ON IMPLEMENTING A NEW EXPLICIT NAVIER-STOKES SOLVER ON A DISTRIBUTED COMPUTING SYSTEM

[1]Norhashidah Hj. Mohd. Ali, [2]Lim Lay Ngor and [3]Rosni Abdullah
[1, 2]School of Mathematical Sciences
Universiti Sains Malaysia
11800 Penang, Malaysia
e-mail: [1]shidah@cs.usm.my

[3]School of Computer Science
Universiti Sains Malaysia
11800 Penang, Malaysia
e-mail: [1]rosni@cs.usm.my

ABSTRACT

A new group explicit inner-outer iterative procedures for solving sparse linear system derived from the *rotated* (skewed) finite difference discretisation of the steady state Navier-Stokes equation was introduced in Ali & Abdullah (1996) and Ali & Abdullah (1999). The Navier-Stokes equation was approached as a stream-function and vorticity formulation, which results in a Poisson equation coupled with a convection-diffusion equation. This method was compared with another outstanding method, i.e. the Alternating Group Explicit (AGE) scheme proposed by Sahimi and Evans [7] where the former method was shown to possess lower computational complexity than the latter since the iterative procedure derived from this discretisation need only involve nodes on half of the total grid points in the solution domain. With the availability of high performance computer technology, it is worthwhile to investigate the versatility of this method by incorporating parallelism into the algorithm as a way to further improve its execution timings. In this paper, we describe the parallel implementation of this method in solving the steady-state Navier-Stokes equation intended for a distributed parallel computer, specifically on a cluster of workstations set up at our Parallel Research Laboratory in the Department of Computer Science, Universiti Sains Malaysia. Using PVM programming environment, the results of some computational experiments are reported and discussed.

KEY WORDS: Numerical methods, *Navier-Stokes* equation, high performance computing, parallel algorithm.

1. INTRODUCTION

Consider the following coupled system of partial differential equations:

$$\nabla^2\psi = -\omega \tag{1.1}$$

$$\nabla^2\omega + \mathrm{Re}(\psi_x\omega_y - \psi_y\omega_x) = -c \tag{1.2}$$

where $x,y \in \Omega = (0,L)x(0,L)$ with a set of conditions for ψ and ω prescribed at the boundary. Here, c and Re (the Reynolds number) are non-negative constants and $\nabla^2 = \frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2}$ is the usual Laplacian operator. Note that if $\mathrm{Re} \neq 0$, then the coupled system represents the two dimensional steady state Navier-Stokes equations which describe the basic viscous, incompressible flow problems. ψ and ω are known respectively as the stream and vorticity functions. Suppose we impose the boundary conditions $\psi = 0$ and $\frac{\partial^2\psi}{\partial\eta^2} = 0$, where η is the normal to the boundary $\partial\Omega$ of Ω, then our problem amounts to solving (1.1) and (1.2) successively with $\psi = 0$ and $\omega = 0$ respectively along $\partial\Omega$.

There are several ways to discretise Equations (1.1)-(1.2) using the finite difference techniques. The aim of this paper is to investigate the applicability of the four-point explicit decoupled group (EDG) method (Ali and Abdullah, 1999) on a distributed computing system in solving this fundamental problem in fluid dynamics. A brief discussion of the finite difference approximations for (1.1)-(1.2) with the specified boundary conditions will be given in Section 2. In Section 3, the development of the EDG scheme for the vorticity equation (1.2) will be presented. The derivation of the algorithm for the stream solutions will then readily follow. Section 4 describes the

parallel numerical algorithm for solving the coupled system (1.1)-(1.2) by incorporating the four-point EDG in its iteration scheme. The concluding remarks is presented in Section 5.

2. FINITE DIFFERENCE APPROXIMATIONS

The discretisation is carried out on a uniform 2D grid with a uniform mesh size h where $x_i = ih$, $y_j = jh$, $i,j = 0,1,...,n$. Observe that if ω is known, then (1.1) is a linear *elliptic* equation in ψ, and if ψ is known, then (1.2) is a linear *elliptic* equation in ω. Suppose $\psi^{(0)}$ and $\omega^{(0)}$ are the initial guesses, we can use the $\omega^{(0)}$ in (1.1) to produce $\psi^{(1)}$. Use this $\psi^{(1)}$ in (1.2) to produce $\omega^{(1)}$. Then use this $\omega^{(1)}$ in (1.1) to produce $\psi^{(2)}$, and this $\psi^{(2)}$ to produce $\omega^{(2)}$ and so on. This indicates that at the grid point (x_i, y_j), alternating sequences of outer iterates can be generated resulting in an inner-outer iteration method [6]. The finite difference approximations of equations (1.1) and (1.2) using the *centred difference* formula at the point (x_i, y_j) will result in the following:

$$-\psi_{i-1,j}^{(k+1)} - \psi_{i,j-1}^{(k+1)} + 4\psi_{ij}^{(k+1)} - \psi_{i,j+1}^{(k+1)} - \psi_{i+1,j}^{(k+1)} = h^2\omega_{ij}^{(k)} \qquad (2.1)$$

$$-\left[1-\sigma\left(\psi_{i,j-1}^{(k+1)} - \psi_{i,j+1}^{(k+1)}\right)\right]\omega_{i-1,j}^{(k+1)} - \left[1+\sigma\left(\psi_{i-1,j}^{(k+1)} - \psi_{i+1,j}^{(k+1)}\right)\right]\omega_{i,j-1}^{(k+1)} + 4\omega_{ij}^{(k+1)}$$
$$-\left[1-\sigma\left(\psi_{i-1,j}^{(k+1)} - \psi_{i+1,j}^{(k+1)}\right)\right]\omega_{i,j+1}^{(k+1)} - \left[1+\sigma\left(\psi_{i,j-1}^{(k+1)} - \psi_{i,j+1}^{(k+1)}\right)\right]\omega_{i+1,j}^{(k+1)} = h^2c_{ij}^{(k)}, \qquad (2.2)$$

here $\sigma = Re/4$ and $i,j = 1,2,...,n-1$. Another type of approximation that can represent the differential equations (1.1) and (1.2) is the cross orientation [5] which can be obtained by rotating the i-plane axis and the j-plane axis clockwise by 45°. With this displacement, equations (2.1) and (2.2) become (2.3) and (2.4) respectively:

$$-\psi_{i-1,j+1}^{(k+1)} - \psi_{i-1,j-1}^{(k+1)} + 4\psi_{ij}^{(k+1)} - \psi_{i+1,j+1}^{(k+1)} - \psi_{i+1,j-1}^{(k+1)} = 2h^2\omega_{ij}^{(k)} \qquad (2.3)$$

$$-[1-\sigma(\psi_{i-1,j-1}^{(k+1)} - \psi_{i+1,j+1}^{(k+1)})]\omega_{i-1,j+1}^{(k+1)} - [1+\sigma(\psi_{i-1,j+1}^{(k+1)} - \psi_{i+1,j-1}^{(k+1)})]\omega_{i-1,j-1}^{(k+1)} + 4\omega_{ij}^{(k+1)}$$
$$-[1-\sigma(\psi_{i-1,j+1}^{(k+1)} - \psi_{i+1,j-1}^{(k+1)})]\omega_{i+1,j+1}^{(k+1)} - [1+\sigma(\psi_{i-1,j-1}^{(k+1)} - \psi_{i+1,j+1}^{(k+1)})]\omega_{i+1,j-1}^{(k+1)} = 2h^2c_{ij}^{(k)}. \qquad (2.4)$$

Clearly it can be seen that the application of (2.3)-(2.4) will result in a large and sparse system with the coefficient matrix being a block matrix depending on the ordering of points taken.

3. THE FOUR POINT EXPLICIT DECOUPLED GROUP FORMULATION

Assume that the solution at any group of four points on the solution domain is solved using the *rotated* equation (2.4). This will result in a (4x4) system of equations

$$\begin{bmatrix} 4 & -[1-\sigma(\psi_{i-1,j+1}-\psi_{i+1,j-1})] & 0 & 0 \\ -[1+\sigma(\psi_{i,j+2}-\psi_{i+2,j})] & 4 & 0 & 0 \\ 0 & 0 & 4 & -[1-\sigma(\psi_{i,j-1}-\psi_{i+2,j+1})] \\ 0 & 0 & -[1+\sigma(\psi_{i-1,j}-\psi_{i+1,j+2})] & 4 \end{bmatrix} \begin{bmatrix} \omega_{ij} \\ \omega_{i+1,j+1} \\ \omega_{i+1,j} \\ \omega_{i,j+1} \end{bmatrix}$$

$$= \begin{bmatrix} [1-\sigma(\psi_{i-1,j-1}-\psi_{i+1,j+1})]\omega_{i-1,j+1} + [1+\sigma(\psi_{i-1,j+1}-\psi_{i+1,j-1})]\omega_{i-1,j-1} + [1+\sigma(\psi_{i-1,j-1}-\psi_{i+1,j+1})]\omega_{i+1,j-1} + 2h^2c_{ij} \\ [1-\sigma(\psi_{ij}-\psi_{i+2,j+2})]\omega_{i,j+2} + [1-\sigma(\psi_{i,j+2}-\psi_{i+2,j})]\omega_{i+2,j+2} + [1+\sigma(\psi_{ij}-\psi_{i+2,j+2})]\omega_{i+2,j} + 2h^2c_{i+1,j+1} \\ [1-\sigma(\psi_{i,j+1}-\psi_{i+2,j-1})]\omega_{i+2,j+1} + [1+\sigma(\psi_{i,j+1}-\psi_{i+2,j-1})]\omega_{i,j-1} + [1+\sigma(\psi_{i,j-1}-\psi_{i+2,j+1})]\omega_{i+2,j-1} + 2h^2c_{i+1,j} \\ [1-\sigma(\psi_{i-1,j+2}-\psi_{i+1,j})]\omega_{i+1,j+2} + [1+\sigma(\psi_{i-1,j+2}-\psi_{i+1,j})]\omega_{i-1,j} + [1-\sigma(\psi_{i-1,j}-\psi_{i+1,j+2})]\omega_{i-1,j+2} + 2h^2c_{i,j+1} \end{bmatrix}$$

