the domain or when the non-uniform grids are used. As a result, several literatures attempted to develop a semi-coarsening multigrid technique to alleviate this problem [1,6,7,8,9]. Such efforts focused on the development of a smoother that can capture the anisotropic characteristics. For example, Montero et al [1] developed an alternating-plane smoother combined with the full-coarsening multigrid technique and a plane-implicit smoother in conjunction with a semi-coarsening multigrid technique for smoothing out the errors.

In this paper the combination of the parallel computing and the semi-coarsening multigrid method for the 3D Reynolds-Averaged Navier-Stokes solver is presented. The solver code is performed for the steady turbulent flow in a three-dimensional cavity. The numerical solutions are then compared with the experimental data of Prasad and Koseff [13]. To take into account the data synchronization, interchanging and updating the boundary data at the interface between a pair of adjacent processors are essential. Therefore, the Message-Passing Interface (MPI) library is used in the present work to perform such a task.

2. Numerical Method

2.1 Governing Equations

The flow considered in this work is 3D steady turbulent incompressible flow. The governing equations are composed of the conservation laws of mass, momentum, turbulence kinetic energy and dissipation rate of turbulence kinetic energy described as follows:

The conservation of mass can be expressed as

$$\frac{\partial(\rho u_j)}{\partial x_j} = 0$$

where ρ is the fluid density and u_j are the mean-flow velocity components in x, y and z directions for j=1, 2 and 3 respectively. The Reynolds-averaged momentum equations can be formulated as

$$\frac{\partial}{\partial x_j}(\rho u_i u_j) = \frac{\partial}{\partial x_j}(\tau_{ij} - \rho\overline{u_i' u'}_j) - \frac{\partial p}{\partial x}$$

where p and τ_{ij} are respectively the pressure and the laminar stress tensor and

$$\tau_{ij} = \mu\left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i}\right)$$

and the Reynolds-stress tensor $-\rho\overline{u_i' u_j'}$ is

$$-\rho\overline{u_i' u_j'} = \mu_t\left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i}\right) - \frac{2}{3}\delta_{ij}(\rho k) \ .$$

The turbulence kinetic energy equation is expressed as

$$\frac{\partial}{\partial x_j}(\rho u_j k) = \frac{\partial}{\partial x_j}\left(\mu_{eff}\frac{\partial k}{\partial x_j}\right) - \rho\overline{u_i' u_j'}\frac{\partial u_i}{\partial x_j} - \rho\varepsilon - \rho D$$

The dissipation rate of turbulence kinetic energy is given as

$$\frac{\partial}{\partial x_j}(\rho u_j \varepsilon) = \frac{\partial}{\partial x_j}\left(\mu_{eff}\frac{\partial \varepsilon}{\partial x_j}\right) + C_{\varepsilon 1} f_{\varepsilon 1}\frac{\varepsilon}{k}\left(-\rho\overline{u_i' u_j'}\frac{\partial u_i}{\partial x_j}\right) - \rho C_{\varepsilon 2} f_{\varepsilon 2}\frac{\varepsilon^2}{k} + \rho E$$

where $\mu_{eff} = \mu + \dfrac{\mu_t}{\sigma_{k,\varepsilon}}$ and $\mu_t = \rho C_\mu f_\mu \dfrac{k^2}{\varepsilon}$.

The extra terms and damping function arising to account for the low Reynolds-number region are

$$D = \frac{2\mu}{\rho}\left(\frac{\partial\sqrt{k}}{\partial x_i}\right)^2$$

$$E = \frac{2\mu}{\rho}\frac{\mu_t}{\rho}\left(\frac{\partial^2 u_i}{\partial x_k \partial x_m}\right)^2$$

$$f_{\varepsilon 2} = 1 - 0.3\exp(-\mathrm{Re}_t^2)$$

$$\mathrm{Re}_t = \frac{\rho k^2}{\mu\varepsilon}$$

and the turbulence model constants are as follows:

