

Numerical and Scalability Analysis of an Unstructured Cartesian Flow Solver

Y.H. Yau^{1,*}, A. Badarudin¹ and P.A. Rubini²

¹ *Department of Mechanical Engineering, Faculty of Engineering, University of Malaya, Kuala Lumpur, Malaysia*

² *Department of Engineering, University of Hull, East Yorkshire, United Kingdom*

*Corresponding Author: Tel: 603 7967 5210, Fax: 603 7967 5317, E-mail: yhyau@um.edu.my

Abstract

This paper describes a systematic approach in building a flow solver for Large Eddy Simulation (LES). Finite volume discretisations of the filtered, incompressible, Navier-Stokes equations were explained. The theory progresses to the description of the step-by-step process (mainly in increasing functionality or capability) in developing a three-dimensional, unstructured Cartesian mesh, parallel code after evaluating numerical factors and available options carried out earlier. This was followed by a presentation of results produced from the simulations of laminar flow, related to the validation of the source codes, which indicates that the flow solver is behaving satisfactorily.

Keywords: Large Eddy Simulation; Flow solver; Three-Dimensional Poisson Solver; Navier-Stokes equation; CFD

Yat H. Yau Ph.D., P.Eng., MASHRAE, MIEM is an Associate Professor and principal M&E consulting engineer at the Department of Mechanical Engineering, University of Malaya, Kuala Lumpur, Malaysia. **A. Badarudin Ph.D.** is a Senior Lecturer and CFD Analyst at the Department of Mechanical Engineering, University of Malaya. **P.A. Rubini Ph.D.** is a Reader and Head of Department at the Department of Engineering, University of Hull, East Yorkshire, United Kingdom.

1. INTRODUCTION

Engineering flows, which are completely laminar, are rare. One way of computing turbulent flow is using Direct Numerical Simulation, which is not very practical considering the amount of computing power of present computers to solve engineering flows, which is proportional to $Re^{9/4}$ where Re is the Reynolds number characterising the flow. Reynolds Averaged Navier-Stokes (RANS) models are much more economical but they are not universally accurate. For example, Wilcox [1] pointed out that for boundary layer flows with adverse pressure gradient, the value of skin friction coefficient, c_f , is over predicted in a number of cases, when using the low Reynolds number version of the $k-\epsilon$ turbulence model proposed by Jones and Launder [2]. In between these two extremes, a technique for simulating large eddies and modelling smaller ones, which is called Large Eddy Simulation (LES), is available. The rationale behind this technique is, firstly, large eddies are the ones carrying most of the turbulent stresses or energy and they are largely responsible for the transport of conserved properties such as momentum and mass and secondly, small scale motions are more isotropic and are thus more suitable for modelling. The study gives an explanation on the governing equations and numerical schemes used in all the test cases. These include:

- finite volume discretisation
- numerical differencing scheme
- treatment of no-slip wall boundaries

Meanwhile, this study also gives a detailed account of the systematic development of the flow solver, which is rarely found in the literature, and includes some validation results for the new code. The novel feature of this study is the development and delivery

of an Adams-Bashforth Crank-Nicolson flow solver employing unstructured Cartesian grids. So far, flow solvers using the same algorithm as the one used here utilised either structured or nonorthogonal unstructured grids. Furthermore, unlike previous LES studies using the Adams-Bashforth Crank-Nicolson method, this paper present a detail and in-depth look on the behaviour of this method with regards to accuracy and parallel performance, which can be rarely found in the literature. The message passing routine for this flow solver allows the computational domain to be divided in an arbitrary manner. An unstructured Cartesian mesh imposes less computation overhead in the calculation of fluxes at cell faces (there is no need to calculate direct-gradient and cross-diffusion terms) compared to fully unstructured grids while maintaining some capability to handle curved surfaces if extra features such as the immersed boundary method is included in the code.

It is important to point out that the tetrahedral and non-orthogonal hexahedral cells frequently used in CFD with face normals, not aligned in the direction of the primary axes of a Cartesian coordinate system incur an extra cost resulting from the computation of the flux components normal to these faces. Numerical difficulties may arise if the quality of the non-orthogonal cells is mediocre. Chalasani *et al.* [3] state that highly skewed and non-orthogonal anisotropic cells near convex and concave regions may result in a poor quality transition from the extruded mesh in the boundary layer and the void filling tetrahedral. A uniform hexahedral mesh does not have to face these issues. In contrast, it offers an advantage over non-orthogonal meshes with regards to the computation cost and also the quality of results in circumstances where the use of a non-orthogonal mesh is not necessary.

Although the test case used in this study permits the use of a structured flow solver, the future plan here is to apply the code developed in this study to tackle domains containing multiple geometric features such as those found in buildings and built environment, which requires an extensive and time consuming grid generation process if a structured flow solver is used.

2. BACKGROUND

The flow solver used in this paper is the second order accurate in time Adams-Bashforth Crank-Nicolson scheme. The method described here is similar to the one by Peyret and Taylor [4]. Other variants of this method are described by Ferziger and Peric [5] and Kim and Moin [6]. The reasons for using this algorithm are, firstly it allows the use of larger time steps than explicit schemes and secondly, the order of temporal accuracy delivered strikes a balance between computing cost and the quality of results. It is also favourable to other popular algorithm such as SIMPLE of Patankar [7], because it has no internal loops. This algorithm is also very popular with LES researchers and is quite straightforward to program. The Adams-Bashforth Crank-Nicolson scheme offer an advantage in that the former requires less calculations per time step compared with the Runge-Kutta and pressure-correction methods.

Any flow solver for the Navier-Stokes equation can be used though some are more efficient than others. For example, Rollet-Miet *et al.* [8] used the Adams-Bashforth Crank-Nicolson scheme though some discrepancies were encountered in the prediction of the normal stress budget. This was attributed to the second order accuracy with respect to time of this method [4]. Haworth and Jansen [9] used the Semi Implicit Method for

Pressure Linked Equations (SIMPLE) algorithm to predict the flow inside a reciprocating internal combustion engine. Okong'o and Knight [10] and Simons and Pletcher [11] used the fourth order Martinelli *et al.* [12] Runge-Kutta method for the prediction of compressible turbulent flows. Bijl *et al.* [13] emphasised that the fourth-order Runge-Kutta method was more efficient than the Backward Differencing Formulations by at least a factor of 2.5 for all error tolerance levels. Park [14] compared the performance of the Adams-Bashforth Crank-Nicolson predictor step in a PISO transient flow solver over the fractional step method, and he concluded that the former performed equally as the latter in predicting turbulent channel flows.

3. FLOW SOLVER AND NUMERICAL SCHEME

A typical cell (not located next to physical or sub-domain boundaries) is shown in Figure 1 as an aid in the explanation of finite volume discretisation in the present research. In three dimensions, there are six plane faces belonging to a cell. These are marked as e, w, n, s, t and b indicating the orientation of cell faces according to compass direction. This typical cell has a volumetric centre at node P while E, W, N, S, T and B indicate the nodes of neighbouring cells according to compass direction. The cell edge length is denoted by Δ while the unit vector normal to a face is \vec{n} . To derive the mass conservation equation, the mass balance inside a control volume must be considered. This can be explicitly stated as follows. The rate of increase of mass in a fluid volume must be equal to the net rate of flow of mass into the fluid volume. The mass conservation equation in integral form dropping the overbars for the resolved quantities is then:

$$\int_S \rho \vec{v} \cdot \vec{n} dS = 0. \quad (1)$$

The starting point for the derivation of the momentum equations is Newton's second law, which states that the rate of change of momentum of a fluid particle must be equal to the sum of forces on the particle. The forces acting on a fluid particle can be divided into two groups. The first one is called surface forces, typically originating from pressure and viscous forces. The second type of force is called body forces; for instance, gravitational force acting on the mass of the fluid particle. The momentum equation with no body force can be written as:

$$\frac{\partial}{\partial t} \int_{\Omega} \rho u_i d\Omega + \frac{3C^n - C^{n-1}}{2} - \frac{D^{n+1} + D^n}{2} + \int_{\Omega} \frac{\partial P^{n+1/2}}{\partial x_i} d\Omega = 0 \quad (2)$$