$$= \begin{bmatrix} rhs_{i,j} \\ rhs_{i+1,j+1} \\ rhs_{i+1,j} \\ rhs_{i,j+1} \end{bmatrix},$$

which leads to a decoupled system of (2x2) equations whose explicit forms are given by

$$\begin{bmatrix} \tilde{\omega}_{ij} \\ \tilde{\omega}_{i+1,j+1} \end{bmatrix} = \frac{1}{16 - [(1-\sigma(\psi_{i-1,j+1}-\psi_{i+1,j-1}))(1+\sigma(\psi_{i,j+2}-\psi_{i+2,j}))]} X \begin{bmatrix} 4 & 1-\sigma(\psi_{i-1,j+1}-\psi_{i+1,j-1}) \\ 1+\sigma(\psi_{i,j+2}-\psi_{i+2,j}) & 4 \end{bmatrix} \begin{bmatrix} rhs_{ij} \\ rhs_{i+1,j+1} \end{bmatrix} \qquad .....(3.1)$$

and

$$\begin{bmatrix} \tilde{\omega}_{i+1,j} \\ \tilde{\omega}_{i,j+1} \end{bmatrix} = \frac{1}{16 - [(1-\sigma(\psi_{i,j-1}-\psi_{i+2,j+1}))(1+\sigma(\psi_{i-1,j}-\psi_{i+1,j+2}))]} X \begin{bmatrix} 4 & 1-\sigma(\psi_{i,j-1}-\psi_{i+2,j+1}) \\ 1+\sigma(\psi_{i-1,j}-\psi_{i+1,j+2}) & 4 \end{bmatrix} \begin{bmatrix} rhs_{i+1,j} \\ rhs_{i,j+1} \end{bmatrix} \qquad .....(3.2)$$

The computational molecule of Eq. (3.1) and (3.2) are given in Fig. 2 and Fig. 3 respectively:

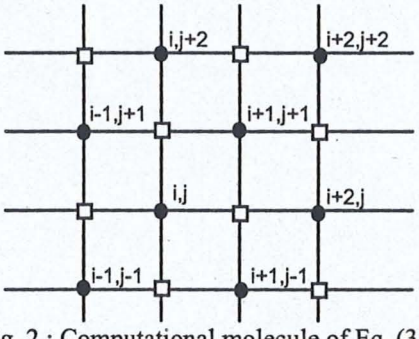

Fig. 2 : Computational molecule of Eq. (3.1)

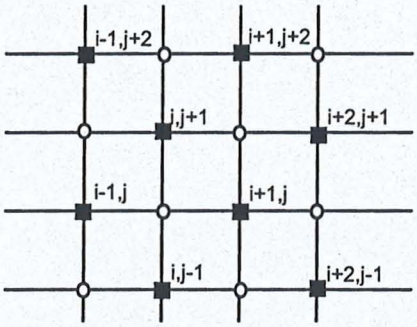

Fig. 3 : Computational molecule of Eq. (3.2)

Note that for both equations, iterative evaluation of points from each group requires contribution of points only from the same group. This means the iteration of points for the vorticity solutions from Eq. (3.1) can be carried out by only involving points of type ● only, while the iterations arised from Eq. (3.2) can be implemented by involving points of type ■ only. Due to this independency, the iterations can be carried out on either one of the two type of points, which means we can expect the execution time to be reduced by nearly *half* since iterations are done on only about *half* of the total nodal points.

In summary, the four-point EDG scheme corresponds to iterating the solutions at approximately *half* of the points in the solution domain using either (3.1) or (3.2) until convergence is achieved. If convergence is achieved, evaluate the solutions at the rest of the nodal points *(points of opposite type)* using the *centred difference* formula (2.2). Otherwise, repeat the iteration cycle.

With boundary conditions specified as before, an algorithm can now be formulated to solve the coupled system (1.1) and (1.2):

ALGORITHM 1:

Step 1 Choose h and construct the number of nodal points. Set $\psi_{ij}^{(0)} = \omega_{ij}^{(0)} = 0 = \text{outer\_}\psi_{ij}^{(0)} = \text{outer\_}\omega_{ij}^{(0)}$ as initial approximations for the outer iteration.