$\sigma_k = 1.0$, $\sigma_\varepsilon = 1.3$, $C_\mu = 0.09$, $C_{\varepsilon 1} = 1.44$, $C_{\varepsilon 2} = 1.92$ and $f_{\varepsilon 1} = 1.0$

2.2 Solution Procedure

To numerically solve the system of partial differential equations (PDEs), the equations have to be converted into the system of

algebraic equations and the physical domain has to be transformed into the computational domain in which the domain is subdivided into a finite number of cells or control volumes (CVs). The first process is well known in terms of the discretization process and the second one is the so-called grid generation procedure. To discretize the governing equations, a finite volume method (FVM) is adopted here in conjunction with a collocated grid arrangement in which all variables are stored at the cell center of CV. In contrast to a collocated grid system, a staggered grid system is the system that the scalar variables are stored at the cell center but the velocity components, u, v and w, are stored at the cell face of CV. There are some literatures that compared these two strategies, for example, Meier et al [11] concluded that the staggered grid arrangement had an advantage over the collocated grid when dealing with high pressure gradient and multi-phase flow; however, Peric et al [12] stated that the collocated grid converged faster in some cases, 3 cases in their work, and had advantages when multigrid techniques, non-orthogonal grids and 3D problem were considered.

The partial differential equations that govern the flow equation described in section 2.1 can be written in the general form as

$$\frac{\partial}{\partial x_j}(\rho u_j \phi) = \frac{\partial}{\partial x_j}\left(\Gamma_\phi \frac{\partial \phi}{\partial x_j}\right) + S_\phi$$

where Table 1 summarizes all the relevant terms.

Governing equations	ϕ	Γ_ϕ	S_ϕ
Continuity	1	0	0
Momentum	u_i	$\mu + \mu_t$	$-\frac{\partial p}{\partial x_j} - \frac{2}{3}\frac{\partial}{\partial x_j}(\rho k)$
Turbulence kinetic energy	k	$\mu + \frac{\mu_t}{\sigma_k}$	$-\rho\overline{u_i' u_j'}\frac{\partial u_i}{\partial x_j} - \rho\varepsilon - \rho D$
Turbulence dissipation rate	ε	$\mu + \frac{\mu_t}{\sigma_\varepsilon}$	$C_{\varepsilon 1} f_{\varepsilon 1}\frac{\varepsilon}{k}\left(-\rho\overline{u_i' u_j'}\frac{\partial u_i}{\partial x_j}\right) - \rho C_{\varepsilon 2} f_{\varepsilon 2}\frac{\varepsilon^2}{k} + \rho E$

Table 1. Relevant terms for the governing equations

By applying the finite volume formulation to the governing equations, the convection terms are discretized by the first-order upwind differencing scheme and the second-order central differencing scheme is employed for the diffusion and source terms. Discretizing all equations gives rise to a system of algebraic equation which one can be formulated in the standard finite volume form as

$$a_P \phi_P = \sum_{E,W,N,S,T,B} a_{nb}\phi_{nb} + S_\phi \forall$$

where $\forall$ is the volume of CV and E, W, N, S, T and B stand respectively for the East, West, North, South, Top and Bottom nodes with respect to the cell-centered node at P while ϕ and S_ϕ are respectively the dependent variable and the source terms which are given in Table 1. The coefficients a_P and a_{nb} contain the convection and diffusion fluxes and the characteristics of each CV.

In general there are two approaches for iteratively solving a system of discretised equations [1]: First, the couple method where the momentum and continuity equations are satisfied simultaneously and second, the distributive (segregated) or uncouple method in which the momentum equations are solved in the first step, and then the velocity and pressure are corrected in order to satisfy the continuity equation. The second approach is adopted in this work and the SIMPLE algorithm is used as the process of correcting the velocity and pressure to satisfy the continuity equation by solving the Poisson type of pressure correction equation arising from an imbalance mass of continuity equation, and since the collocated grid is used; therefore, adopting the Rhie and Chow interpolation is necessary to avoid the checkerboard of pressure. Once the velocity and pressure have been treated in such a way that the continuity equation is satisfied, then the k and ε equations are solved subsequently. This work employs an explicit point-wise Gauss-Seidel iterative method for solving the algebraic system of discretized equations. The direction of sweeping is initiated at the lower-left corner, as shown in Fig. 1, and goes to the upper-right corner for the even number of iterations. For the

