# A FINITE DIFFERENCE TECHNIQUE FOR SOLVING INCOMPRESSIBLE FLUID FLOW AND ENERGY EQUATIONS ON DISTRIBUTED PARALLEL COMPUTER SYSTEM

Bukhari bin Manshoor
aculty of Mechanical Engineering and
lanufacturing, KUiTTHO Batu Pahat,
Johor
07-4537828
bukhari@kuittho.edu.my

Mat Nawi Wan Hassan, Prof. Dr
Faculty of Mechanical Engineering,
UTM Skudai, Johor
07-5534755

Norma Alias, Dr.
Faculty of Science, UTM Skudai, Johor
07-5534416
norma@mel.fs.utm.my

## BSTRACT

s paper is concerned with the development of an efficient eme for solving the finite difference Navier-Stokes and energy ations using distributed parallel computer system. The nerical procedure is based on SIMPLE (Semi Implicit Method Pressure Link Equations) developed by Spalding. The verning equations are transformed into finite difference forms ng the control volume approach. The hybrid scheme which is nbination of the central difference and up wind scheme is used obtaining a profile assumption for parameter variations ween the grids points. Parallelization method used on this tributed parallel computer system is Domain Decomposition thod (DDM). The accuracy of the parallelization method is ne by comparing with a benchmark solution of a standardized blem related to the two dimensional buoyancy flow in a square closure. The results shown that the distributed parallel mputer system will reduced an execution time to solve the blem about 70% compared to the serial computer.

## eywords

MPLE algorithm, Parallel Algorithm, Domain Decomposition thod, Navier-Stokes Equations.

## INTRODUCTION

e equations governing the fluid dynamics and energy flow have n know for the most part for more than a century and yet have tinued to defy analytical solution. Instead their solutions have ely been obtained by experimental simulations in wind nels, water tables and shock tubes [4]. Now with the ability of anced scientific computer such as distributed parallel puter system, the equations can be solved using the methods omputational fluid dynamic (CFD). Now, it surprising that, d dynamics and heat transfer are contributing to and benefiting n current development in finite difference numerical analysis.

recent years, several finite difference schemes have been posed and develop. Some methods have used the primitive iables, while some have solved the equations in terms of icity and stream function as the dependent variables. The erning equations are often transformed into the non-ensional form. The advantage is that it is more convenient to k with dimensionless variables. The characteristic parameter such as Reynold number, Prandt number and Rayleigh number can be varied independently. Furthermore, by non-dimensionalising the equations, the flow parameters such as velocity and temperature are normalized so that their values can be adjusted to fall between certain prescribed limits. A number of general purpose computer programs using finite difference methods have been developed. Some of these programs using serial computer have relied on works of the Argonne National Laboratory Group, Illinious, USA [5] and methods based on the works at Imperial College, London [8].

This paper deals with a development of an efficient scheme for solving the finite difference Navier-Stokes and energy equations using distributed parallel computer system. The numerical procedure is based on SIMPLE (Semi Implicit Method for Pressure Link Equations) developed by Spalding [2]. As we know, the analysis of an incompressible flow become more complicated and need a high performance computer to solve the problem. One of the problem during to solve the complicated problem on incompressible flow is time constraint. More complicated of the problem means more time should be spend to solve the problem.

To overcome this problem, parallel computer was used and to determine the performance of this parallel computations, the corresponding parallel algorithms was developed and it based on method of parallelization there is Domain Decompositions Method. As the number of the nonlinear simultaneous equations formed after discretisation of the modelling equations is large, an iterative technique is used to update the flow variables. Control volume approach is selected and the matrix formed used to solved using matrix tri-diagonal solver. At the end of this project, the result of simulation using distributed parallel computer system are in terms of how the parallel computer can reduced an execution time compare with the serial computer are presented and discussed.

