Multigrid Implementation in Computation of Turbulent Natural Convection in an Enclosure with Installed Partitions

Kiattisak Ngiamsoongnirn[1*], Varangrat Juntasaro[2], Putchong Utthayopas[3] and Ekachai Juntasaro[4]

[1,4]School of Mechanical Engineering, Institute of Engineering, Suranaree University of Technology, Nakhon Ratchasima, Thailand
[2]Department of Mechanical Engineering, Faculty of Engineering, Kasetsart University, Bangkok, Thailand
[3]Department of Computer Engineering, Faculty of Engineering, Kasetsart University, Bangkok, Thailand
Email: kiatt2000@hotmail.com*

Abstract

This paper presents the numerical simulation of turbulent flow driven by natural convection in a complex domain. The domain is a square enclosure and there are five partitions installed inside the enclosure. The finite volume method with a structural cartesian grid is used in this work. Thus the domain is divided into several simple sub-domains resulting in an easy-to-use of a structured grid. Then all the sub-domains are solved simultaneously by using a parallel computing in which one sub-domain is assigned to one process and the exchange of necessary information among relevant processes are performed by message-passing interface via the MPI instructions. Moreover, in each sub-domain, the multigrid algorithm is used to accelerate rate of convergence. The performance test of parallel computing is investigated by comparing the computing time in the following three cases: parallel computing with one process per one compute node, parallel computing with all processes running in one compute node, and a sequential computing. In case of turbulence modeling and numerical validation, this work uses the k-ω-SST turbulence model of Menter [1] and the numerical results are validated with the experimental data of Ampofo [2].

Introduction

With the advanced computer hardware today, the numerical methods are commonly used in fluid flow and heat transfer calculations on personal computer (PC). In some cases, however, the PC may take a long time in computation and in another cases the main memory of one PC may not be enough to launch. The use of supercomputer is unaffordable in general, with regard to the cost. One alternative viable way is to construct a relatively inexpensive PC cluster, the PCs connected together via the Ethernet switch, for computing in parallel approach. In this case, each PC formed the cluster is called the compute node. The basic idea of parallel computing is to perform the tasks concurrently in expecting that the memory usage and computing time are reduced proportionally to the number of compute nodes used. As well-known that one PC can run several programs simultaneously by means of multitasking operation. There are two types of multitasking: thread- and process-based multitasking. The process is the program that is executed by a processor and the thread is a path of execution within a process. Each process can contain multiple threads in which at least one thread contains in the process as the main thread. This paper concerns only in process-based multitasking. As just stated that one processor (compute node) can run several programs or processes at a time. Therefore this work will assign one set of data to one process. This is a parallel processing or data parallelization, that is, multiple processes or multiple sets of data are executed in parallel.

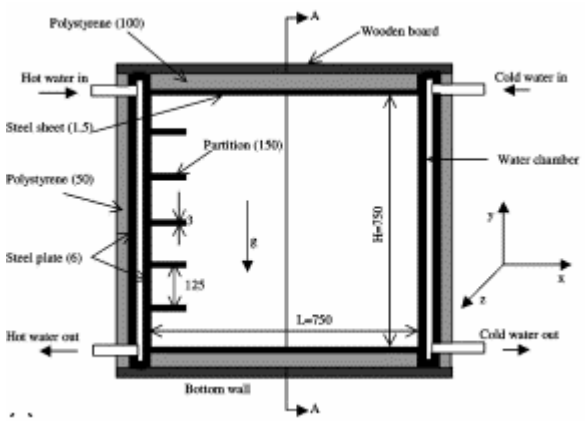


Figure 1. The schemetric diagram of the considered domain (the experimental set-up of Ampofo [2])

The objective of the present work is to divide the complex domain into several rectangular sub-domains and then to take the advantage of using parallel computing which is well-known to be suited for the multiblock technique. Moreover, the multigrid method, well-known as the accelerative algorithm for iterative solvers, is adopted here to accelerate the rate of convergence. The selected problem is the natural convection inside a square enclosure with installed partitions, which is shown in Fig. 1. The detail of experimental set-up can be found in Ampofo [1].