odd number of iterations the starting point is at the upper-right corner and then sweep down to the lower-left corner. Using this strategy, the boundary conditions are spread throughout the domain faster. Moreover, the alternating direction of sweeping can alleviate the problem of anisotropies in the case of a sharp gradient that does not align with the direction coordinate and in the case of non-uniform grid that causes the solution coefficients vary throughout the domain. The algorithm is depicted in Fig. 1.

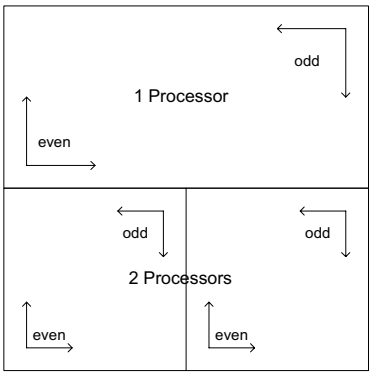

Figure 1. Sweeping direction of an iterative smoother

Considering the source terms of the k and ε equations in Table 1, most of the terms have negative quantities. This may generate the negative values of k and ε, physically being positive, that cause the solutions diverge eventually. This problem can be alleviated by moving the negative source terms to combine with the main coefficient a_P on the left-hand side, and hence the main coefficients become dominant. This treatment also has an advantage in the viewpoint of numerical stability.

3. Multigrid Technique

As well known that the conventional iterative methods are deteriorated by the error not comparable to the mesh size, the errors are eliminated rapidly for the first few iterations and then they decrease very slowly. From the Fourier analysis, the line of error tracing can be split into the combination of several sine and cosine waves and these waves can be classified into two groups: long and short wave lengths. The short wave length error components are well eliminated on fine mesh of which the mesh size can be comparable to or has the same dimension as the wave length of error components. This is the reason why the convergence rate is quick during the first few steps because the short wave length error components can be eliminated and the long ones that cannot be eliminated on the fine mesh still remain and cause the rate of convergence slow down. The main idea of the multigrid method is to use the coarser grid to smooth out the longer wave length error components and determine the correction to correct the solutions on the finer mesh. The full approximation storage (FAS) version of multigrid technique is adopted in this work due to the nonlinearity of the governing equations and the multigrid V cycle type is employed.

To incorporate the FAS technique with the SIMPLE algorithm, some special treatment must be taken carefully. The velocity components are nonlinear but the pressure is linear and both the velocities and pressure appear in the system of equations. Therefore, the pressure is solved through the pressure correction equation by the multigrid correction scheme and the FAS is used for the solution of the velocity components. The current approximation of the velocity components, u, v and w, as well as the residual of the momentum and pressure correction equations are restricted to the next coarser grid. Once a coarse grid is visited, the coarse-grid pressure is initialized with the guessed value of zero every time, and all the coefficients together with the mass flux have to be recalculated on this grid with the restricted solution, and then the process of the SIMPLE algorithm is proceeded for a few iterations in a similar manner to a single-grid problem with additional source terms. As the restriction process has gone down to the coarsest grid and the coarsest-grid problem is solved, the change in the velocity components and the current solution of the pressure are then prolonged up to the next finer grid for correcting the fine-grid current approximation of pressure and velocity components. The residuals are transferred to the coarser grid by summing up the residual of fine grid that contains in the coarse grid CV. The restriction and prolongation of variables are bilinear interpolation (not trilinear). Even though this work is concerned with a 3D case, but in order to exploit a full ability of multigrid, the semi-coarsened grid is employed in such a way that grid is coarsened as coarse as possible only on each yz plane in which the number of grid points in the x-direction is fixed. Therefore this is a similar manner to a 2D case. In addition, this is done for the sake of simplicity and appropriately when incorporated with parallel computing. The

summary of 2 grid levels of multigrid algorithm in conjunction with the SIMPLE algorithm is shown step by step below:

1. $a_P^h u_P^h = \sum a_{nb}^h u_{nb}^h + S^h$,
   $A_P^{2h} p'_P = \sum A_{nb}^{2h} p'_{nb} - m^k$
   $u^h \leftarrow p'$, $p^h \leftarrow p'$
2. $R^h = \sum a_{nb}^h u_{nb}^h + S^h - a_P^h u_P^h$
3. $u^{2h} = I_h^{2h} u^h$, $R^{2h} = I_h^{2h} R^h$
4. $f^{2h} = a_P^{2h} u_P^{2h} - \sum a_{nb}^{2h} u_{nb}^{2h} + R^{2h}$, $p^{2h} = 0$
5. $a_P^{2h} u_P^{2h} = \sum a_{nb}^{2h} u_{nb}^{2h} + S^{2h} + f^{2h}$,
   $A_P^{2h} p'_P = \sum A_{nb}^{2h} p'_{nb} + m^0 - m^k$,
   $u^{2h} \leftarrow p'$, $p^{2h} \leftarrow p'$
6. $u^h = u^h + I_{2h}^h (u^{2h} - I_h^{2h} u_h)$,
   $p^h = p^h + I_{2h}^h p^{2h}$
7. $a_P^h u_P^h = \sum a_{nb}^h u_{nb}^h + S^h$,
   $A_P^{2h} p'_P = \sum A_{nb}^{2h} p'_{nb} - m^k$,
   $u^h \leftarrow p'$, $p^h \leftarrow p'$

where $m^k = (F_t^k - F_b^k + F_n^k - F_s^k + F_e^k - F_w^k)$ and k is the iteration number and m^0 is evaluated in step 4 , F is the mass flux. It should be noted that this algorithm is different from several literatures and even among them they are also different from each other.

In multigrid method, when the turbulence model equations are employed on the coarse grid, the coarse grid correction often causes the turbulence quantities become negative and the solutions diverge eventually. Therefore, in this work, the turbulence quantities are solved only on the finest grid.

4. Parallelization Technique

The basic idea of the parallel computing is that a number of processors work in cooperation on a single task. There are generally two types of memory model architecture of parallel processors: distributed and shared memory models. The first one, each processor possesses its own local memory and connects to each other through the high-speed interconnection network. For the second type, all processors do not have its own local memory but use the same global memory. However, in the viewpoint of programming for controlling the instruction, it can be classified into two categories: Single Instruction Multiple Data (SIMD) and Multiple Instruction Multiple Data (MIMD) parallel computing. The first type is also called the data parallelization and the second one is called the functional parallelization. In this work the SIMD is used in conjunction with the distributed memory architecture parallel machine. The message passing is done by using the MPI library.

In the present work, the data parallel computing is chosen because it is more appropriate for the present CFD algorithm than the functional parallel computing. This is because the functions used in the present algorithm are the same throughout the calculation procedure. Therefore it is easier to partition a whole data into several sub-data and use the same operation solving simultaneously all the sub-data. The details about the exchanging of data among processors is documented in the two previous works by the authors [14,15].

The parallel computing model used in this work is the master-slave model. The master is assigned to initialize the necessary data such as boundary and initial conditions and send this data to the slaves. The slaves receive the corresponding data from the master and each slave performs the calculation on its own sub-data with the same code as the sequential algorithm does. In the present algorithm, there is one processor, typically root processor, which is treated as both master and slave. Therefore, it has to allocate both overall data and sub-data. This leads to the limitation of memory usage. In the future work, all the initialized data will be stored in the data file and used as the input data. Then let all slaves read their own sub-data and write into the output file by themselves. Thus there is no longer the master.