Step 2 Generate sequences $\psi^{(k+1)}$ and $\omega^{(k+1)}$ on Ω by the alternating procedure described before for k = 0,1,2,...

Generate $\psi_{ij}^{(k+1)}$ of (2.3) using the four-point EDG inner iterative procedure described in Section 3.0 for a prescribed tolerance ε. (Use the same Eq. (2.4) but replace ω with ψ, c_{ij} with ω_{ij}, and $\sigma = 0$.)

Generate $\omega_{ij}^{(k+1)}$ of (2.4) using the four-point EDG inner iterative procedure for a prescribed tolerance ε .(Here, use the $\psi_{ij}^{(k+1)}$ just obtained previously in the place of ψ, and $\sigma = \text{Re}/4$.)

Store the converged values $\psi_{ij}^{(k+1)}$ in $\text{outer\_}\psi_{ij}^{(m)}$, and $\omega_{ij}^{(k+1)}$ in $\text{outer\_}\omega_{ij}^{(m)}$.

Step 3 Check the convergence of the outer iteration process over the whole mesh points for a prescribed convergence criterion δ .

If convergence is achieved, the numerical solution of the given problem is given by the generated $\text{outer\_}\psi_{ij}^{(m+1)}$ and $\text{outer\_}\omega_{ij}^{(m+1)}$. Otherwise, go back to ***Step 2***.

4. TWO COLOUR STRATEGY FOR PARALLEL IMPLEMENTATION

There are two EDG inner iterative processes involved for each outer iteration; one for ψ and another one for ω . In parallelising this algorithm, the outer iteration is kept to be sequential while the parallelising steps are done inside the inner iterative processes. Since we have a linear elliptic equation in ψ for the first inner iteration, and a linear elliptic equation in ω for the second inner iteration, we will adopt a strategy which parallelises the EDG scheme in each inner iterative process ([1], [2]). For each inner iteration in Step 2, the domain is decomposed into a number of horizontal strips consisting of two rows arranged in the order shown in Figure 4 for the case n = 9. In this case, the block of points can be divided into two-coloured groups (for simplicity, we use grey (G) and white (W)) so that the parallel sweeps may be carried out within each group.

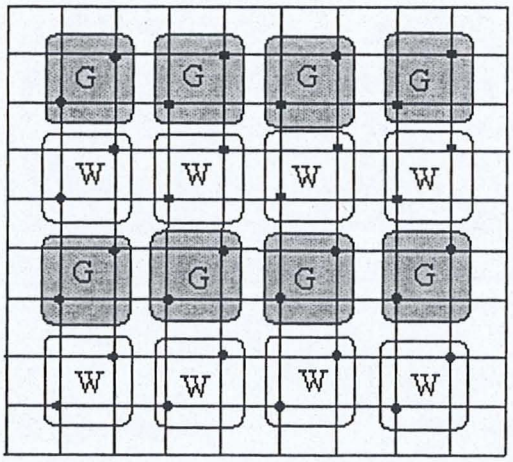

Fig. 4: Decoupling the block of points with 2 colours

Each iteration is split into 2 stages; the first stage updates on the blocks of points of white colour, the second stage updates on the blocks of grey colour. Note that updating the pairs of points in a particular white block needs the values of 4 grid points of the grey colour (of type ●) as depicted in Fig. 5; 2 points in the upper grey blocks and another 2 in the lower grey blocks. Meanwhile, updating the pairs of points in a particular grey block needs the values of 4 grid points of the white colour. Thus,

parallelisation is obtained since the grid points with each colour are decoupled and the updates of a single colour can be computed independently of the equations of the other colour.

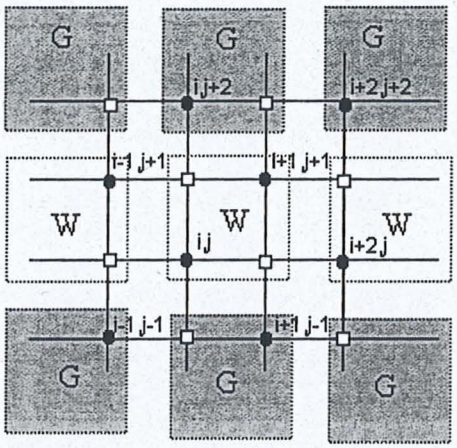

Fig. 5: The points involved in updating a pair of points (of type ●) in a white block

The coloured strips of two rows in Fig. 4 are distributed evenly to the available processors at each stage. If the nodal point in a particular strip is on the boundary of a subdomain held by a particular processor, then the updated values will have to be exchanged with processors holding adjacent subdomains after each stage. Each processor iterates on its own group of points and checks for its own local convergence. After local convergence is achieved, a check for global convergence is made. The evaluation of solutions at the remaining two-pair of points per block will be made only after global convergence is achieved by assigning each strip of two rows to a processor in natural order.