Governing Equations

The Reynolds-averaged Navier-Stokes equation and the time-averaged energy equation are considered in the present work. For a steady incompressible flow, the equations governing fluid flow and heat transfer can be expressed as follows:

the mean continuity equation:

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

the mean momentum equation:

$$\frac{\partial}{\partial x_j}(\rho u_j u_i) = -\frac{\partial p}{\partial x_i} + \frac{\partial}{\partial x_j}\left[\mu\left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i}\right) - \rho\overline{u_i' u_j'}\right]; \\ -\rho g_i \beta (T - T_{ref})$$

and the mean energy equation represented by mean temperature equation:

$$\frac{\partial}{\partial x_j}(\rho u_j T) = \frac{\partial}{\partial x_j}\left[\frac{\mu}{\mathrm{Pr}}\frac{\partial T}{\partial x_j} - \rho\overline{u_j' T'}\right].$$

This averaging process gives rise to the two unknowns: the Reynolds stress $\rho\overline{u_i' u_j'}$ and the turbulent heat flux $\rho\overline{u_i' T'}$. Based on the Boussinesq approximation, the Reynolds stress can be expressed as

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

and with the standard gradient diffusion hypothesis, the turbulent heat flux is in the form

$$\rho\overline{u_j' T'} = \frac{\mu_t}{\sigma_T}\frac{\partial T}{\partial x_j},$$

where g_i represent to g_x and g_y in which g_x is omitted and $g_y = -g = -9.81\, m/s^2$ which is the gravitational acceleration, and for ideal gas $\beta = 1/T_{ref}$.

The Reynolds stress based on the Boussinesq approximation is related to the velocity gradient, the turbulent kinetic energy and the eddy viscosity. The eddy viscosity remains the unknown quantity, which needs further modeling. This leads to the eddy viscosity model. It is modeled relating to the turbulent quantities in which these turbulent quantities possess their own transport equations. There are several zero, one, two or more equations turbulence models proposed in the literature. In this work, the SST-k-ω variance of two-equation turbulence model is used. The detail is described in the following.

The eddy viscosity is defined as

$$\mu_t = \rho \min\left(\frac{k}{\omega}, \frac{ak}{b\|\Omega\|}\right),$$

where $\|\Omega\| = \sqrt{\left(\frac{\partial v}{\partial x} - \frac{\partial u}{\partial y}\right)^2}$, $a = 0.31$, $b = \tanh(\arg_2^2)$, $\arg_2 = \max\left(\frac{2\sqrt{k}}{\alpha^* d_n \omega}, \frac{500\mu}{\rho d_n^2 \omega}\right)$, α^*=0.09; d_n is the nearest normal distance from the wall to the first discrete point. The eddy viscosity is related to the turbulent kinetic energy k and its specific dissipation rate ω which are respectively written in PDE form as

$$\frac{\partial}{\partial x_j}(\rho u_j k) = \frac{\partial}{\partial x_j}\left[(\mu + \sigma_k \mu_t)\frac{\partial k}{\partial x_j}\right] + P_k + G_B - \rho\alpha^* \omega k$$

$$\frac{\partial}{\partial x_j}(\rho u_j \omega) = \frac{\partial}{\partial x_j}\left[(\mu + \sigma_\omega \mu_t)\frac{\partial \omega}{\partial x_j}\right] + \frac{C_\omega}{\mu_t}(P_k + C_3 G_B) \\ -\rho\alpha\omega^2 + 2(1 - B_f)\frac{\rho\sigma_\omega}{\omega}\frac{\partial k}{\partial x_j}\frac{\partial \omega}{\partial x_j}$$

The turbulent production term P_k and the buoyancy production term G_B are respectively defied in two-dimensions as

$$P_k = \mu_t\left[2\left(\frac{\partial u}{\partial x}\right)^2 + 2\left(\frac{\partial v}{\partial y}\right)^2 + \left(\frac{\partial v}{\partial x} + \frac{\partial u}{\partial y}\right)^2\right],$$

$$G_B = -\frac{\mu_t g \beta}{\sigma_T}\frac{\partial T}{\partial y}.$$

The blending function B_f appeared in the ω-equation is defined as $B_f = \tanh(\arg_1^4)$

where

$$\arg_1 = \min\left[\max\left(\frac{\sqrt{k}}{\alpha^* d_n \omega}, \frac{500\mu}{\rho d_n^2 \omega}\right), \frac{4\rho\sigma_{\omega 2} k}{CD_{k\omega} d_n^2}\right] \quad \text{and}$$

$$CD_{k\omega} = \max\left(\frac{2\rho\sigma_{\omega 2}}{\omega}\frac{\partial k}{\partial x_j}\frac{\partial \omega}{\partial x_j}, 10^{-20}\right).$$