Where C and D are the convective and diffusive terms respectively. Ω is the volume of a cell. Finite volume discretisation of the first term on the left hand side of Equation (2) yields:

$$\frac{\partial}{\partial t} \int_{\Omega} \rho u_i d\Omega \approx \rho \Delta \Omega \frac{u_i^{n+1} - u_i^n}{\Delta t}. \quad (3)$$

Using discretisation result from Equation (3), Equation (2) is split into an explicit second order (in time) Adams-Bashforth predictor step:

$$\rho \Delta \Omega \frac{u_i^* - u_i^n}{\Delta t} + \frac{3C^n - C^{n-1}}{2} - \frac{D^n}{2} = 0 \quad (4)$$

and an implicit second order (in time) Crank-Nicolson corrector step:

$$\rho\Delta\Omega \frac{u_i^{n+1} - u_i^*}{\Delta t} - \frac{D^{n+1}}{2} + \int_{\Omega} \frac{\partial P^{n+1/2}}{\partial x_i} d\Omega = 0. \quad (5)$$

Applying divergence operator and enforcing mass conservation to Equation (5) yields the Poisson equation for modified pressure in integral form:

$$\int_{\Omega} \frac{\partial^2 P^{n+1/2}}{\partial x_i^2} d\Omega = \frac{\rho\Delta\Omega}{\Delta t} \frac{\partial u_i^*}{\partial x_i} \quad (6)$$

and by the application of Gauss' divergence theorem:

$$\int_S \text{grad } P \cdot \vec{n} dS = \frac{\rho\Delta\Omega}{\Delta t} \frac{\partial u_i^*}{\partial x_i}. \quad (7)$$

Approximating the convective term C in the u momentum equation on a uniform Cartesian mesh gives:

$$\int_S \rho u_i \vec{v} \cdot \vec{n} dS \approx (\rho u_e A) u_e - (\rho u_w A) u_w + (\rho v_n A) u_n - (\rho v_s A) u_s + (\rho w_t A) u_t - (\rho w_b A) u_b. \quad (8)$$

The terms in parentheses in Equation (8) are the mass flow rates across each face

of the cell. To avoid oscillation of pressure, a correction term is added to the interpolated velocity in the calculation of mass flow rates. This technique is similar to the one proposed by Rhie and Chow [15]. Essentially this correction term experiences an increase in absolute value when pressure oscillates rapidly in space and avoids producing a velocity vector field which satisfies mass conservation from a spurious pressure field (for example, a checkerboard pressure field). Approximating the diffusive term D for an incompressible, Newtonian fluid in the u momentum equation on a uniform Cartesian mesh gives:

$$\int_S \mu \text{grad } u_i \cdot \vec{n} dS \approx \mu A \left(\frac{\partial u}{\partial x} \right)_e - \mu A \left(\frac{\partial u}{\partial x} \right)_w + \mu A \left(\frac{\partial u}{\partial y} \right)_n - \mu A \left(\frac{\partial u}{\partial y} \right)_s + \mu A \left(\frac{\partial u}{\partial z} \right)_t - \mu A \left(\frac{\partial u}{\partial z} \right)_b \quad (9)$$

Where μ is the effective viscosity. A secondary contribution to the diffusive term resulting from the spatial variation of viscosity is given by:

$$\int_S \left(\mu \frac{\partial u_j}{\partial x_i} i_j \right) \cdot \vec{n} dS \quad (10)$$

The magnitude of this term is assumed small compared to the term in Equation (9). The diffusive term associated with a cell face that lies on an impermeable wall is the shear stress at the wall. The pressure gradient term in Equation (7) is approximated as follows:

$$\int_S \text{grad } P \cdot \vec{n} dS \approx A \left(\frac{\partial P}{\partial x} \right)_e - A \left(\frac{\partial P}{\partial x} \right)_w + A \left(\frac{\partial P}{\partial x} \right)_n - A \left(\frac{\partial P}{\partial x} \right)_s + A \left(\frac{\partial P}{\partial x} \right)_t - A \left(\frac{\partial P}{\partial x} \right)_b. \quad (11)$$

At this point, the steps involved in the numerical solution of the flow solver algorithm is summarised in the following manner. Equation (4) is solved first to yield the provisional velocity components u_i^* . These values are then used in Equation (7) to give modified pressure P at an intermediate time level $n+1/2$. The last step is the solution of Equation (5) to obtain the velocity components u_i at the new time level $n+1$. This process is repeated until the required number of time steps is achieved. This scheme can be categorised as a projection scheme because the provisional velocities obtained from the explicit step in non-trivial cases do not satisfy mass conservation. The role of pressure is to correct the velocities by projecting out the divergence that produces part of the velocity field.

As pointed out in Section 2, the dissipative nature of second order upwind schemes favour the use of a second order central differencing scheme. While this scheme satisfies conservativeness, there is an upper limit on the cell size since for the solution to be bounded, the Peclet number for any cell must be less than two. It is also worth mentioning that a second order central differencing scheme does not possess the transportiveness property. Another option is to use an upwind scheme with flux limiters. With the central differencing scheme, the velocity component at a cell face (in this case the east face is considered) can be approximated as:

$$u_e \approx \frac{(u_E + u_P)}{2} \quad (12)$$

and the discretisation of a diffusive term in Equation (9) can be written as:

$$\mu A \left(\frac{\partial u}{\partial x} \right)_e \approx \mu A \frac{(u_e - u_p)}{\Delta} \quad (13)$$

and similarly for the pressure gradient term in Equation (11):

$$A \left(\frac{\partial P}{\partial x} \right)_e \approx A \frac{(P_e - P_p)}{\Delta} \quad (14)$$

The notations used in Equations (12), (13) and (14) follow that of Figure 1. Calculation of the shear stresses at a solid wall in Equations (4) and (5) involves first derivatives of the resolved velocity components tangential to the wall. This is discretised using two inner nodes and a parabolic fit to the boundary, resulting in a second order accurate approximation. This is described in more detail in Ferziger and Peric [5]. In this method, the normal stress at a solid wall is set to zero. The pressure at a solid boundary required in the solution of pressure through Equation (7) is approximated by projecting Equation (5) onto the unit normal \vec{n} to the boundary, further details can be found in Peyret and Taylor [4].

Rearrangement of terms in Equations (5) and (7) results in four sets of linear algebraic equations, i.e., one, for each field variable. The generic form of this equation is:

$$A_p \phi_p = \sum_m A_m \phi_m + Q_p \quad (15)$$

Where the subscript P indicates the cell node at which the finite volume discretisation is carried out. Q_P contains all the terms with known variable values and ϕ is the field variable to be solved, in this case they are u , v , w and modified pressure P . The summation in Equation (15) is over all neighbouring nodes. For the filtered momentum equation, A_m in Equation (15) represents the effects of convection and diffusion while for the modified pressure equation, A_m contains the coefficients for the modified pressure P as written in Equation (14). Combining A_P and A_m will give the square sparse coefficient matrix A_c . Equation (15) is suitably modified to include boundary conditions. A suitable technique for the solution of a set of linear algebraic equations in the form of Equation (15) for an unstructured mesh is based on the Krylov subspace class of methods, described in Trefethen and Bau [16] as well as Golub and van Loan [17]. Becker [18] provides a thorough overview of linear algebraic equation solvers.

In this study, the set of linear algebraic equations were solved by the Restarted Generalised Minimum Residual (RGMRES) method of Saad and Schultz [19]. Essentially, this method works on minimising the norm of the residual. At every iteration step, an Arnoldi iteration is performed followed by the solution of a matrix least squares problem, which may be solved using QR factorisation. To improve the rate of convergence of this method, the coefficient matrix needs to be preconditioned.

$$M^{-1}A_c\phi = M^{-1}Q_P \quad (16)$$

Where M^{-1} is a left preconditioned. An example of this type of preconditioning is

the diagonal (or Jacobi) left preconditioner:

$$M = \text{diag}(A) \tag{17}$$

The Adams-Bashforth predictor step of Equation (4) can be calculated explicitly without further iteration since the only unknown value at the provisional time level is the value of the field variable at that particular node.