The general outline of the main algorithm just described in solving the coupled equation (1.1)-(1.2) in a message-passing environment can then be summarized as the following:

ALGORITHM 2

Program EDG Navier-Stokes
Declaration Section
Create multiple processes
Calculate the size of work for each processor in each stage
Allocate memory for the parameters used
Initialization Section
BEGIN
DO

10 FOR p=1 to 2
(*When p=0, do stage 1, when p=2 do stage 2*)
IF No. processor >1 THEN send and receive appropriate boundary of ψ for each slave
Calculate all the black points of ψ using Gauss_Seidel (EDG)
Check *local convergence* for inner iteration of ψ (*If converge set the flag to true*)
Each slave sent the value of flag to master
(*If all the flags are true go to 20, else increment of inner iteration for ψ and go to 10*)

20 FOR p=1 to 2
(*When p=0, do stage 1, when p=2 do stage 2*)
IF No. processor >1 THEN send and receive appropriate boundary of ω for each slave
Calculate all the black points of ω using Gauss_Seidel (EDG)
Check *local convergence* for inner iteration of ω (*If converge set the flag to true*)
Each slave sent the value of flag to master
(*If all the flags are true go to 30, else increment of inner iteration of ω and go to 20*)

30 Check *local convergence* for outer iteration
(*If converge set the flag to true*)
Each slave sent the value of flag to master
(*If all the flags are true go to 40, else increment of outer iteration and go to 10*)

40 END DO

IF No. processor >1 THEN update all the boundary of ψ and ω for each slave
Evaluate only once for all the remaining points of ψ and ω using Gauss_Seidel (Centred Difference)
Obtain timings and iteration counts from each process
Kill all child processes and Summing up results

END

5. NUMERICAL EXPERIMENTS AND DISCUSSION OF RESULTS

To test the parallel algorithm just described, we use the following model problem

$$\nabla^2\psi = -\omega \tag{5.1}$$

$$\nabla^2\omega + \mathrm{Re}\left(\psi_x\omega_y - \psi_y\omega_x\right) = -1 \tag{5.2}$$

$$\psi(x,0) = \psi(x,1) = \omega(x,0) = \omega(x,1) = 0, \quad 0 \le x \le 1,$$
$$\psi(0,y) = \psi(1,y) = \omega(0,y) = \omega(1,y) = 0, \quad 0 \le y \le 1. \tag{5.3}$$

The experiments were conducted on a LINUX cluster located at the Parallel And Distributed Research Lab, Dept. of Computer Science, USM. The problem was solved for various values of Reynolds number Re ≥ 1. The grid sizes used were n = 241, 481, 721 and 961 with Reynolds number Re = 1 and 1000. Throughout the experiment, a tolerance of $\delta = \varepsilon = 10^{-7}$ were used as the termination criteria for both the outer and inner iterations.

TABLES 1 and 2 list the outer and inner iteration counts required for the scheme with different mesh sizes and Re =1 and 1000 respectively. TABLES 3 and 4 show the final computed values of ψ and ω using EDG and centred difference schemes respectively for Re = 1 when the grid size is h = 1/9. TABLES 5 and 6 show the computed results when Re = 1000. Since the exact solution for this problem is not available, the numerical solutions obtained by the conventional *centred difference* scheme as the benchmark solutions are collected for different mesh sizes for comparison purposes. The approximated values obtained for both methods in these experiments are very close to one another which suggests that they are good approximations to the exact ones.

The CPU time in seconds with different number of processors are shown in TABLES 7 and 8 for Re = 1 and 1000 respectively. Speedup graph for different mesh sizes is shown in Figure 6. The efficiency (Effic.) values are also included in the tables. Figure 6 illustrates the execution times for the two parallel methods for different sizes of n.

TABLE 1: Number of outer and inner iterations with different mesh sizes and Re = 1

	Outer iter	Inner iter ψ	Inner iter ω
241	1	1	13064
	2	6476	1
	3	1	1
481	1	1	39886
	2	13570	1
	3	1	1
721	1	1	73628
	2	14080	1
	3	1	1
961	1	1	110637
	2	669	1
	3	1	1

TABLE 2: Number of outer and inner iterations with different mesh sizes and Re = 1000

	Outer iter	Inner iter ψ	Inner iter ω
241	1	1	13064
	2	6476	84
	3	37	1
	4	1	1
481	1	1	39886
	2	13570	1716
	3	79	3
	4	1	1
721	1	1	73628
	2	14080	848
	3	44	10
	4	1	1
961	1	1	110637
	2	669	2
	4	1	1

TABLE 3: Numerical solution of the Navier-Stokes equations for **Re=1** for EDG when n = 9 (in the form $a(b)=a\times10^{b}$)