The empirical constants of the model, C_ω, α, σ_k and σ_ω, are obtained by blending the model constants of the original $k-\omega$ model in boundary layers, denoted as ϕ_1, with those of the transformed high-Re-number $k-\varepsilon$ model (ϕ_2) in free-shear layers. The resulting relation becomes

$$\phi = B_f \phi_1 + (1 - B_f)\phi_2;$$

the model constants of the original $k-\omega$ model are $\alpha_1 = 0.075$ $\sigma_{k1} = 0.85$, $\sigma_{\omega 1} = 0.5$, and $C_{\omega 1} = 0.533$;

and the model constants of the transformed $k-\varepsilon$ model are
$\alpha_2 = 0.0828$, $\sigma_{k2} = 1.0$, $\sigma_{\omega 2} = 0.856$, and $C_{\omega 2} = 0.44$.

Numerical Method

The governing equations are in the form of partial differential equation and can be written in general transport form as

$$\frac{\partial}{\partial x_j}(\rho u_j \phi) = \frac{\partial}{\partial x_j}\left(\Gamma^{\phi}\frac{\partial \phi}{\partial x_j}\right) + S^{\phi}$$

To solve this equation with the use of computer simulation, it is necessary to convert the partial differential equation form into an algebraic form of equation. This is the so-called discretization process. This paper uses the finite volume formulation to discretize the governing equations based on a structural non-staggered cartesian grid system. The convection and diffusion terms are respectively interpolated with typical first order upwind and second order central schemes. The discretized equations written in the standard finite volume formulation become

$$a_P \phi_P = \sum_{n=E,W,N,S} a_n \phi_n + S^{\phi}.$$

The resulting algebraic equations are solved with a segregated manner in which all algebraic equations are solved separately. Therefore, the SIMPLE algorithm is adopted in this work to couple the continuity and momentum equations. Then each algebraic equation is solved iteratively by an alternating line-by-line TDMA. To stabilize the computation, the source terms in turbulent kinetic energy and its specific dissipation rate equations are divided into two group: normally be negative (N) and normally be positive (P) groups. The modified algebraic equation can be written as

$$a_P \phi_P = \sum_{n=E,W,N,S} a_n \phi_n + S_P^{\phi} + S_N^{\phi}\left(\frac{\phi_P}{\phi_P}\right).$$

Then the negative source terms are lumped in the diagonal coefficient. However, there are some terms that can be both negative and positive during the iteration process. Those terms are G_k and $(\partial k/\partial x_j)(\partial \omega/\partial x_j)$. Therefore, they will be examined in each iteration whether being positive or negative. Unless being positive, they are lumped in the diagonal coefficient. In acceleration of convergence rate, the multigrid algorithm is adopted by solving the system of algebraic equations on hierarchic grids. To do this, one is to solve each equation on hierarchic grids by sequence of equations, and the other is to solve all equations by sequent level of hierarchic grids. The latter approach is well-known to have more efficient than the former approach; therefore, the latter one is used in this paper and the algorithm will be described in the next section.

Multigrid Methodoly

In the multigrid method the computation is carried out on a number of grids set with different grid size *h*, the finest grid is denoted by *h* and each level of the coarse grids is represented by multiplying an integer number, i.e., *2h, 3h, 4h* and so on. The exact solution for any variable ϕ^h on grid level *h* satisfies the following equation:

$$A^h \phi^h = S^h.$$

Unless the approximate solution satisfies the exact solution and boundary conditions, the equation will contain a residual R^h given by

$$A^h \tilde{\phi}^h = \tilde{S}^h - R^h,$$

where the approximate solution is denoted by the superscript '~'. The exact and approximate solutions are related each other with $\phi^h = \tilde{\phi}^h + e^h$, where e^h is the correction of an approximate solution. By subtracting the approximate equation from the exact equation, the result becomes

$$A^h(\tilde{\phi}^h + e_{\phi}^h) - A^h \tilde{\phi}^h = S^h - \tilde{S}^h + R^h.$$