4. RESULTS AND DISCUSSION

The philosophy underlying the code development stage is to start from simpler units or routines, which make up the flow solver and build on these developed units until a satisfactory working flow solver is obtained. In this case, the point of embarkation was the Poisson solver. Details of the stages are as follows:

4.1 Unstructured, Three-Dimensional Poisson Solver

This task was carried out to develop subroutines that would build lists for the unstructured grid. By using the cell and vertex partition data as well as a list of cell vertices, ghost or halo cells (cells surrounding the internal cells of the sub-domain) are obtained for a process. After determining the ghost cells of a process, subroutines inside this program build lists for the linear algebraic equations solver. Important lists include a face list for the calculation of fluxes at cell faces and a mapping list to transfer the correct data from neighbouring processes and storing them in the correct location. To ease the different tasks of data communication, computation and storage, three indexing systems

for cells were used. Data were stored using the Harwell-Boeing format described in Duff *et al.* [20]. In all the cases described below, one process owns only one sub-domain. The transfer of field variable data for the ghost cells was carried out by the Message Passing Interface (MPI) library.

Figures 2 and 3 show the data transfer routines to pass field variables from one process (internal cells) to another (ghost cells). Both methods use non-blocking communications. No data transfer takes place between two processes if one process has no ghost cells belonging to the other process.

The first successful (non-deadlocking) message passing routine is shown in Figure 2. The sequence of message passing is as follows. The first pass involves data transfer from one process to the next (higher ranking) process as shown in Figure 2. On the second pass, Process 1 transfers data to Process 3 and Process 2 transfers data to Process 4. On the third pass, Process 1 transfers data to Process 4, while the other two processes are idle. These steps are repeated in the reverse direction.

The second method of data transfer sends and receives messages in a cyclic manner. The first pass is shown in Figure 3. The second pass involves data transfer from Process 1 to Process 3, Process 2 to Process 4 and so on. The last pass transfers data from Process 1 to Process 4, Process 2 to Process 1 and so on.

The details of message passing for the second method can be described as follows:

Step 1: The number of cells involved in data transfer between two sub-domains is determined independently by both processes.

Step 2: Cells containing the data to be transferred (required by the ghost cells in the neighbouring sub-domain) are pinpointed by the use of a mapping list and these data are stored in an array.

Step 3: Data are passed to the neighbouring sub-domain using the `MPI_ISEND` and `MPI_IRECV` non-blocking subroutines every time the call to the message passing routine is executed in the flow solver.

Step 4: The receiving sub-domain inserts the received data into the list of field variable for the ghost cells after the call to the message passing routine has been executed in the flow solver.

This process is then repeated for all neighbouring sub-domains. It can be seen that to send a set of data, only one pair of send and receive is required. This cuts down communication time during the time integration loop by avoiding multiple sends and receives to transfer data. The Poisson solver is used (with some modification) in the third step, to solve the pressure equation. Test runs were made using simple geometries to verify results from the code at each stage of the development. The outcome of these test runs are presented here. Test runs were made to ensure that lists have been built correctly and inter-process data transfer has been carried out correctly. This should cut down development time for the unstructured Navier-Stokes solver.

The test case resembles a heat conduction problem. The domain is a cube with an edge length of one meter in all three directions. The number of cells used is ten in each coordinate direction. Heat conductivity was set to 1 W/mK. South, west and bottom boundaries were set to 100K while the north, east and top boundaries were set to 200K. Two processors were used in this test case. Another test run was also made with eight

processors and 64 cells. The results of these two cases are in agreement with each other as shown in Figures 5 and 6. Domain decomposition was done using METIS, a freely available package by Karypis and Kumar [21]. To obtain equal partitions (in terms of the number of cells in each partition), the mesh file is converted to its dual graph file. This graph file is then partitioned.

To solve the linear algebraic equations, the Restarted Generalised Minimum Residual (RGMRES) subroutine of PIM by da Cunha and Hopkins [22] was used. The tolerance for stopping the RGMRES iteration was set to 10^{-10} based on the 2-norm residual values, whereby the diagonal (or Jacobi) left preconditioning was used. It can be seen in Figure 4 that the sub-domain edge-cuts performed by METIS are not straight lines. Note that results are compared to a structured grid code with a conjugate gradient Poisson solver by Mohamad Badry [23].

From Figures 5 and 6 the code under investigation gave results, which are in agreement with the reference data; this gives some confidence that the message passing routine is working as it was intended. Results are compared to a structured grid code with a conjugate gradient Poisson solver by Mohamad Badry [23].

The speed-up obtained from the Poisson solver is shown in Figure 7. For a fixed problem size, the speed-up decreases with increasing number of processes. This can be explained by applying Amdahl's law. The fraction of the serial part of the code must be reduced in order to improve speed-up. A practical way to achieve this is to increase the mesh size but this is not a universal rule since some algorithms have a ratio of communication to computation cost independent of the mesh size. The loop time for three different mesh sizes is shown in Figure 8. Loop time is the total time for running the

simulation minus the initialisation time. The number of processors used is fixed at eight in this case. The line obtained is close to linear. This is to be expected as the amount of computation increases as the number of cells is increased. The slight increase in gradient with the number of cells increased could be due to the growing communication time as the size of data transferred gets larger. The slope of the line can be reduced by cutting down the number of repeated computations, by increasing cache hits. By performing communication that is more efficient; such as carrying out collective communication only once for multiple time steps, will also have a beneficial effect on the loop time.

4.2 Structured, Two-Dimensional Flow Solver

The main reason for developing a structured, two-dimensional flow solver is to have a functional, working Adams-Bashforth Crank-Nicolson solver. This program ensures that the algorithm and numerical schemes have been implemented correctly before the unstructured version is developed. The test case considered was a driven cavity flow at a bulk Reynolds number of 1,000. The geometry for this case is similar to that of the Poisson solver case, described in Figure 4.

The diffusive and convective terms were treated using central differencing scheme. All flow variables were initialised to zero. A ramp input to the speed of the lid was then applied for one second to avoid numerical difficulties during start-up. The steady-state lid speed was set to give a Reynolds number of one thousand based on the edge length of the cube. Time integration was carried out for seventy lid cycle time (the time taken for the lid to travel a distance equal to the edge length of the domain) given

that the difference in field variable values at the monitoring location between two successive time steps is less than 10^{-4} .

Figure 9 shows the presence of the well-known primary vortex and a sequence of secondary vortices at the lower corners of the domain. At this stage, it can be stated that qualitatively correct results are produced by the code. To check the spatial order of accuracy of the Adams-Bashforth Crank-Nicolson scheme numerically, runs involving four different mesh sizes (20^2 , 40^2 , 80^2 and 160^2) were carried out. The magnitude of the vertical velocity component at a point on the vertical centreline bisecting the domain, 0.03 lid length down from the lid was used to determine the spatial order of accuracy. The reference value to calculate the error was obtained using a mesh size of 320^2 .

The change in percentage error as the mesh size is reduced, is shown in Figure 10. It can be seen that the numerically obtained slope tends to the ideal slope for a second-order accurate method, as the mesh is refined. In order to move towards the ideal slope as the cell size is reduced, the time step size must also be decreased. Another reason why the time step size needs to be reduced is due to transportiveness constraint in which case the Peclet number of a cell must not be larger than two cells, but using different time step sizes will result in different levels of spatial discretisation errors. Therefore, the time step size for all mesh sizes was set to a single, sufficiently small value of 0.001s. To eliminate iteration errors as not to influence the results in Figure 10, the 2-norm residual limit in the linear algebraic equation solver was set to 1×10^{-20} . The order of accuracy of a method describes the rate of error reduction either spatially or temporally, but it does not state absolute magnitude of the error. Ferziger and Peric [5] gave a formal derivation of an

equation to approximate the order of the scheme by using values of a field variable on systematically refined grid.

The temporal order of accuracy was also investigated for this code. This is shown in Figure 11 for time step sizes ranging from 0.001s to 0.000125s with the flow time set to 0.01s. A constant grid size of 82×82 cells was utilised. The bulk Reynolds number based on lid length and lid velocity magnitude was set to 1000. To isolate iteration error from the temporal discretisation error, the 2-norm residual limit was set to a very small value of 1×10^{-20} in the linear algebraic equation solver. By using the same grid size for all values of time step sizes, the spatial discretisation error cancels out and is isolated, allowing a meaningful comparison of temporal discretisation error when the time step size is varied. Data is compared with a reference value obtained from using a time step size of 0.0000625s. For time step sizes less than 0.0005s, the slope is essentially following the ideal case for a second order accurate method as shown in Figure 11. The spatial discretisation error from the largest time step case, gave a better than expected result since the error level for this case is well below the ideal slope, as shown in Figure 10.