The parallel algorithm for solving the steady incompressible turbulent flow is summarized as follows:

1. Solve the momentum equations with an initial guess of the pressure field
2. Calculate the pressure gradient
3. Exchange the momentum central coefficient, a_P , the updated velocity components, u_i , the updated pressure p and the pressure gradient, $\frac{\partial p}{\partial x_j}$, between the adjacent nodes which are used in the pressure correction equation and in the Rhie and Chow interpolation respectively
4. Solve the pressure correction, p' , for three iterations and in each iteration

exchange p' between the adjacent processors

5. Correct the velocity components, u_i, and the pressure field, p, with pressure correction, p'
6. Exchange the velocity components, u_i, and the pressure p
7. Solve the turbulence kinetic energy equation, k
8. Exchange k
9. Solve the dissipation rate of turbulence kinetic energy equation, ε
10. Exchange ε
11. Update the boundary condition
12. Check convergence, if not then go to step 1

5. Parallel Multigrid Algorithm

There are two approaches to incorporate the parallel computing with the multigrid method [1]: the domain decomposition with multigrid and the multigrid with grid partitioning. The domain decomposition is first applied to the finest grid. Then, the multigrid method is used to solve the problem inside each block, that is, the multigrid V-cycle is applied in each local finest grid. Since inter-domain connection is limited to the finest level, thus communications are only required at this level. For the multigrid with grid partitioning, however, the multigrid method is used to solve the problem over the whole grid in which the grid is partitioned among processors at each level. Therefore, communications are required at each level.

Domain decomposition methods are often used with the finite element method on parallel computers [12]. They are easier to implement and require fewer communications only on the finest grid. Additionally, they can be applied to general multiblock grids. On the other hand, the domain decomposition methods lead to algorithms which are numerically different from the sequential version and have a negative impact on the convergence rate. Grid partitioning retains the convergence rate of the sequential algorithm. Moreover, it requires more communication overhead because the data exchange at each grid level. Several literatures adopted the grid partitioning technique, for examples, [10, 12]. In the present work, the domain decomposition is employed in combination with the semi-coarsening multigrid technique. Some literatures adopted the semi-coarsened grid, for examples [6,7,8,9], for the reason of an anisotropic characteristic of a problem such as grid clustering. In this work, a coarsened grid is used on the yz plane which is the x-semicoarsening and the domain decomposition is applied in the x-direction. Llorente et al [6] reported that for the problem size of 64^3, the computing time of z-semi-coarsening, say xy plane coarsened grid, is about 17% larger than x-semicoarsening (yz-plane coarsened grid). This paper will pursue such a work.

6. Results and Discussion

This work is based on the development of a CFD code. Therefore, the code is firstly validated by solving the standard benchmark problem and the results are compared with the reference data. Since the solver code is focused on the 3D turbulent flow. The popular benchmark problem is a lid-driven cavity flow and the reference data are the experimental results of Prasad and Koseff [13]. The u- and v-velocity components along the centerline of a cavity are plotted against the reference data at Re= 3,200 and 10,000 as shown in Fig. 2 and Fig. 3 respectively. The present results agree very well with the experimental data.

The next step is to find an optimal number of multigrid levels. The solver code performs the computation with the 64^3 uniform grid points at Re=3,200 only because a highly nonlinear behavior cause the multigrid method fail to perform a computation at Re=10,000 and also because of the limitation of memory usage. The results are depicted in Fig. 4. It is found that, for 6 multigrid levels, the line of the residual reduction is nearly coincided with 5 multigrid levels. Therefore, this work, in the case of 64^3 grid points, uses 5 grid levels for further testing the parallel multigrid performance. Although some literatures stated that 4 grid levels are sufficient for solving the incompressible flow problems especially in [3] that uses the full-coarsening multigrid for solving both recirculating laminar and turbulent flows with a non-orthogonal collocated grid. For the result of this work, in the series of 2^k grid points, where k= 1, 2, 3, … are the k^{th} grid level and k=1 is the coarsest grid, the optimal number of grid levels is the number that the finest grid is coarsened in such a way that the coarsest grid is the 2^2 grid points.