To determine the correction e_{ϕ}^h, the approximate solutions $\tilde{\phi}^h$ on grid level *h* is transferred onto the coarser grid level *2h*:

$$A^{2h}(I_h^{2h}\phi^h + e_{\phi}^{2h}) - \hat{A}\tilde{\phi}^{2h} = \tilde{S}^{2h} - \hat{S}^{2h} + I_h^{2h} R^{2h},$$

this is the restriction process, where the super script '^' denotes the information calculated from the restricted solutions. Moving the known quantities to the right-hand side, the equation is reduced to

$$A(\tilde{\phi}^{2h}) = \hat{C}^{2h} + \tilde{S}^{2h},$$

where
$\tilde{\phi}^{2h} = I_h^{2h}\tilde{\phi}^h + e_{\phi}^{2h}$ and $\hat{C}^{2h} = \hat{A}^{2h}(I_h^{2h}\tilde{\phi}^h) - \hat{S}^{2h} + I_h^{2h} R^h$.

The term $\hat{C}^{2h}$ is kept constant during the iteration. It should be noted that $\hat{S}^{2h}$ and $\tilde{S}^{2h}$ were identical at the first iteration, as the iterations have progressed, they will differ each other resulting in driving a coarse grid iteration process. The restricted residual and the different in source terms behave like the source-driving term. Thus the coarse grid equation is the source-driven procedure. Subsequently, having a required few iterations at coarser grid finished, the correction is calculated using the following formula:

$$e_{\phi}^{2h} = \tilde{\phi}_{new}^{2h} - \tilde{\phi}_{old}^{2h} = \tilde{\phi}_{new}^{2h} - I_h^{2h}\tilde{\phi}^h.$$

Hence the correction is transferred back to the finest grid, this is the prolongation process. Then the finest grid solution is corrected there using the prolonged correction as follows:

$$\tilde{\phi}^h_{new} = \tilde{\phi}^h_{old} + I^h_{2h} e^{2h}_{\phi} .$$

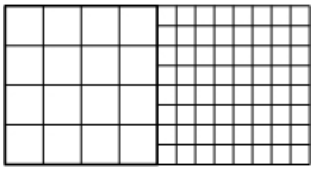
So far, the process have finished for one step, i.e., one multigrid V-cycle is complete (the process pattern looks like the letter 'V'). This version of multigrid method is the full approximation storage (FAS) scheme in which the approximate solution at fine grid is also restricted onto the coarser grid, not only the residual like the correction storage (CS) scheme. The restriction and prolongation are done by bi-linear interpolation for the field variables, but the residuals are restricted by area weight-averaged. In regard to the turbulent quantities correction, having to prolong onto the coarser grid, some mathematical operation have to be modified to avoid being negative value in the correction step. As a result, the correction step for the turbulent quantities is modified as

$$\tilde{\phi}^h_{new} = | \tilde{\phi}^h_{old} + \alpha I^h_{2h} e^{2h}_{\phi} | ,$$

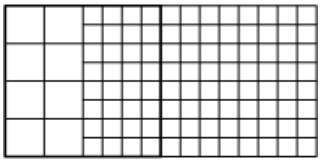
where α is an appropriate small value ranging from 0 to 1 and ϕ stands for k and ω.

Domain Decomposition Technique

It is well-known that the construction of a structured grid suffers from the complexity of the considering domain, i.e., it does not appropriate to generate a single structured grid to cover all region of the complex domain. Even though, it is possible to do, however, this manner gives rise to the complexity in code programming. An efficient way to remedy this problem is a so-called block-structured grid. The block-structured grid is generated by breaking the complex domain into several simple sub-domains, hence, a single structured grid can be generated in each sub-domain. Therefore, this strategy can resolve a complex geometry into simple geometries. The block-structured grid system, moreover, can be broadly classified into two categories: patched grid (see Fig. 2(a)) and overlapping gird (see Fig. 2(b)). The patched grid is the two grid blocks joined together at a common grid line without extended over each other. With overlapping grid, however, the grid block can extend to cover on each other. An advantage of the overlapping grid is its more flexibility in grid generation that can be more easily fitted into the complex geometry than the patched grid. On the other hand, in transferring information between adjacent block, the patched grid is more easily done. The patched grid is used in this paper. In addition, not only the block-structured grid can resolve the problem of complex geometry, but also it provides a natural manner for computing each block simultaneously in terms of parallel computation as will be discribed in the next section.