4.3 Unstructured, Three-Dimensional Flow Solver

This program extends the Adams-Bashforth Crank-Nicolson algorithm developed in the previous step to handle unstructured Cartesian grids by utilising the list building subroutines developed in the first step. The addition of the third coordinate direction is straightforward. The communication steps follow directly from the Poisson solver.

The west and east boundaries are assigned as symmetry boundaries, which means, the shear stress is zero on this boundary. The normal stress for a symmetry boundary is not zero but for the purpose of comparing results from this code to those obtained from the structured two-dimensional code above, convective and diffusive fluxes in the momentum equations as well as the pressure gradient in the east-west direction are set to zero. This step is a continuation of the work done in developing the Poisson solver described earlier in this chapter. The momentum equations for the structured grid code developed previously were modified for the unstructured grid, taking into account the third coordinate direction for three-dimensional flows. The set of lists built for the Poisson solver was extended to cater for the numerical solution of the splitted momentum equations via the Adams-Bashforth Crank-Nicolson algorithm.

As illustrated in Figure 12, the spatial discretisation error reduces in a monotonous fashion as the grid size is doubled in both directions. Good agreement with the results of Erturk *et al.* [24] for a laminar flow at a Reynolds number of 1000 based on the length of the lid and lid velocity magnitude was obtained by the finest grid. However, in regions of relatively strong convection, in comparison with diffusion, the differences are still significant.

The speed-up obtained when using up to sixteen processes from a test case with 80^3 cells shown in Figure 13 is quite satisfactory. With eight processes, the speed-up obtained was about seven. It is highly possible that the speed-up will improve by increasing the number of cells in the computation, thus decreasing the serial fraction of the code as given by Amdahl's law.

The change in residual values as a function of iteration number of the linear algebraic equation solver for the first time step is shown in Figure 14. The field variables are initialised to zero in the domain and on the boundaries. The y -axis is on a logarithmic scale whereas a linear scale is maintained on the x -axis. From the start of iteration until about 35 time steps, the reduction of the residual values formed almost a straight line and beyond 35 time steps, a steady average value of the residual at approximately $1.0E-13$ was obtained. This plot helps in determining the maximum number of iterations for the RGMRES linear algebraic equations solver assuming that the successive time steps have a similar residual plot as the first one.

An attempt is carried out to perform overlapping computation and communication with the unstructured flow solver successfully. The original form of the parallel program performed better in terms of wall clock time. Another strategy to improve this parameter is to group smaller messages into one big bundle and to reduce the total message volume. Ding and He [25] realised this approach by employing a shrinking sub-domain method which do not require values of field variables at ghost cells to be updated for every single time step. Memory access time can also be reduced by resizing the coefficient matrix into smaller block matrices to fit cache size.

5. CONCLUSIONS

This paper has described the specific form of the Navier-Stokes equations required to solve large eddies together with some physical insight into the process of filtering. Finite volume discretisation of the splitted momentum equations in the framework of the second-order accurate in time Adams-Bashforth Crank-Nicolson flow

solver has also been carried out successfully. Modifications required near a solid no-slip wall boundary on the turbulence model and differencing scheme has been outlined.

Furthermore, the stages in the development of the Adams-Bashforth Crank Nicolson flow solver have been described in this paper. This was followed by a presentation of results related to the validation of the source codes, which indicates that the flow solver is behaving satisfactorily. Up to this point, only laminar flows have been considered. The results obtained from the present research will be used to investigate the behaviour of the turbulence model by comparing simulation results to available data based on experiments and simulations in the future research.

NOMENCLATURE

a	Element of coefficient matrix
A	Cell face area (m^2)
\vec{A}	Surface vector
A_c	Square, sparse coefficient matrix
A_P	Coefficient of cell with node P
A_m	Coefficients of cells with nodes neighbouring node P
C	Convective terms in the momentum equation
D	Diffusive terms in the momentum equation
H	Domain length (m)
H	Convective term
K	Number of subdomains from mesh partitioning
M	Preconditioning matrix

n	Time level
\bar{n}	Unit normal vector
p	Instantaneous static pressure (N/m ²)
Q_P	Source term
\vec{r}	Displacement vector (m)
S	Surface area encompassing a volume of fluid (m ²)
t	Time (s)
u, v, w	Instantaneous velocity components in x, y, z directions (m/s)
u_i	Instantaneous velocity in tensor notation (m/s)
u'_i	Fluctuating velocity in tensor notation (m/s)
V	Volume of fluid (m ³)
x, y, z	Rectangular Cartesian coordinates (m)
x', y'	Position displacement in rectangular Cartesian coordinates (m)
x_i	Position vector in tensor notation (m)
\bar{x}	Position vector (m)
\bar{x}'	Position displacement vector (m)

Greek

Δx	Incremental change in length (m)
$\Delta \Omega$	Incremental change in volume (m ³)
μ	Dynamic molecular viscosity (kg/ms)
ν	Kinematic molecular viscosity (m ² /s)
ν_T	Smagorinsky eddy viscosity (m ² /s)

ρ	Mass density (kg/m ³)
τ_w	Wall shear stress (N/m ²)
ϕ	Scalar field variable
Ω	Cell volume (m ³)

Superscripts

$*, **$	Temporal intermediate values of a field variable
n	Time level

Subscripts

b	Bottom face of grid cell
B	Bottom cell centre
e	East face of grid cell
E	East cell centre
n	North face of grid cell
N	North cell centre
p	Discrete vortex particle
P	Cell centre enclosed by the e, w, n, s, t, b faces
s	South face of grid cell
S	South cell centre
t	Top face of grid cell
T	Top cell centre

w	West face of grid cell
W	West cell centre

ABBREVIATIONS

ABCN	Adams-Bashforth Crank-Nicolson
CFD	Computational fluid dynamics
LES	Large eddy simulation
MPI	Message passing interface
PDE	Partial differential equation
PIM	Parallel iterative method
RANS	Reynolds averaged Navier-Stokes
RGMRES	Restarted generalised minimum residual
SIMPLE	Semi implicit method for pressure linked equations

ACKNOWLEDGEMENTS

The authors would like to thank the Ministry of Higher Education (MOHE), Malaysia, for partially providing financial support to the research project through SLAB scholarship, PJP FS126/2008B and PJP FS131/2008B research grants. Thanks are extended to Cranfield University High Performance Computing Facility for technical support during critical period of the research. Special thanks are extended to Mr. M.H.H. Mohyi and Behzad Rismanchi at the HVAC & R Lab, University of Malaya for helping to prepare the manuscript.

REFERENCES

- 1 **Wilcox, D. C.** *Turbulence Modeling for CFD*, Second Edition, 2002, DCW Industries, La Cañada, California, USA.
- 2 **Jones, W. P.** and **Launder, B. E.** The Prediction of Laminarization with a Two-Equation Model of Turbulence. *Int.J. Heat and Mass Transfer*, 1972, **15**, 301-314.
- 3 **Chalasani, S., Luke, E. A., Senguttuvan, V. A. and Thompson, D. S.** Assessing Generalized Mesh Quality via CFD Solution Validation. *AIAA Paper 2005-687*, 2005, Reno, Nevada, USA.
- 4 **Peyret, R.** and **Taylor, T. D.** *Computational Methods for Fluid Flow*, 1983, Springer-Verlag, New York, USA.
- 5 **Ferziger, J. H.** and **Peric, M.** *Computational Methods for Fluid Dynamics*, Third Edition, 2002, Springer, Berlin, Germany.
- 6 **Kim, J.** and **Moin, P.** Application of a Fractional Step Method to Incompressible Navier-Stokes Equations. *J. Computational Physics*, 1985, **59**, 308-323.
- 7 **Patankar, S. V.** *Numerical Heat Transfer and Fluid Flow*, 1980, McGraw-Hill, New York, USA.
- 8 **Rollet-Miet, P., Laurence, D. and Ferziger, J.** LES and RANS of Turbulent Flow in Tube Bundles. *Int. J. Heat and Fluid Flow*, 1999, **20**, 241-254.
- 9 **Haworth, D. C.** and **Jansen, K.** Large Eddy Simulation on Unstructured Deforming Meshes: Towards Reciprocating IC Engines. *J. Computers and Fluids*, 2000, **29**, 493-524.
- 10 **Okong'o, N.** and **Knight, D. D.** Accurate Three-Dimensional Unsteady Flow Simulation Using Unstructured Grids. *AIAA Paper 98-0787*, 1998, Reno, Nevada, USA.
- 11 **Simons, T. A.** and **Pletcher, R. H.** Large Eddy Simulation of Turbulent Flows Using Unstructured Grids. *AIAA Paper 98-3314*, 1998, Reno, Nevada, USA.