## 2. NUMERICAL ANALYSIS

### 2.1 Governing equations

Two-dimensional incompressible laminar constant-density flow [7] and energy equation is governed by set of partial differential

equations. The continuity, momentum and energy equations in their primitive form are shown in equation (1-4) where the equation for conservation of mass is given by:

$$\frac{\partial u}{\partial x}+\frac{\partial v}{\partial y}=0 \tag{1}$$

The conservation of momentum in $x$ and $y$ directions are governed by the $u$-momentum equation expressed as:

$$\frac{\partial(uu)}{\partial x}+\frac{\partial(uv)}{\partial y}=\Pr Ra^{-1/4}\frac{H}{L}\frac{\partial^2 u}{\partial x^2}\left(1+\nu_t\right) \\ -\frac{H}{L}\frac{\partial p}{\partial x}+\Pr Ra^{-1/4}\frac{L}{H}\frac{\partial^2 u}{\partial y^2}\left(1+\nu_t\right)-\frac{2}{3}\frac{H}{L}\frac{\partial k}{\partial x} \tag{2}$$

as well as the $v$-momentum equation:

$$\frac{\partial(uv)}{\partial x}+\frac{\partial(vv)}{\partial y}=\Pr Ra^{-1/4}\frac{H}{L}\frac{\partial^2 v}{\partial x^2}\left(1+\nu_t\right)-\frac{L}{H}\frac{\partial p}{\partial y}+ \\ \Pr Ra^{-1/4}\frac{L}{H}\frac{\partial^2 v}{\partial y^2}\left(1+\nu_t\right)-\frac{2}{3}\frac{L}{H}\frac{\partial k}{\partial y}+\frac{H}{L}\Pr Ra^{1/2}T \tag{3}$$

The conservation of energy will express as:

$$\frac{\partial(uT)}{\partial x}+\frac{\partial(vT)}{\partial y}=Ra^{-1/4}\frac{H}{L}\frac{\partial^2 T}{\partial x^2}\left(1+\frac{\Pr}{\sigma_T}\nu_T\right) \\ Ra^{-1/4}\frac{L}{H}\frac{\partial^2 T}{\partial y^2}\left(1+\frac{\Pr}{\sigma_T}\nu_t\right) \tag{4}$$

In the above equations, $u$ and $v$ are the x and y components of the velocity, p is the pressure, $\rho$ and v are the density and viscosity respectively.

## 2.2 Finite Difference Equations

In the development of the control volume approach, the governing partial differential equations are first transformed into divergence force. Let the dependent variables (u, v, and T) are denoted by $\emptyset$, the general differential equation can be written as:

$$div(\rho u\phi)=div(\Gamma\, grad\phi)+S$$

where $\Gamma$ is the diffusion coefficient, or:

$$div\left(\rho u\phi-\Gamma\, grad\,\phi\right)=S$$

When the above finite difference scheme is applied to each momentum equation, the final difference equations can be written as:

$$a_{Pu}u_P=\sum a_{nb}u_{nb}+b_u+\frac{H}{L}(P_P-P_E)y_i \tag{5}$$

$$a_{Pv}v_P=\sum a_{nb}v_{nb}+b_v+\frac{L}{H}(P_P-P_N)x_j \tag{6}$$

The summations are over the four neighboring velocities where *nb* in above equations denotes neighbors.

## 2.3 Correction Equation

In the SIMPLE method, the true pressure field, $P$, which will produce the true velocity fields satisfying the continuity equation is given as:

$$P=P^*+P' \tag{7}$$

where $P'$ is the pressure correction. Similarly, the true velocity fields are given by:

$$u=u^*+u' \\ v=v^*+v' \tag{8}$$

where $u'$ and $v'$ are the velocity corrections. Expressions for these velocity corrections can be obtained from the momentum equations and they are of the forms:

$$u'=\frac{H}{L}\frac{y_i}{a_{Pu}}\left(P'_P-P'_E\right) \tag{9}$$

$$v'=\frac{L}{H}\frac{x_j}{a_{Pv}}\left(P'_P-P'_E\right) \tag{10}$$

The true velocity fields are then obtained by adding the intermediate velocity fields to the velocity corrections. For the control volume shown the true velocity fields can be written as:

$$u_p=u_p^*+\frac{H}{L}\frac{y_i}{a_{pe}}\left(P'_P-P'_E\right) \tag{11}$$

$$v_p=v_p^*+\frac{L}{H}\frac{x_j}{a_{pn}}\left(P'_P-P'_N\right) \tag{12}$$

$$u_w=u_w^*+\frac{H}{L}\frac{y_i}{a_{pw}}\left(P'_W-P'_P\right) \tag{13}$$

$$v_s=v_s^*+\frac{L}{H}\frac{x_j}{a_{ps}}\left(P'_S-P'_P\right) \tag{14}$$

We now turn to the task of deriving a difference equation for the pressure correction using the continuity equation. The integrated continuity equation is given by:

$$F_e-F_w-F_n-F_s=0$$

or:

$$u_p y_i-u_w y_i+v_p x_j-v_s x_j=0 \tag{15}$$

Substitute the expressions given in equations (11) to (14) for the velocity components into equation (15), we have:

$$a_P P'_P=a_E P'_E+a_W P'_W+a_N P'_N+a_S P'_{PS}+b \tag{16}$$

where:

$$a_E=\frac{H}{L}\frac{y_i^2}{a_{pe}}$$

$$a_W=\frac{H}{L}\frac{y_i^2}{a_{pw}}$$

$$a_N = \frac{L}{H}\frac{x_j^2}{a_{pn}}$$

$$a_S = \frac{L}{H}\frac{x_j^2}{a_{ps}}$$

$$a_P = a_E + a_W + a_N + a_S$$

$$b = u_w^* y_i - u_p^* y_i + v_s^* x_j + v_p^* x_j$$

## 2.4 Solution of the Differential Equation

When all the governing equations are transformed into finite difference form, we have a set of algebraic equations which can be solved by any suitable method. For the present calculations, we have employed a line by line iteration method on distributed parallel computer system. Parallelization method used known as Domain Decomposition Method (DDM). Using this method, a grid line is chosen and the values of $\Phi$ for the nodes along the chosen line are assumed to be unknowns. However, the values of $\Phi$ for the nodes along the neighboring lines are assumed to be known and these values are taken from previous iteration. The equations for the grid points along the chosen line are then solved using tridiagonal matrix algorithm (TDMA).

## 2.5 Solution Procedure of the SIMPLE Algorithm

The SIMPLE method proceeds by a cyclic series of guess and correct operations. The important operations are described in the following steps below. The flow chart of the algorithm was showed in Figure 1.

i. Guess the pressure field, $p^*$.
ii. Solve the momentum equation to obtain $u^*$ and $v^*$.
iii. Solve the pressure correction equation to obtain $p'$.
iv. Calculate $p$ form equation " $p = p^* + p'$ " by adding $p'$ to $p^*$.
v. Calculate $u$ and $v$ from their starred values using velocity correction equation.
vi. Solve the discretization equation for other $\phi$'s (for this case, we solve the energy equation to obtain temperature $T$)
vii. Treat the corrected pressure p as new guessed $p^*$, return to step 2 and repeat the whole procedure until a converged solution is obtained.