(a)



(b)

Figure 2. The schematic diagram of (a) a patched grid and (b) an overlapping grid.

The generation of the block-structured grid system is not a difficult task, but its challenging task is how to transfer the information between the adjacent sub-domains (thereafter called block). Each block will contain a separated special cells, a number of control volumes, for storing the information transferred from the adjacent blocks. If the considering block joining with the other blocks, it will detect whether which boundaries being joined with the neighboring blocks and then generates the special cells for those boundaries. Once the special cells have somehow obtained the necessary information from the neighboring blocks, then the direct interpolation is conduced for the special cells and the cells aligning along the boundary belonging to the special cells. However, the direct interpolation is done only for the field variables. For the gradient terms, e.g. $\partial p/\partial x_i$, they are calculated directly from the field variable after conducting the direct interpolation. Besides the field variables, u, v, p, p', T, k, ω are interpolated, it is also necessary to interpolate diagonal coefficients, a_P, of the momentum equations in order to fulfill the coefficients of the pressure correction equation

Parallelization Technique

A usual way in calculation procedure of the block-structured grid is to calculate each block in sequence. During any block is being calculated, the other blocks is idle and waiting for their task queue. This undesired waste time can be revised by using a parallel computing in which all blocks are calculated simultaneously by using a number of processes. In this paper, the processor is the same meaning as the compute node and the process is any program running on the processor. The main task in parallel computing is a partitioning data among processes. Partitioning a single structured grid is straightforward for data parallelization applications. For a composite grid, there are two ways to partition: one is to apply the single grid partitioning technique to one sub-grid at a time and the other is to distribute different sub-grids to different processors. The first approach is applied for serial blocks with parallel data of each block while the second one is applied for parallel blocks. Therefore, the composite grid exhibits both coarse-grain parallelism, i.e. computing on different sub-grids in parallel, and fine-

grain parallelism at the mesh point, i.e., parallel computing within one sub-grid [3]. The coarse-grain parallelism undertakes an advantage of minimal program changes by assigning each entire sub-grid to each processor. Only the grid interconnection is affected. The coarse-grain parallelism is usually referred to as the *domain decomposition* technique and is a popular approach in computational fluid dynamics [3]. In contrast to the coarse-grain parallelism, the fine-grain parallelism is exploited by distributing one sub-grid at a time to all processors, usually by assigning an index set of a region of the grids to the respective processes. This latter approach is a similar manner to the parallel single grid but involves several sets of single grids and it is called the *grid partitioning* technique. The fine-grain parallelism, moreover, requires two sets of data: one for the local array of data and the other for the global array of data. The data of one block is copied from the global data to the corresponding local data, and the computation is performed using the local data in a similar manner to a single block grid. The updated local data is then returned to the global data. Block connectivity is achieved using the global data. After the global data of one block is updated, the other blocks are treated in a sequential manner as shown in Fig. 3 (a) where the solid line of arrows represents the direct exchange of global data between any adjacent blocks and the broken line of arrows denotes the exchange/transfer of data by employing the message-passing interface. Unlike the fine-grain parallelism, the coarse-grain parallelism, where the global data is omitted, requires only the local data and the block computation is performed simultaneously in a parallelization manner as depicted in Fig. 3(b). As shown, in Fig. 3 (b), the processes themselves contain the local data and additional 4 extra cells of data of the interface block. These cells may be called the ghost cells. The ghost cells will be allocated if and only if a block belonging to them interfaces with other blocks. In addition, the dimension of the ghost cells is the same as those of the corresponding faces of neighboring blocks.

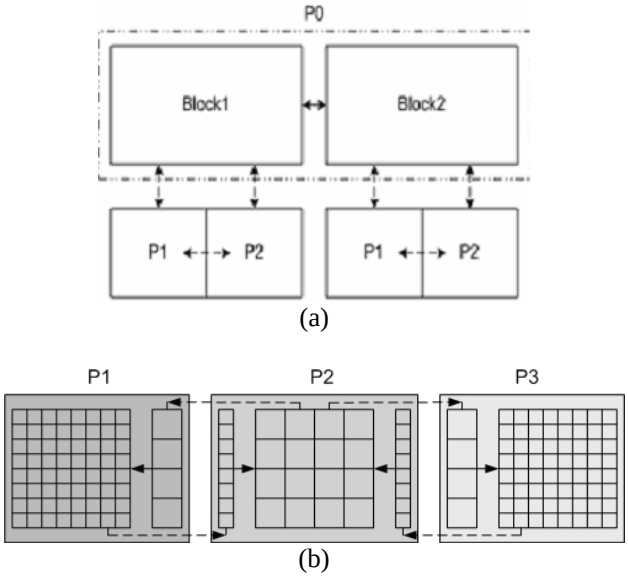


Figure 3. The schematic diagram of (a) coarse-grain parallelism and (b) fine-grain parallelism