- 12 **Martinelli, L., Jameson, A. and Grasso, F.** A Multigrid Method for the Navier-Stokes Equations. *AIAA Paper* 86-0208, 1986, Reno, Nevada, USA.
- 13 **Bijl, H., Carpenter, M. H. and Vatsa, V. N.** Time Integration Schemes for the Unsteady Navier-Stokes Equations. *AIAA Paper* 2001-2612, 2001, Anaheim, California, USA.
- 14 **Park, T. S.** Effects of Time-Integration Method in a Large-Eddy Simulation using the PISO Algorithm: Part I – Flow Field. *J. Numerical Heat Transfer, Part A*, 2006, **50**, 3, 249-245.
- 15 **Rhie, C. M. and Chow, W. L.** Numerical Study of the Turbulent Flow Past an Airfoil with Trailing Edge Separation. *AIAA Journal*, 1983, **21(11)**, 1525-1532.
- 16 **Trefethen, L. N. and Bau III, D.** *Numerical Linear Algebra*, 1997, Society for Industrial and Applied Mathematics, Pennsylvania, USA.
- 17 **Golub, G. H. and Van Loan, C. F.** *Matrix Computations*, Third Edition, 1996, Johns Hopkins University Press, Baltimore, Maryland, USA.
- 18 **Becker, D.** *Parallel Unstructured Solvers for Linear Partial Differential Equations*. PhD Thesis, Cranfield University, United Kingdom, 2006.
- 19 **Saad, Y. and Schultz, M. H.** GMRES: A Generalized Residual Algorithm for Solving Non-Symmetric Linear Systems. *SIAM J. Sci. Stat. Comput.*, 1986, **7**, 856-869.
- 20 **Duff, I. S., Grimes, R. G. and Lewis, J. G.** User's Guide for the Harwell-Boeing Collection (Release I). *Technical Report TR/PA/92/86*, 1992, CERFACS, Toulouse, France.
- 21 **Karypis, G. and Kumar, V.** METIS – A Software Package for Partitioning Unstructured Graphs, Partitioning Meshes, and Computing Fill-Reducing Orderings of Sparse Matrices. Version 4.0. Department of Computer Science/Army HPC Research Centre, University of Minnesota, Minneapolis, Minnesota, USA, 1998.

- 22 **da Cunha, R. D.** and **Hopkins, T.** The Parallel Iterative Methods (PIM) Package for the Solution of Systems of Linear Equations on Parallel Computers. *Technical Report 19-94*, Computing Laboratory, University of Kent, Canterbury, United Kingdom.

- 23 **Mohamad Badry, A. B.** *First Year Review Report*, 2004, School of Engineering, Cranfield University, Cranfield, Bedfordshire, United Kingdom.

- 24 **Erturk, E., Corke, T. C.** and **Gökçöl, C.** Numerical Solutions of 2-D Steady Incompressible Driven Cavity Flows at High Reynolds Numbers. *Int. J. Numerical Methods in Fluids*, 2005, **48**, 747-774.

- 25 **Ding, C.** and **He, Y.** A Ghost Cell Expansion Method for Reducing Communications in Solving PDE Problems. In *Proceedings of Supercomputing 2001*, Nov 2001.

FIGURE CAPTIONS

- Figure 1** A typical cell in three dimensions with the notation used for a Cartesian grid following Ferziger and Peric (2002).
- Figure 2** Linear message-passing.
- Figure 3** Cyclic message-passing.
- Figure 4** Computational domains partitioning by METIS.
- Figure 5** Comparison of scalar distribution with reference data (left) and developed code (right) with units in Kelvin.
- Figure 6** Plot of temperature against distance in the Y direction at $x=0.5$, $y=0.45$.
- Figure 7** Speed-up of the parallel Poisson solver compared to the ideal slope.
- Figure 8** Variation of loop time with increasing number of cells.
- Figure 9** Streamlines of the driven cavity flow with units of streamfunction in $\text{kg}\cdot\text{m}/\text{s}$.
- Figure 10** Percentage errors in the velocity magnitude with decreasing cell edge length (normalised with the edge length of the largest cell) and uniform mesh for the structured two-dimensional code.
- Figure 11** Percentage errors in the strength of primary vortex with decreasing time step size (normalised with the largest time step size) and uniform mesh for the structured two-dimensional code.
- Figure 12** u and v velocity profiles along the vertical centreline.
- Figure 13** Speed-ups of the three-dimensional unstructured Adams-Bashforth Crank-Nicolson code.

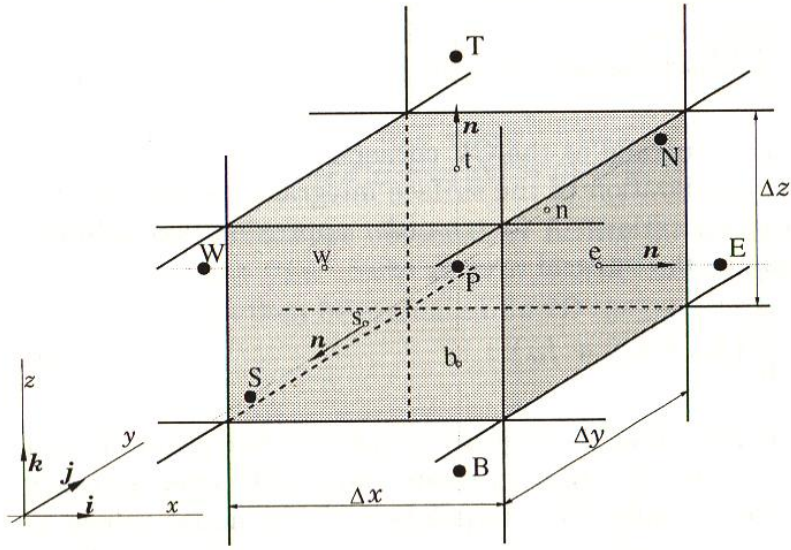


Figure 1 A typical cell in three dimensions with the notation used for a Cartesian grid following Ferziger and Peric (2002).

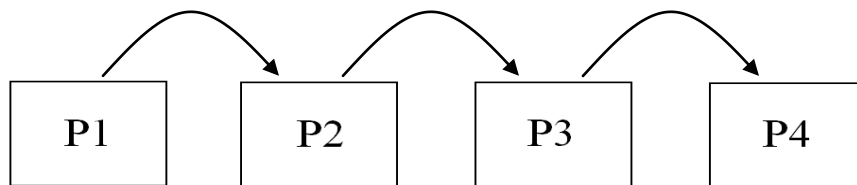


Figure 2 Linear message-passing.