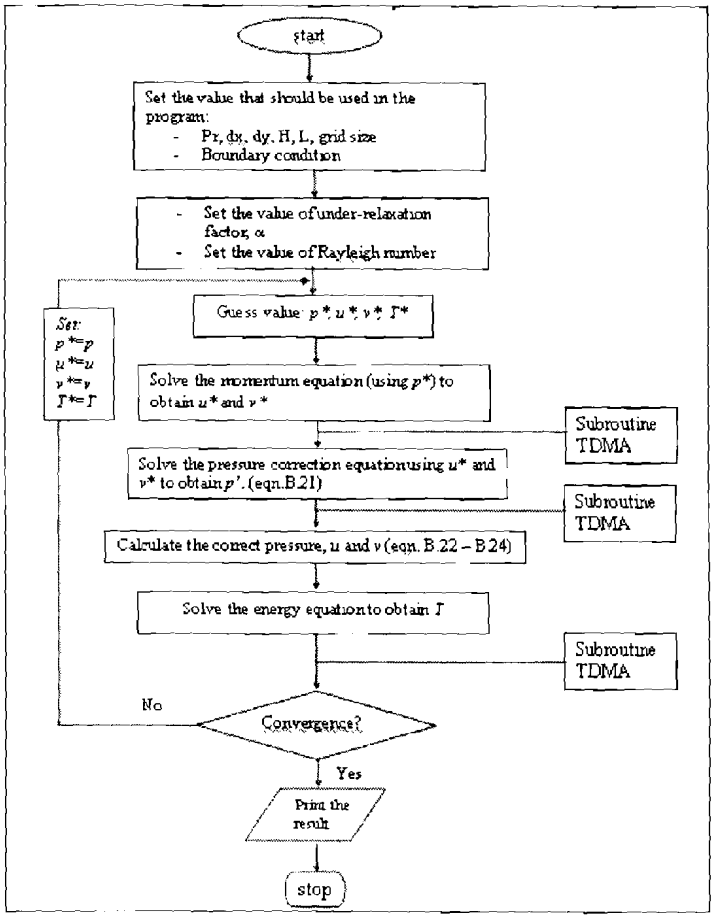


**Figure 1. Flow chart of SIMPLE algorithm**

# 3. PARALLEL IMPLEMENTATION

A parallel implementation can provide a further reduction in computing time. Parallel implementation also makes solution possible to problems that would require too much memory to solve on a single processor. During to solve this problem, the parallel implementation is based on message passing (distributed memory systems) using the PVM software. Portability is ensured because PVM is available on many types of parallel computers.

The implementation uses a layer of subroutines on top of PVM, symbolically denoted by;

- *start*: start entire parallel application
- *stop*: stop parallel application
- *send*: send a message
- *receive*: receive a message

## 3.1 Communication Process

Communication process is the most important process in parallel implementation. As described above, the implementation uses a layer of subroutines on top of PVM, denoted by start, stop, send and receive. For the send and receive subroutines, it consists of

communication process between a data or function that will be send or receive. According to the pseudo code solution in Figure 2, the communication process occurs between the master and slave during to their sending and receiving the data or function.

```
find out if I am MASTER or SLAVES

if I am MASTER
   initialize array
   send each SLAVES starting info and subarray

  do until all SLAVES converge
     gather from all SLAVES convergence data
     broadcast to all SLAVES convergence signal
  end do

  receive results from each SLAVE

else if I am SLAVE
   receive from MASTER starting info and subarray

  do until solution converged
    update time
    send neighbors my border info
    receive from neighbors their border info

   update my portion of solution array

   determine if my solution has converged
     send MASTER convergence data
     receive from MASTER convergence signal
  end do

  send MASTER results
endif
```

**Figure 2. Pseudo code solution**

## 3.2 Communication

Basically this finite difference problem is same with the solution of the problem in this project. From top to bottom of the Figure 3: the one-dimensional vector $X$, where $N$=4; the task structure, showing the 4 tasks, each encapsulating a single data value and connected to left and right neighbors via channels; and the structure of a single task, showing its two inports and outports.