The present paper exploits the coarse-grain parallelism or the domain decomposition. The MPI library is adopted in this work for porting a sequential code to a parallel code. The MPI library identifies each process with a unique number, called *rank*. MPI uses the rank to manage a transferring of data among processes: which process receives data from which process and which process send data to which process. It is a good idea to specify the ID number of block with the rank. The block data are firstly created in the data file containing all necessary information, e.g., block ID number, boundary conditions, ID number of which block the current block joins with. As the program has executed, the MPI is started and specifies the ID number of each process with the *rank*, and then all processes read the same input file and find the block ID in the file that matches its rank. If the matching ID has found, then each process reads the necessary information and allocates memory. Then the calculation processes proceed until finishing. Finally, all processes write computed results into the separated data files.

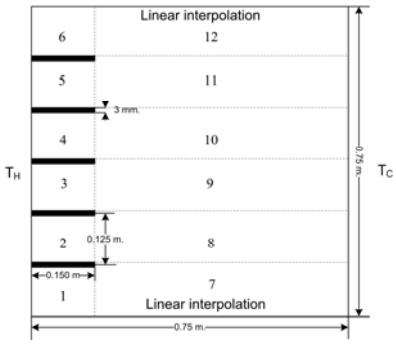


Figure 4. Geometry of considered domain, boundary conditions and domain decomposition (not to scale)

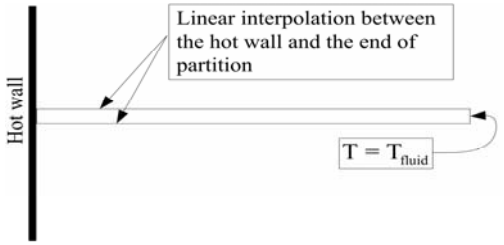


Figure 5. Temperature distribution along the partitions surface (not to scale)