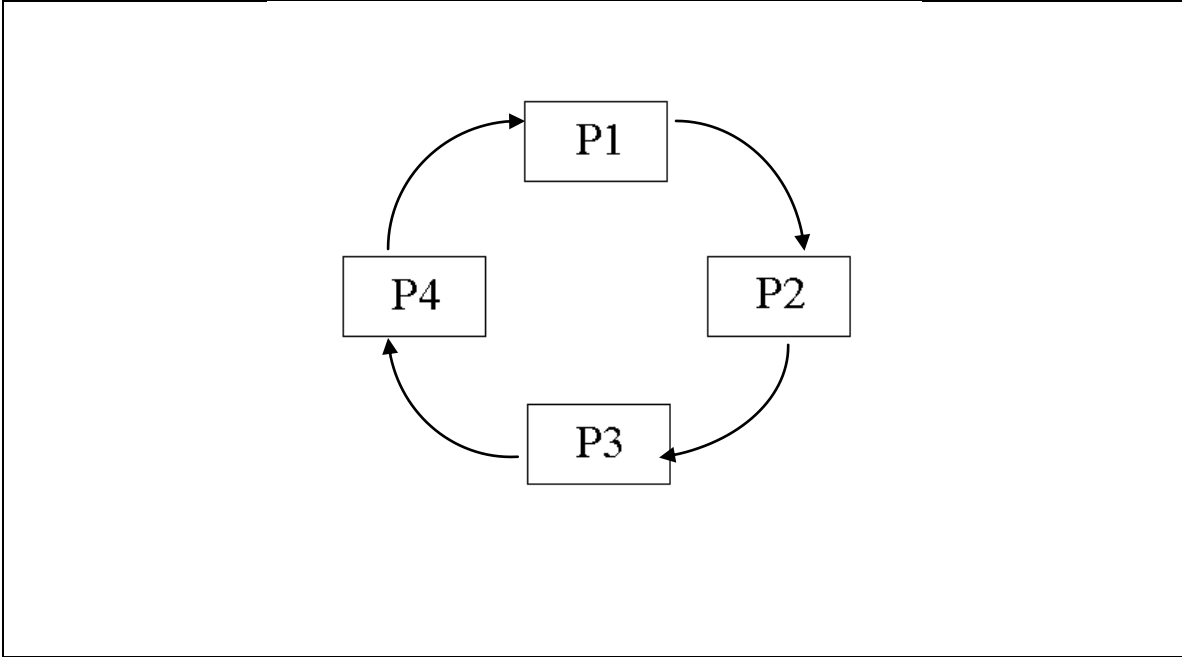


Figure 3 Cyclic message-passing.

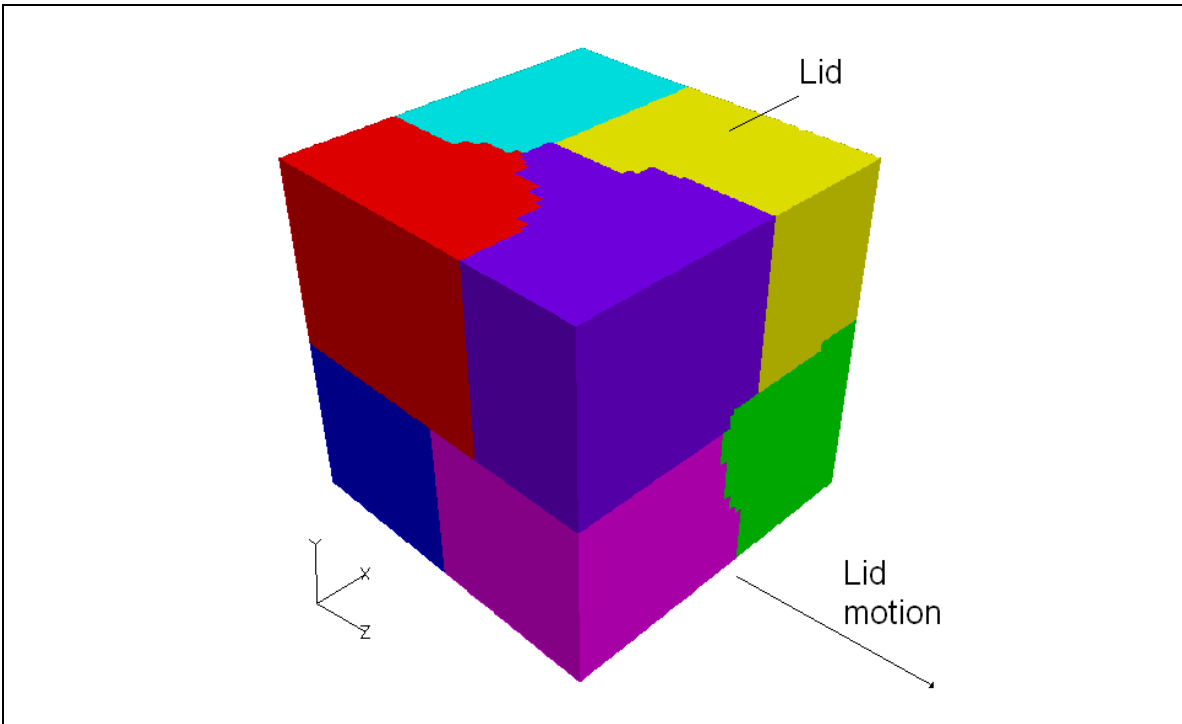


Figure 4 Computational domain partitioning by METIS.

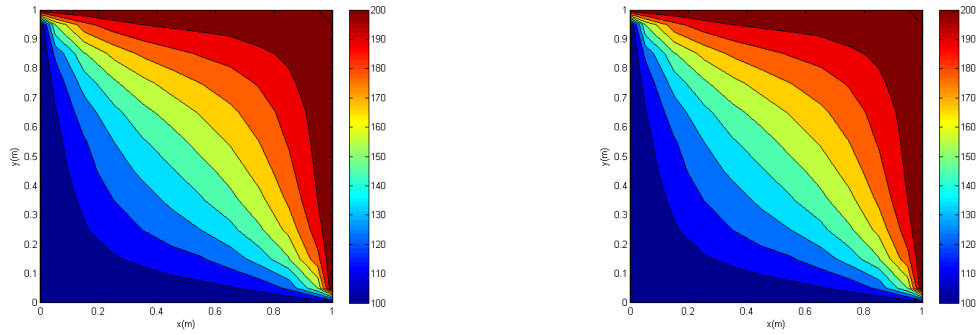


Figure 5 Comparison of temperature distribution with reference data (left) and developed code (right) with units in Kelvin.

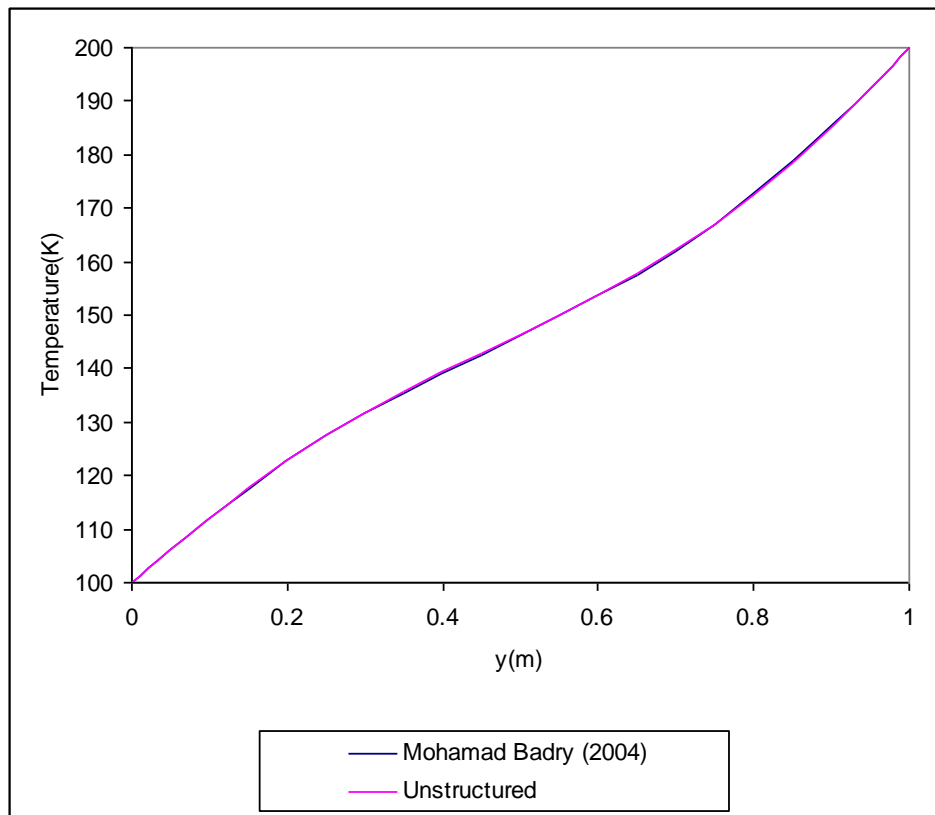


Figure 6 Plot of temperature against distance in the Y direction at $x=0.5$, $y=0.45$.