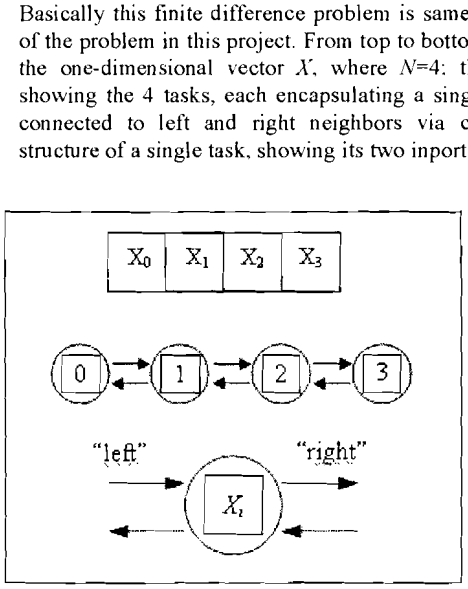


**Figure 3. A parallel algorithms for the finite difference problem**

We first consider a one-dimensional finite difference problem, in which we have a vector $X^{(0)}$ of size N and must compute $X^{(T)}$, where;

$$0 < i < N-1, \quad 0 \le t < T \quad : X_i^{(t+1)} = \frac{X_i^{(t)} + 2X_i^{(t)} + X_{i+1}^{(t)}}{4}$$

That is, we must repeatedly update each element of $X$, with no element being updated in step $t+1$ until its neighbors have been updated in step $t$. A parallel algorithm for this problem creates N tasks, one for each point in $X$. The $i_{th}$ task is given the value $X^{(0)}$ and is responsible for computing, in $T$ steps, the values $X_i^{(1)}, X_i^{(2)}, ..., X_i^{(T)}$.

Hence, at step $t$, it must obtain the values $X_{i-1}^{(t)}$ and $X_{i+1}^{(t)}$ from tasks $i-1$ and $i+1$. We specify this data transfer by defining channels that link each task with "left" and "right" neighbors, as shown in Figure 3, and requiring that at step $t$, each task $i$ other than task 0 and task $N-1$

*i.* sends its data $X_i^{(T)}$ on its left and right outports,

*ii.* receives $X_{i-1}^{(t)}$ and $X_{i+1}^{(t)}$ from its left and right inports, and

*iii.* use these values to compute $X_i^{(t+1)}$.

Notice that the $N$ tasks can execute independently, with the only constraint on execution order being the synchronization enforced by the receive operations. This synchronization ensures that no data value is updated at step $t+1$ until the data values in neighboring tasks have been updated at step $t$. Hence, execution is deterministic.

```
C   broadcast data to slaves

    call pvmfinitsend (PVMDEFAULT, info)
    call pvmfpack (INTEGER4, nproc, 1, 1, info)
    call pvmfpack (INTEGER4, tids, nproc, 1, info)
    call pvmfpack (INTEGER4, n, 1, 1, info)
    call pvmfpack (REAL8, data, n, 1, info)
    msgtype = 1
    call pvmfmcast (nproc, tids, msgtype, info)

C   wait for results from slaves

    msgtype = 2
    do 30 i = 1,nproc
    call pvmfrecv (-1, msgtype, info)
    call pvmfunpack (INTEGER4, who, 1, 1, info)
    call pvmfunpack (REAL8, result(who+1), 1, 1, info)
    if (who.eq.0)
    then
    write (*,1000) result(who+1), who, (nroc-1)
    else
    write (*,1000) result(who+1), who, 2*(who-1)
30  continue
```

**Figure 4. Algorithm master to send and receive data to a from slaves.**

```
c  receive data from master

   msgtype = 1
   call pvmfrecv (mtid, msgtype, info)
   call pvmfunpack (INTEGER4, nproc, 1, 1, info)
   call pvmfunpack (INTEGER4, tids, nproc, 1, info)
   call pvmfunpack (INTEGER4, n, 1, 1, info)
   call pvmfunpack (REAL8, data, n, 1, info)

c  determine which slave I'm (0...nproc-1)

   do 5 i = 0,nproc
   if (tids(i).eq.mytid) me = i
   continue

c  do calculation with the data

   result = work (me, n, data, tids, nproc)

c  send the result to the master

   call pvmfinitsend (PVMDEFAULT, info)
   call pvmfpack (INTEGER4, me, 1, 1, info)
   call pvmfpack (REAL8, result, 1, 1, info)
   msgtype = 2
   call pvmfsend (mtid, msgtype, info)
```

**Figure 5. Algorithm slaves to receive and send data from and to master.**

Figure 4 and 5 above showed the algorithms for the sending and receiving data from master and slaves.