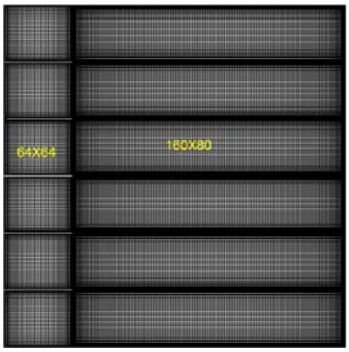


Figure 6. Grid layout for the present calculation

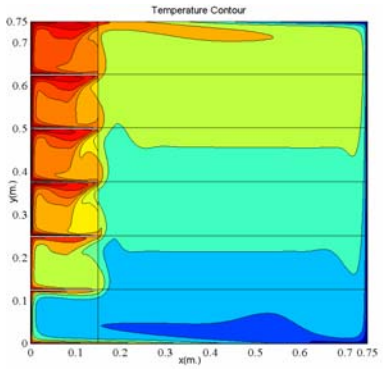


Figure 7. Plot of temperature contour

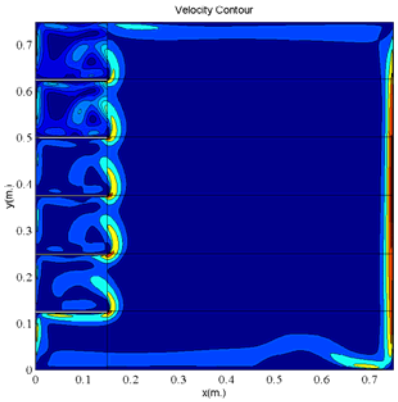


Figure 8. Plot of velocity contour

Problem Descrition and Calculation Detail

The domain considered in this work is depicted in Fig. 4. , that is a square enclosure having five partitions installed inside one. The size of enclosure is 750 mm wide and 750 mm high. The left vertical wall is heated at temperature T_H = 50 ^{0}C and the right one is cooled at T_C = 10 ^{0}C resulting in Ra = 1.58 x 10^9. The temperature distribution along the upper and lower horizontal walls are linearly interpolated between the left and right walls temperature, since the upper and lower walls are made from the highly conductive material in the experiment. The partitions inside the enclosure are arranged in vertically symmetric manner, where the consecutive partitions are 125 mm apart. Each partition is 3 mm thick and 150 mm long. With this partition size, therefore, the temperature at the end of partition can be assumed as that of the surrounding air, as depicted in Fig. 5, and then the temperature distribution along the upper and lower surfaces are interpolated between the temperature of the connecting hot enclosure wall and the end of partition, this practice results in the assumption that the partitions are highly thermal conductivity. As shown in Fig. 4, the domain is decomposed into 12 sub-domains and each domain (block) is denoted by an integer number ranging from 1 to 12. Fig. 6 shows the grid layout for this calculation. The grid points for blocks 1-6 are 64x64 grid points and blocks 7-12 are 160x80 grid points. Grid distribution is generated from an algebraic cubic polynomial function in which the spacing of the first grid point up to the two wall in each direction can be controlled. As a result, high grid density is enforced near the walls and at the interfaces as shown in Fig. 6.

Results and Discussion

The computed contour lines of temperature and resultant velocity, see Fig. 7 and 8 respectively, are fairly smooth across the block interface, this shows the capability and availability of the multiblock technique used. The computed horizontal velocity profiles along centrally consecutive two partitions, see Fig. 9, are validated with the experimental data. The results are in fairly good agreement with the experiment. The discrepancy might be arisen from the difference in partitions arrangement between this work and the experiment, because the information in partitions arrangement of the experiment is not clear in which this work arranges the partitions in vertically symmetric manner.

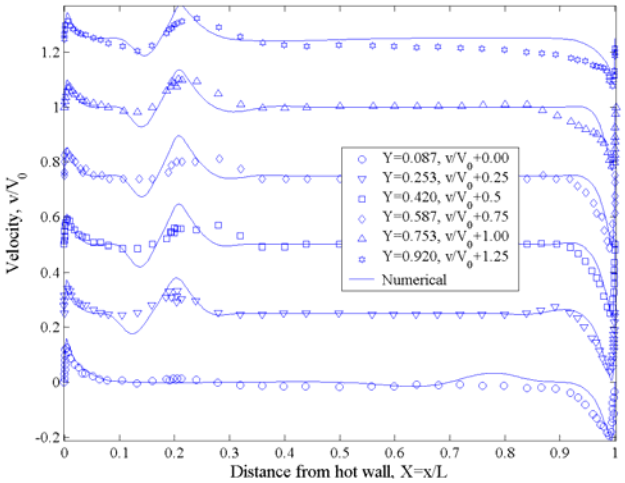


Figure 9. Vertical velocity profiles along central of consecutive partitions at different heights. Symbols stand for experimental data.

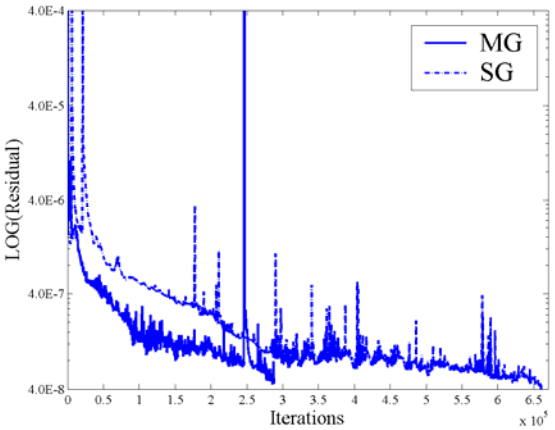


Figure 10. The multigrid efficiency in terms of the number of iterations

Fig. 6 shows the multigrid efficiency in reducing the number of iterations. In parallel performance test, only 5,000 computing steps are performed due to very long computation time at converged state of using one

compute node. In Fig. 11, the residual reduction line, solide line, of parallel computing with one compute node is almost coincident with the one, broken line, of sequential computing (non-parallel), this implies that the time used in exchange necessary information across processes is minor loss compared to the computation time.