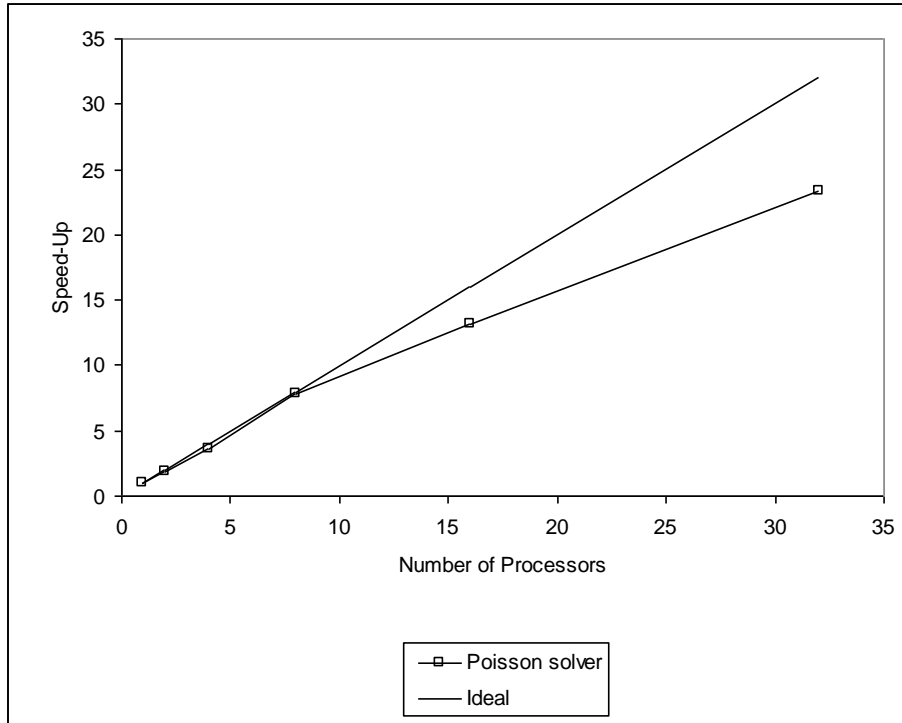


Figure 7 Speed-up of the parallel Poisson solver compared to the ideal slope.

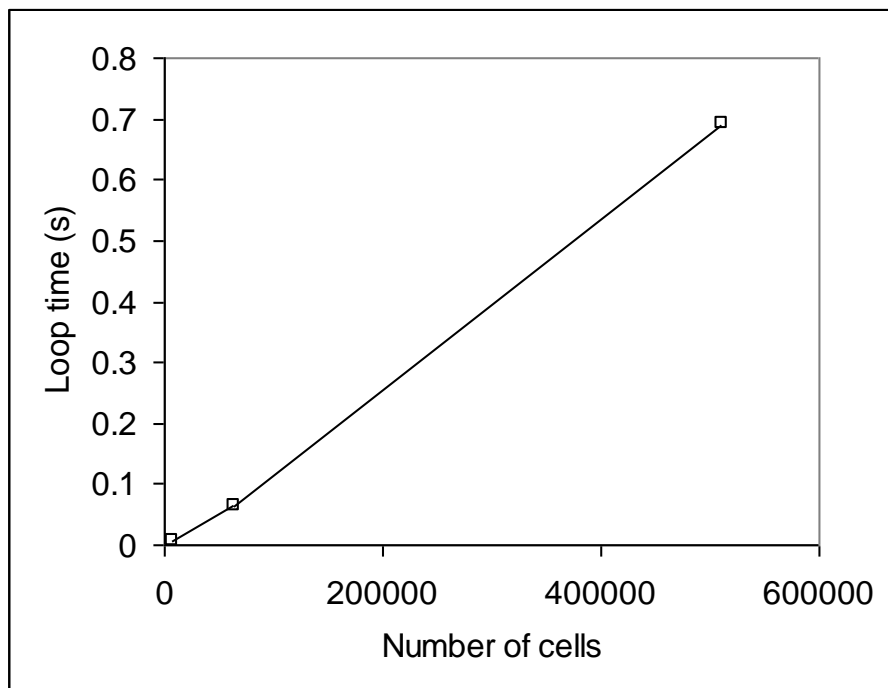


Figure 8 Variation of loop time with increasing number of cells.

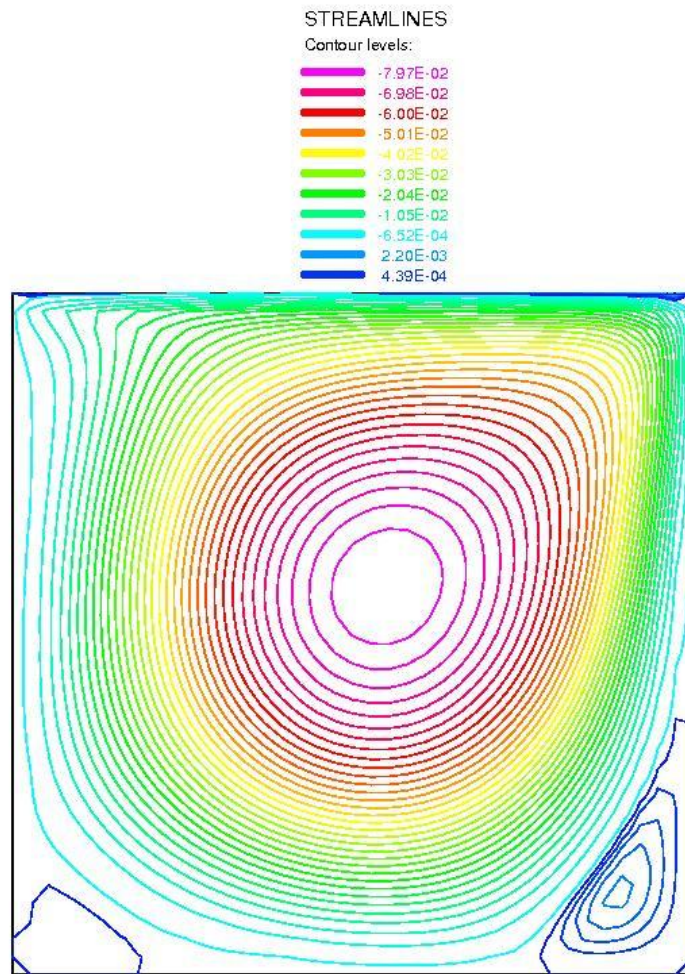


Figure 9 Streamlines of the driven cavity flow with units of streamfunction in kgm/s.

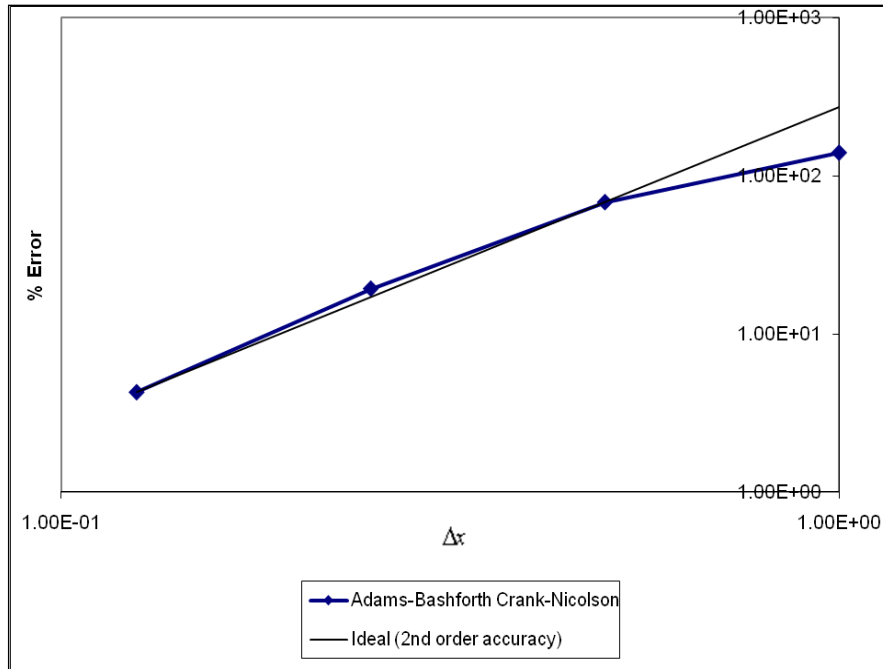


Figure 10 Percentage error in the velocity magnitude with decreasing cell edge length (normalised with the edge length of the largest cell) and uniform mesh for the structured two-dimensional code.