Stream solutions, ψ							
0.576450(-3)	0.960580(-3)	0.136270(-2)	0.142550(-2)	0.152670(-2)	0.126710(-2)	0.103820(-2)	0.519120(-3)
0.960580(-3)	0.190320(-2)	0.234720(-2)	0.281270(-2)	0.264130(-2)	0.250360(-2)	0.177090(-2)	0.103820(-2)
0.136270(-2)	0.234710(-2)	0.331010(-2)	0.350880(-2)	0.372210(-2)	0.311290(-2)	0.250360(-2)	0.126710(-2)
0.142550(-2)	0.281270(-2)	0.350880(-2)	0.419040(-2)	0.395630(-2)	0.372210(-2)	0.264130(-2)	0.152670(-2)
0.152670(-2)	0.264130(-2)	0.372210(-2)	0.395620(-2)	0.419040(-2)	0.350880(-2)	0.281270(-2)	0.142550(-2)
0.126710(-2)	0.250360(-2)	0.311280(-2)	0.372210(-2)	0.350880(-2)	0.331000(-2)	0.234710(-2)	0.136270(-2)
0.103820(-2)	0.177090(-2)	0.250360(-2)	0.264130(-2)	0.281270(-2)	0.234710(-2)	0.190320(-2)	0.960580(-3)
0.519120(-3)	0.103820(-2)	0.126710(-2)	0.152670(-2)	0.142550(-2)	0.136270(-2)	0.960580(-3)	0.576460(-3)
Voticity solutions, ω							
0.163070(-1)	0.247390(-1)	0.297650(-1)	0.319900(-1)	0.320200(-1)	0.296690(-1)	0.247500(-1)	0.154620(-1)
0.247370(-1)	0.405360(-1)	0.495230(-1)	0.538290(-1)	0.537440(-1)	0.495610(-1)	0.402430(-1)	0.247520(-1)
0.297630(-1)	0.495220(-1)	0.616170(-1)	0.671490(-1)	0.672220(-1)	0.614780(-1)	0.495620(-1)	0.296700(-1)
0.319890(-1)	0.538280(-1)	0.671490(-1)	0.735840(-1)	0.734890(-1)	0.672220(-1)	0.537450(-1)	0.320210(-1)
0.320210(-1)	0.537450(-1)	0.672220(-1)	0.734890(-1)	0.735840(-1)	0.671490(-1)	0.538280(-1)	0.319890(-1)
0.296700(-1)	0.495620(-1)	0.614780(-1)	0.672220(-1)	0.671490(-1)	0.616170(-1)	0.495220(-1)	0.297630(-1)
0.247520(-1)	0.402430(-1)	0.495610(-1)	0.537440(-1)	0.538290(-1)	0.495230(-1)	0.405360(-1)	0.247370(-1)
0.154620(-1)	0.247500(-1)	0.296690(-1)	0.320200(-1)	0.319900(-1)	0.297650(-1)	0.247390(-1)	0.163070(-1)

TABLE 4: Numerical solution of the Navier-Stokes equations for **Re=1** for centred difference when n = 9 (in the form $a(b)=a\times10^{b}$)

Stream solutions, ψ							
0.570001(-3)	0.101984(-2)	0.131436(-2)	0.145780(-2)	0.144807(-2)	0.128509(-2)	0.974543(-3)	0.533132(-3)
0.102225(-2)	0.184557(-2)	0.239919(-2)	0.267311(-2)	0.265980(-2)	0.239952(-2)	0.178527(-2)	0.972672(-3)
0.131295(-2)	0.239251(-2)	0.313883(-2)	0.351560(-2)	0.350650(-2)	0.311177(-2)	0.235156(-2)	0.127814(-2)
0.143595(-2)	0.263855(-2)	0.349018(-2)	0.392879(-2)	0.392904(-2)	0.349033(-2)	0.263737(-2)	0.143225(-2)
0.140045(-2)	0.259071(-2)	0.344969(-2)	0.389978(-2)	0.391007(-2)	0.347877(-2)	0.263089(-2)	0.142937(-2)
0.120030(-2)	0.226793(-2)	0.303438(-2)	0.344128(-2)	0.345811(-2)	0.308229(-2)	0.233512(-2)	0.127085(-2)
0.911216(-3)	0.169686(-2)	0.227660(-2)	0.258693(-2)	0.260411(-2)	0.232573(-2)	0.176648(-2)	0.964354(-3)
0.493326(-3)	0.917494(-3)	0.123170(-2)	0.140046(-2)	0.141130(-2)	0.126290(-2)	0.962224(-3)	0.527685(-3)
Voticity solutions, ω							
0.192670(-1)	0.282974(-1)	0.308236(-1)	0.320444(-1)	0.315520(-1)	0.290119(-1)	0.238673(-1)	0.150095(-1)
0.291931(-1)	0.444230(-1)	0.506070(-1)	0.534562(-1)	0.528060(-1)	0.482827(-1)	0.391020(-1)	0.238249(-1)
0.324855(-1)	0.513561(-1)	0.613793(-1)	0.660218(-1)	0.655876(-1)	0.598654(-1)	0.488875(-1)	0.288424(-1)
0.317358(-1)	0.522289(-1)	0.651869(-1)	0.713185(-1)	0.713115(-1)	0.651580(-1)	0.521946(-1)	0.311114(-1)
0.287406(-1)	0.491008(-1)	0.634753(-1)	0.704083(-1)	0.708364(-1)	0.649150(-1)	0.520758(-1)	0.310631(-1)
0.244274(-1)	0.428845(-1)	0.568599(-1)	0.636563(-1)	0.643655(-1)	0.592441(-1)	0.477849(-1)	0.287195(-1)
0.189841(-1)	0.335488(-1)	0.450778(-1)	0.506461(-1)	0.513798(-1)	0.475651(-1)	0.387546(-1)	0.236842(-1)
0.117117(-1)	0.200948(-1)	0.269108(-1)	0.301248(-1)	0.305967(-1)	0.285365(-1)	0.236385(-1)	0.149171(-1)