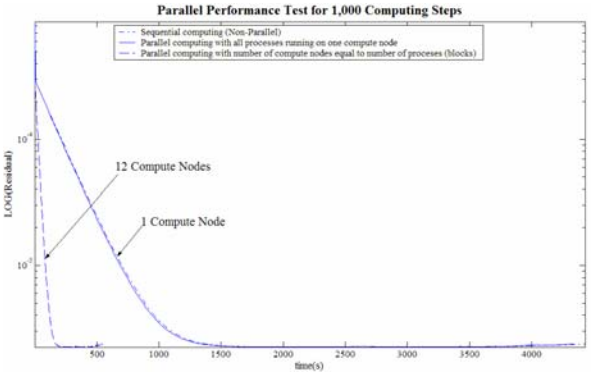


Figure 11. Plot of residual reductions versus computing times of parallel computing and sequential computing

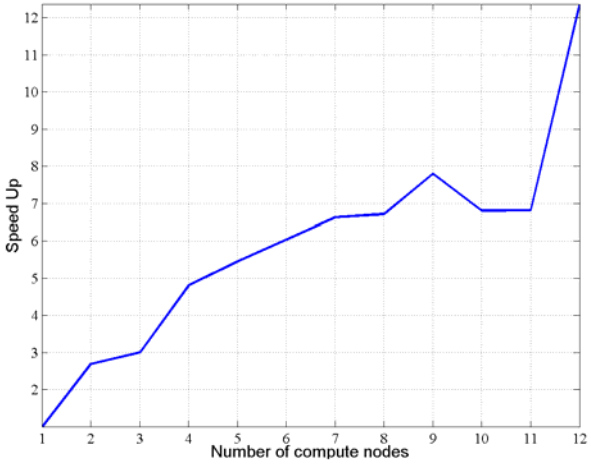


Figure 12. Plot of residual reductions versus computing times of parallel computing by varying the number of compute nodes.

The parameter of interest for parallel computing is the speed up, Sp=T1/Tn, where T1 is the time taken by one compute node in computing entire sub-domains or entire blocks and Tn is the time taken by n compute nodes in computing entire blocks as well. In Fig. 12, it is found that the speed up is increase with increasing the number of compute nodes, but it drops when reaching 10 and 11 compute nodes, why? This situstion can be discribed as follows: the time lost in sending/recieving message across compute nodes through network connection is increase with increasing the number of compute nodes; for using a small number of compute node, the time taken in exchaging information does not dominate; for using 9, 10 and 11 compute nodes, the time taken in computing the remaining blocks left after matching one-by-one between the number of blocks and compute nodes is nearly the same amount because the compute nodes that complete the asigned computation steps will wait for the incomplete compute nodes; and in case of using 6, 7, 8, 9, 10 and 11 compute nodes, there is the same amount of additional task added in one arbitary compute node for all cases (for example, using 6 compute nodes, two tasks will assigned for each compute node and using 10 compute nodes, there will be 2 compute nodes performing a task and one addition task). As shown in Fig. 12, it is found that the maximum speedup is approximately 12.

Conclusions

The numerical simultion of turbulent natural convection in the enclosure with installed five partitions is perfomed by using parallel computing procedure. The multiblock techinqure is used to resolve the problem of domain complexity. It is found that the multiblock technique is well appropriate to the parallel computing. The convergence rate is improved by using the multigrid algorithm. The computed velocity profiles are in fairly agreement with the experimental data. The contour lines of velocity and temperature are pass smoothly through the block interface, this shows the availabilty and capabilty of the multiblock technique used. In test of parallel computing, it is found that the time used in exchanging data across process is very small compared to the computation time. In this paper, the maximum speed up gaind is 12.

Acknowledgments

This work is part of the Thai National Grid Project funded by SIPA, Ministry of Information and Communication Technology, Thailand, and this work is performed on the 16-nodes CAMETA cluster, located at SUT (Thailand), which is financially supported from NECTEC (Thailand). These supports are greatly appreciated.

Refferences

[1] Menter, F.R., 1994. Two-Equations Eddy-Viscosity Turbulence Models for Engineering Applications. AIAA Journal, Vol 32, No 8, pp. 1598-1605.

[2] Ampofo, F., 2005. Turbulent natural convection of air in a non-partitioned or partitioned cavity with differentially heated vertical and conducting horizontal walls. Experimental Thermal and Fluid Science, Vol 29, No 2, pp. 137-157.

[3] J. Rantakokko, 2000. Partitioning Strategies for Structured Multiblock Grids. Parallel Computing, 26, pp. 161-168.