# 4. DISCUSSION

## 4.1 Validation of the Results

Table 1 to 3 compared the results from the present simulation with the literature results obtained by de Vahl Davis [2]. The results of de Vahl Davis are the standard against which all other codes have been evaluated. Maximum horizontal velocity on the vertical midplane of the cavity, $U_{max}$, maximum vertical velocity on the horizontal midplane of the cavity, $V_{max}$, and an average of Nusselt number was compared at Rayleigh numbers of $10^3$, $10^4$, $10^5$ and $10^6$. The comparison was done between the benchmark results obtained by de Vahl Davis which in serial processor and the present study that is simulation using serial processor and parallel processor or parallel computer.

From the tables, it showed that all these results are in excellent agreement with the benchmark results of de Vahl Davis. Percentage error for the three methods of solution is below than 3% compare with benchmark result. Besides that, the result that was showed in the forms of contour maps of non-dimensional temperature and velocities also was compared with the results that obtained by de Vahl Davis.

**Table 1. Comparison of the numerical result of present study for $U_{max}$**

| Ra | $10^3$ | $10^4$ | $10^5$ | $10^6$ |
|---|---|---|---|---|
| G. de Vahl Davis | 3.649 | 16.193 | 34.620 | 64.593 |
| Present study: | | | | |
| i) Serial processor | 3.652 | 16.163 | 34.871 | 65.812 |
| % error | 0.082 % | 0.185 % | 0.725 % | 1.880 % |
| ii)Parallel processor | 3.592 | 16.376 | 34.852 | 65.847 |
| % error | 1.560 % | 1.131 % | 0.670 % | 1.941 % |

**Table 2. Comparison of the numerical result of present study for $V_{max}$**

| Ra | $10^3$ | $10^4$ | $10^5$ | $10^6$ |
|---|---|---|---|---|
| G. de Vahl Davis | 3.697 | 19.167 | 68.590 | 216.360 |
| Present study: | | | | |
| i) Serial processing | 3.704 | 19.675 | 69.482 | 220.641 |
| % error | 0.189 % | 2.650 % | 1.300 % | 1.978 % |
| ii)Parallel processing | 3.715 | 19.642 | 69.680 | 221.282 |
| % error | 0.487 % | 2.478 % | 1.589 % | 2.275 % |

**Table 3. Comparison of the numerical result of present study for $\overline{Nu}$**

| Ra | $10^3$ | $10^4$ | $10^5$ | $10^6$ |
|---|---|---|---|---|
| G. de Vahl Davis | 1.118 | 2.243 | 4.519 | 8.800 |
| Present study: | | | | |
| i) Serial processing | 1.120 | 2.282 | 4.583 | 8.983 |
| % error | 0.23 % | 1.74 % | 1.42 % | 2.08 % |
| ii)Parallel processing | 1.123 | 2.272 | 4.594 | 9.008 |
| % error | 0.47 % | 1.31 % | 1.67 % | 2.36 % |

## 4.2 Parallel Computing Results

In order to achieve the objective of this project, parallel execution time was studied to determine the performance of the parallel computations. Two methods of solution there are serial computation and parallel computation were used during to obtain the results of the simulation. Table 4 showed the results for both methods of computational solution in term of execution time. Table 5 was showed the tabulated results of computational time and communication time for parallel with domain decomposition method.