TABLE 5: Numerical solution of the Navier-Stokes equations for **Re=1000** for EDG when n = 9 (in the form $a(b)=a\times10^{b}$)

Stream solutions, ψ							
0.575410(-3)	0.960780(-3)	0.136700(-2)	0.142690(-2)	0.152510(-2)	0.126430(-2)	0.103280(-2)	0.518240(-3)
0.958270(-3)	0.190070(-2)	0.234830(-2)	0.281570(-2)	0.264060(-2)	0.249920(-2)	0.176960(-2)	0.104010(-2)
0.135690(-2)	0.234480(-2)	0.330980(-2)	0.350980(-2)	0.372240(-2)	0.311280(-2)	0.250610(-2)	0.126900(-2)
0.142450(-2)	0.281160(-2)	0.350910(-2)	0.419120(-2)	0.395710(-2)	0.372360(-2)	0.264270(-2)	0.152960(-2)
0.152960(-2)	0.264270(-2)	0.372360(-2)	0.395710(-2)	0.419120(-2)	0.350910(-2)	0.281160(-2)	0.142450(-2)
0.126900(-2)	0.250610(-2)	0.311290(-2)	0.372240(-2)	0.350980(-2)	0.330980(-2)	0.234480(-2)	0.135690(-2)
0.104010(-2)	0.176960(-2)	0.249920(-2)	0.264060(-2)	0.281570(-2)	0.234830(-2)	0.190070(-2)	0.958270(-3)
0.518240(-3)	0.103280(-2)	0.126430(-2)	0.152510(-2)	0.142690(-2)	0.136700(-2)	0.960780(-3)	0.575410(-3)
Voticity solutions, ω							
0.162360(-1)	0.251210(-1)	0.304320(-1)	0.324930(-1)	0.318200(-1)	0.290060(-1)	0.239750(-1)	0.153070(-1)
0.239990(-1)	0.402520(-1)	0.497850(-1)	0.542070(-1)	0.536670(-1)	0.490160(-1)	0.399330(-1)	0.251740(-1)
0.289730(-1)	0.490000(-1)	0.615600(-1)	0.672540(-1)	0.672290(-1)	0.613930(-1)	0.498930(-1)	0.303510(-1)
0.318930(-1)	0.537150(-1)	0.671360(-1)	0.736220(-1)	0.735420(-1)	0.673600(-1)	0.540790(-1)	0.324290(-1)
0.324290(-1)	0.540790(-1)	0.673600(-1)	0.735420(-1)	0.736220(-1)	0.671360(-1)	0.537150(-1)	0.318930(-1)
0.303510(-1)	0.498930(-1)	0.613930(-1)	0.672300(-1)	0.672540(-1)	0.615600(-1)	0.490000(-1)	0.289730(-1)
0.251740(-1)	0.399330(-1)	0.490160(-1)	0.536670(-1)	0.542070(-1)	0.497850(-1)	0.402520(-1)	0.239990(-1)
0.153070(-1)	0.239750(-1)	0.290060(-1)	0.318200(-1)	0.324930(-1)	0.304320(-1)	0.251210(-1)	0.162360(-1)

TABLE 6: Numerical solution of the Navier-Stokes equations for **Re=1000** for centred difference when n = 9 (in the form $a(b)=a\times10^{b}$)