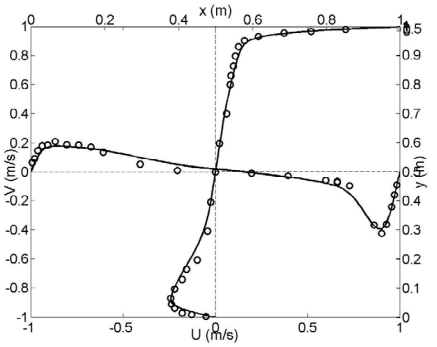

Figure 2. Re=3,200

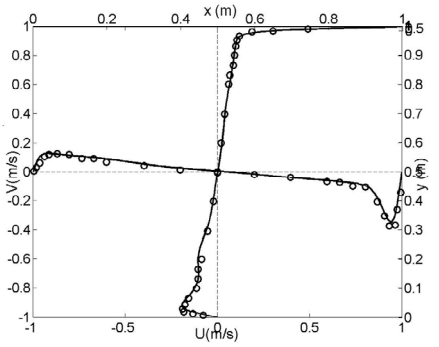

Figure 3. Re=10,000

The optimal number of multigrid levels in this work is 5 levels. The parallel computing is performed by partitioning the data only in the x-direction among all processors. Once the data is partitioned, each processor contains various yz plane of data, and then the multigrid V-cycle is applied on the chunk of data of each processor by doubling the mesh size only on the yz plane and not changing in the x coordinate. The residuals are plotted against a computing time by varying the number of processors including a non-parallel version of a single grid are shown in Fig. 5. The overshoot of all lines is due to initiating the calculation with the laminar flow for the first 200 iterations and then use the latest solution to initialize the turbulent flow. It seems that for the problem size of 64^3 grid points using 8 processors is an optimum because the computing time of 16 processors is slightly smaller than that of 8 processors. Fig. 6 shows the speed up, S, based on the sequential time, T_S, of a single grid.

The final discussion is the effect of using a non-uniform grid. Grid points are generated with the cubic polynomial function in which the two points next to the left and right

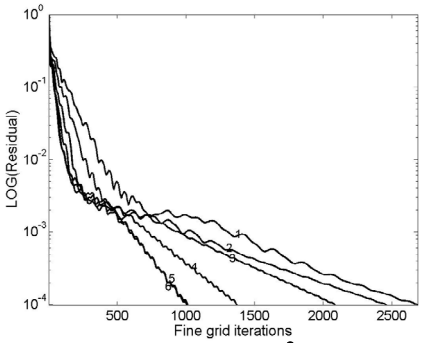

Figure 4. The residual of 64^3 grid points with the different number of grid levels

boundaries are first specified to be the constraints of a cubic polynomial. The rest points are distributed according to the cubic polynomial. A spacing of the two points next to the boundary is calculated by multiplying a clustering factor, more than zero, to the spacing of a uniform grid. If the factor is 1, grid points are uniform, and if it is less than 1, grid points are clustered at boundary. In Fig. 8, the clustering factor used is 0.9. It is found that the rate of convergence of the non-uniform grids is slower than the uniform ones, especially when employing the smaller number of processors. For other clustering factors, less than 0.9, the solution converges very slowly as shown in Fig. 7 in which 4 processors are used. Fig. 9 is an evaluation of a smoother sweeping direction. The numbers at the lines are the clustering factor, 1.0 for uniform grid. The alternating sweeping direction is the one depicted in Fig. 1. It is astonishing that the alternating sweeping direction makes the non-uniform grid faster than the uniform grid. It can be viewed analogously as painting a wall. Moving a brush alternately from left to right and right to left will fill the wall with color faster than moving the brush in one direction only. Moreover, if the wall is divided into four zones and four painters paint the wall simultaneously, the wall will be filled with color even much faster. In addition, it can be seen from Fig. 9 for the early stage, initiated by a laminar flow calculation, that the computing time is nearly equal in all clustering factors. Thereafter, the rate of residual reduction is distinctly different. This means that the non-uniform grid is severely affected by the turbulent flow behavior. Fortunately, see Fig. 9 again, the alternating sweeping direction can significantly remedy the problem of anisotropy caused by the non-uniform grid.