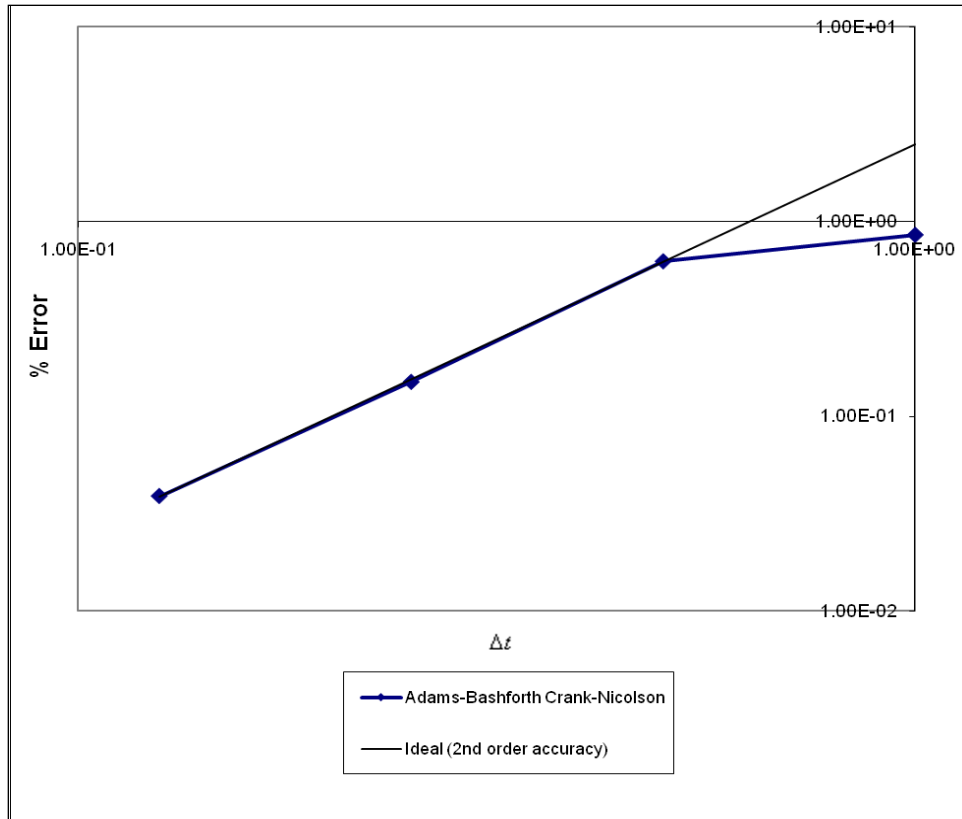


Figure 11 Percentage error in the strength of primary vortex with decreasing time step size (normalised with the largest time step size) and uniform mesh for the structured two-dimensional code.

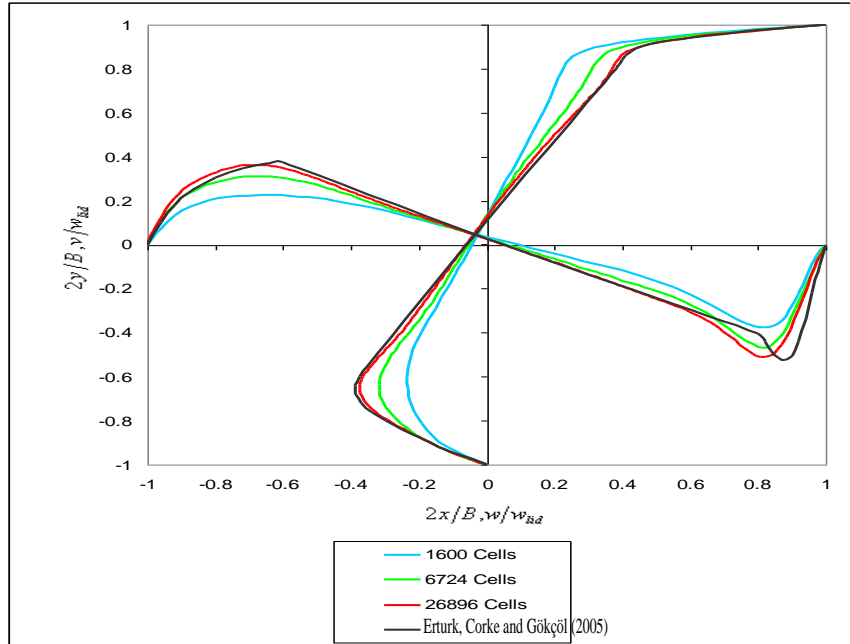


Figure 12 u and v velocity profiles along the vertical centreline.

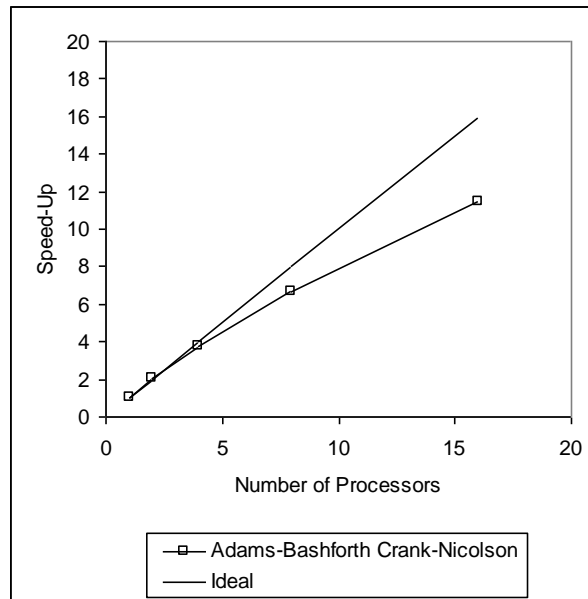


Figure 13 Speed-up of the three-dimensional unstructured Adams-Bashforth Crank-Nicolson code.

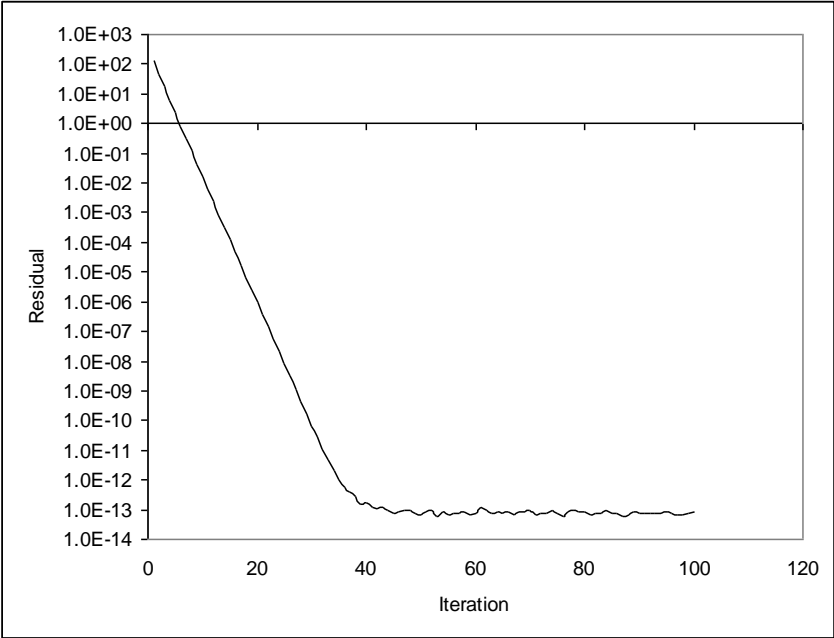


Figure 14 Reduction in the value of residual as iteration progresses.