**Table 4. Execution time for three computational solutions**

| Ra | Sequential time ($t_{seq}$) | Parallel time ($t_p$) |
|---|---|---|
| $10^3$ | 32.8 s | 9.43 s |
| $10^4$ | 135.75 s | 41.39 s |
| $10^5$ | 2040.26 s | 612.06 s |
| $10^6$ | 163602.04 s | 49080.61 s |

**Table 5. Computational and communication time for parallel computation**

| Ra | $t_{comp}$ | $t_{comm}$ | $t_p$ |
|---|---|---|---|
| $10^3$ | 8.41 s | 1.02 s | 9.43 s |
| $10^4$ | 34.62 s | 6.78 s | 41.39 s |
| $10^5$ | 522.82 s | 89.24 s | 612.06 s |
| $10^6$ | 41923.02 s | 7157.60 s | 49080.61 s |

Other parameter that was used to measure a performance of parallel computations is speed-up and efficiency. From the speed-up, we know that how fast the parallel computer solves the problem under consideration. It is sometimes useful to know how long processors are being used on the computation, which can be found from the efficiency. Table 6 below was showed result for speed-up and efficiency for parallel methods. Figure 6, 7 and 8 showed graphically an execution time, speed-up and efficiency against number of processors for Ra=$10^3$ respectively.

**Table 6. Results for speed-up and efficiency**

| Ra | *Speed-Up* | *Efficiency* |
|---|---|---|
| $10^3$ | 3.478 | 86.95 % |
| $10^4$ | 3.279 | 81.97 % |
| $10^5$ | 3.333 | 83.32 % |
| $10^6$ | 3.333 | 83.32 % |

## 4.3 Discussions

From the results that were obtained, we can see that execution time for parallel computation was decrease compare with sequential computation. By using sequential computation, total execution time that we need to complete our simulation at Rayleigh number $10^6$ is 163602.04 seconds or 2726.7 minutes or 45.45 hours. For parallel computation, we were reduced an execution time for the simulation at Rayleigh number $10^6$ to 49080.61 seconds or 818.01 minutes or 13.63 hours. Compare for both methods of simulations, we got the parallel computation with domain decomposition method is more successful for solve this problem with reducing about 70% of execution time.

From the Figure 6 to 8, we can see an effect of number of processors in parallelization to the execution time, speed-up and efficiency. As we can see, the execution time will decrease with increasing of the number of processors. For the speed-up, it will increase with the increasing of the number of processors. However, the efficiency of a simulation was decrease with an increasing of the number of processors.