7. Conclusions

A multigrid technique incorporated with the parallel computing is presented. The FAS multigrid version is adopted in this work with a semi-coarsening technique. The coarsened grid is applied on the yz plane and then the parallel computing is applied for the x-data partitioning. This strategy makes the multigrid method to have the number of grid levels as many as possible. The non-uniform grid degrades the performance of the present algorithm in which the optimum clustering factor used in this work is 0.9. The use of an alternating sweeping direction has an advantage of alleviating the effect of a non-uniform grid. The solver code is developed for the incompressible 3D turbulent flow. The code is validated with the experimental data and the obtained results agree very well.

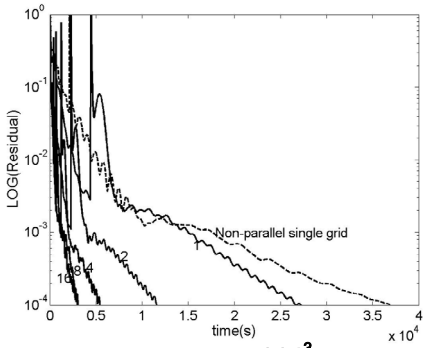

Figure 5. The residual of 64^3 grid points with the different number of processors including the residual of a sequential single grid.

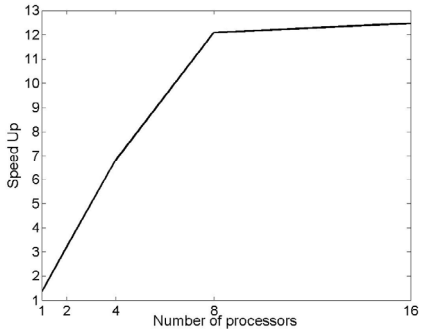

Figure 6. Speed up based on the single grid sequential computing

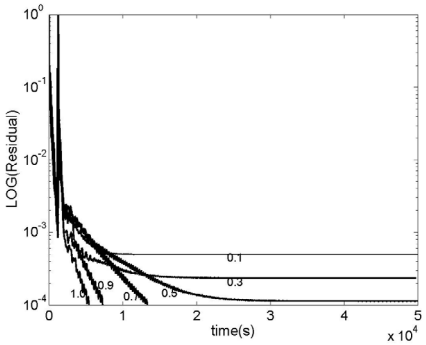

Figure 7. The residual with various clustering factors

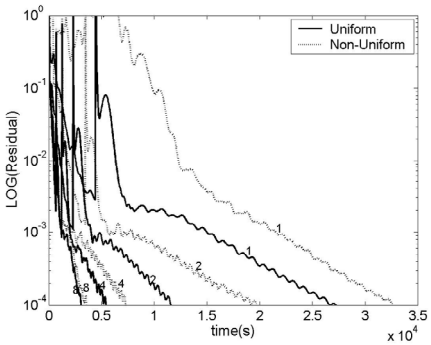

Figure 8. Effect of a non-uniform grid.

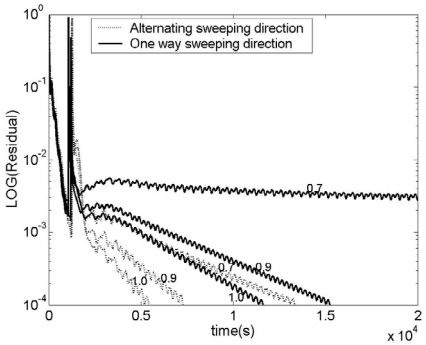

Figure 9. Effect of sweeping direction using 4 processors

8. Acknowledgement

The present work is financially supported by the National Electronics and Computer Technology Center (NECTEC), Thailand. This support is greatly appreciated.

9. References

[1] Montero, R.S., Llorente, I.M., and Salas, M.D., "Robust Multigrid Algorthims for the Navier-Stokes

Equations", J. Com. Physics 173, Academic Press, 2001, pp. 412-432.
[2] Dick, E., Steelant, J., "Coupled solution of the steady compressible Navier-Stokes equations and the $k-\varepsilon$ turbulence equations with a multigrid method", Applied Numerical Mathematics 23, Elsevier Science B.V., 1996, pp. 49-61.
[3] Lien, F.S., Leschziner, M.A., "Multigrid acceleration for recirculating laminar and turbulent flows computed with a non-orthogonal, collocated finite-volume scheme", Compute. Methods Appl. Mech. Enggrg. 118, Elsevier Science S.A., 1993, pp. 351-371.
[4] Drikakis, D., Iliev, O.P., and Vassileva, D.P., "A Nonlinear Multigrid Method for the Three-Dimensional Incompressible Navier-Stokes Equations", J. Com. Physics 146, Academic Press, 1998, pp. 301-321.
[5] Demuren, A.O., "Multigrid acceleration and turbulence models for computations of 3D turbulent jets in cross flow", Int. J. Heat Mass Transfer Vol. 35 No. 11, Pergamon Press Ltd., UK., 1992, pp. 2783-2794.
[6] Llorente, I.M., Prieto-Mati as, M, "A parallel multigrid solver for 3D convection and convection-diffusion problem", Parallel Computing 27, Elsevier Science B.V., 2001, pp. 1715-1741.
[7] Liu, C and Liu, Z., "Multigrid Mapping and Box Relaxation for Simulation of the Whole Process of Flow Transition in 3D Boundary Layers", J. Comp. Physics, Academic Press, Inc., 1995, pp. 325-341.
[8] De Zeeuw, P.M., "Development of semi-coarsening techniques", Applied Numerical Mathematics, Elsevier Science B.V., 1996, pp. 433-465.
[9] Darmofal, D.L., and Siu, K., "A Robust Multigrid Algorithm for the Euler Equation with Local Preconditioning and Semi-coarsening", J. Comp. Physics 151, Academic Press, 1999, pp. 728-756.
[10] Lou, J.Z. and Ferraro, R., "A Parallel Incompressible Flow Solver Package with a Parallel Multigrid Elliptic Kernel", J. Comp. Physics, Academic Press, 1996, pp. 225-243.
[11] Meier, H.F., Alves, J.J.N., and Mori, M., "Comparison between staggered and collocated grids in the finite-volume method performance for single and multi-phase flow", Computers and Chemical Engineering 23, Elsevier Science Ltd., 1999, pp. 247-262.
[12] Peric, M., Kessler, R., and Scheuerer, G., "Comparison of Finite-Volume Numerical Methods with Staggered and Collocated grid", Computers & Fluids Vol. 16, No. 4, Pergamon Press, UK, 1988, pp. 389-403.
[13] Prasad, A.K., and Koseff, J.R., "Reynolds number and end-wall effects on a lid-driven cavity flow", Physics of Fluids A Vol. No.2, American Institute of Physics, 1989, pp. 208-218
[14] Ngiamsoongnirn, K., Juntasaro, E., Juntasaro, V. and Utthayopas, P., "Parallel Computing on the Navier-Stoke Solver", The 7th Annual National Symposium on Computational Science and Engineering, Chulalongkorn University, Bangkok, Thailand, 24-26 March 2003. [On-line] Available: http://cameta.sut.ac.th
[15] Ngiamsoongnirn, K., Juntasaro, E., Juntasaro, V. and Utthayopas, P., "Parallel Computing on the Navier-Stoke Solver with the Multigrid Method" Proceedings of the 17th National Mechanical Engineering Conference, Thavarawadee Hotel, Prachinburi, Thailand, 15-17 October 2003. [On-line] Available: http://cameta.sut.ac.th