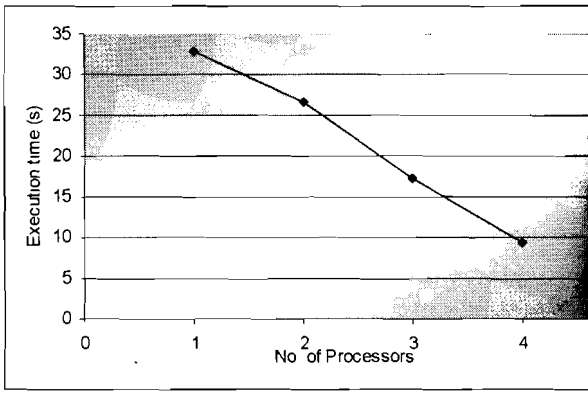


**Figure 6. Execution time against no. of processors for Ra =**

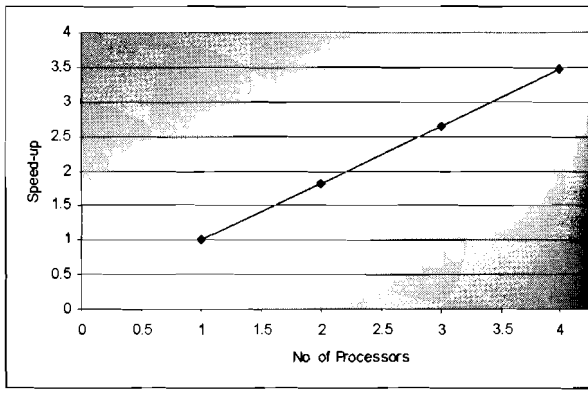


**Figure 7. Speed-Up against no. of processors for Ra = $10^3$**

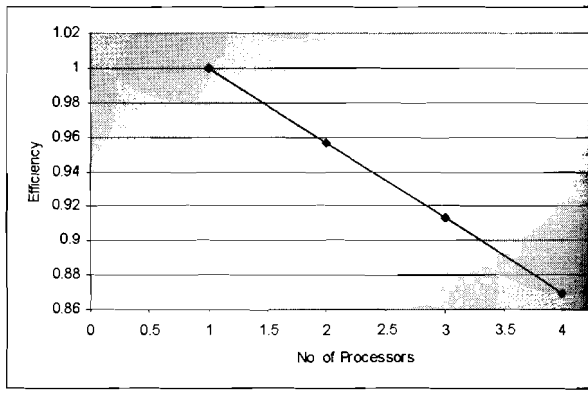


**Figure 8. Efficiency against no. of processors for Ra = $10^3$**

# 5. CONCLUSION

A parallel algorithm has been developed to simulate incompressible flow for the problem of natural convection t occurred in a square cavity with specified boundary conditio The simulations of the incompressible flow using SIMP method on parallel computer are agreement with the benchm result. Thus, the simulation is successful. Percentage errors for

computational solutions which are simulation by serial and lel computer are below than 3% compare with benchmark t by de Vahl Davis.

lelization using distributed parallel computer system with in decomposition method can reduce an exection time to the problem about 70% by using 4 processors. Therefore it proved that clustering personal computers together can ide adequate computing power for large engineering lems.

## ACKNOWLEDGMENTS

ks to Faculty of Science, UTM for allowing us to using their lel Computer Lab and also to each and everyone who have in way contributed to this project.

## REFERENCES

A. W. Date (1985). "Numerical Prediction of Natural Convection Heat Transfer in Horizontal Annulus". Int. J. Heat Mass Transfer.

Davis G. de Vahl (1983). "Natural convection of air in a square cavity: a benchmark numerical solution". Int. Journal Numerical Mech. Fluid (3): 249-264.

D. B. Spalding (1972). "A Novel Finite Difference Formulation for Differential Expressions Involving Both First and Second Derivatives." Int. J. Num. Methods Eng. (3): 551-559.

[4] A. Hayati (1990). "Convective Heat Transfer in Building Energy Analysis", M. Eng Thesis, Faculty of Mech. Engineering, UTM.

[5] S. P. Vanka, G. K. Leaf (1983). "Fully-Coupled Solution of Pressure-Linked Fluid Flow Equations" Argonne National Laboratory, Argonne, Illinois.

[6] H. K. Versteeg and W. Malalasekera. (1995). "An Introduction to Computational Fluid Dynamics." Pearson, Prentice Hall.

[7] M. C. Melaaen (1993). "Nonstaggered Calculation of Laminar and Turbulent Flows using Nonorthogonal Coordinates." Int. J. Num. Heat Transfer: 375-392.

[8] S. V. Patankar (1980). "Numerical Heat Transfer and Fluid Flow." McGraw-Hill Inc, New York.

[9] S. V. Patankar and D. B. Spalding (1972). "A Calculation Procedure for Heat, Mass and Momentum Transfer in 3-Dimensional Parabolic Flows." Int. J. Heat Mass Trasfer.

[10] Dongarra, J. & Eijkhout, U. (2000). "Numerical linear algebra algorithms and software." Journal of Computational and Applied mathematics. 123 (2):489-514.

[11] Geist, A. et al. (1994). "PVM: Parallel Virtual Machine. A Users' Guide and Tutorial for Networked Parallel Computing". Massachusetts: The MIT Press.