Stream solutions, ψ							
0.332278(-3)	0.972101(-3)	0.127944(-2)	0.143651(-2)	0.143654(-2)	0.127930(-2)	0.972172(-3)	0.332330(-3)
0.972107(-3)	0.178306(-2)	0.235330(-2)	0.264568(-2)	0.264572(-2)	0.235342(-2)	0.178319(-2)	0.972205(-3)
0.127945(-2)	0.235331(-2)	0.311203(-2)	0.350209(-2)	0.350215(-2)	0.311219(-2)	0.235350(-2)	0.127959(-2)
0.143652(-2)	0.264568(-2)	0.350209(-2)	0.394300(-2)	0.394307(-2)	0.350227(-2)	0.264591(-2)	0.143669(-2)
0.143652(-2)	0.264570(-2)	0.350213(-2)	0.394306(-2)	0.394314(-2)	0.350232(-2)	0.264594(-2)	0.143671(-2)
0.127946(-2)	0.235337(-2)	0.311215(-2)	0.350225(-2)	0.350232(-2)	0.311233(-2)	0.235360(-2)	0.127964(-2)
0.972129(-3)	0.178314(-2)	0.235345(-2)	0.264588(-2)	0.264593(-2)	0.235360(-2)	0.178332(-2)	0.972269(-3)
0.332299(-3)	0.972163(-3)	0.127956(-2)	0.143667(-2)	0.143670(-2)	0.127963(-2)	0.972268(-3)	0.332379(-3)
Voticity solutions, ω							
0.149871(-1)	0.237979(-1)	0.288381(-1)	0.311527(-1)	0.311522(-1)	0.288363(-1)	0.237939(-1)	0.149833(-1)
0.237990(-1)	0.390156(-1)	0.480562(-1)	0.522750(-1)	0.522743(-1)	0.480537(-1)	0.390101(-1)	0.237938(-1)
0.288404(-1)	0.480579(-1)	0.597505(-1)	0.652713(-1)	0.652707(-1)	0.597484(-1)	0.480535(-1)	0.288362(-1)
0.311534(-1)	0.522754(-1)	0.652710(-1)	0.714434(-1)	0.714431(-1)	0.652702(-1)	0.522737(-1)	0.311519(-1)
0.311502(-1)	0.522720(-1)	0.652692(-1)	0.714425(-1)	0.714427(-1)	0.652700(-1)	0.522736(-1)	0.311518(-1)
0.288320(-1)	0.480485(-1)	0.597458(-1)	0.652689(-1)	0.652695(-1)	0.597479(-1)	0.480533(-1)	0.288361(-1)
0.237886(-1)	0.390044(-1)	0.480506(-1)	0.522723(-1)	0.522730(-1)	0.480531(-1)	0.390099(-1)	0.237937(-1)
0.149795(-1)	0.237896(-1)	0.288342(-1)	0.311509(-1)	0.311514(-1)	0.288360(-1)	0.237937(-1)	0.149833(-1)

TABLE 7: CPU times in seconds with different mesh sizes and Re = 1

Processor	n_x = 241	n_x = 481	n_x = 721	n_x = 961
1	144.690782	1970.945206	7871.971455	18335.867494
2	85.773105	1025.750967	3921.998692	8789.410456
4	64.078650	594.933701	2127.728505	4645.004173
6	58.220974	431.503218	1513.939584	3247.341130
8	46.341573	351.298681	1181.044709	2546.225946
10	55.113341	299.389544	1002.565540	2130.409224
12	55.502256	266.157817	868.026409	1839.309753

TABLE 8: CPU times in seconds with different mesh sizes and Re = 1000

Processor	n_x = 241	n_x = 481	n_x = 721	n_x = 961
1	151.830907	2033.863913	7906.512131	18123.823457
2	88.689624	1064.255100	3975.431953	8902.581275
4	79.447877	620.369839	2152.611467	4664.392306
6	64.794269	445.971197	1529.027050	3269.090209
8	46.439468	361.389018	1201.849925	2547.280551
10	58.305965	308.709011	1015.890856	2127.266721
12	58.829577	277.974215	881.422557	1838.301921

We notice that with one processor, the method takes longer CPU time to converge as the mesh sizes get larger. However, as more processors are utilized, the number of inner and outer iterations are the same as when one processor is used, but the CPU times are substantially reduced. The overall speedup values for the parallel implementation get closer to the 'ideal' value as the mesh size gets larger. For example, the speedup with 12 processors with n=721 is 9.0688 compared to 9.9689 when n=961 for Re = 1. It is also observed that the efficiency of the algorithm increases as the size of the problem becomes larger. This indicates that the amount of computations carried out over the total overheads in this method becomes greater as the mesh size increases. Thus , the problem is able to benefit the most from the parallelisation as the amount of work increases.

5. CONCLUSION

We have derived a new parallel group explicit finite difference scheme for the two dimensional steady state Navier-Stokes equation. The group of grid points can be decoupled with only two colours when a parallel block Gauss- Seidel type iterative strategy is employed. Numerical experiments were conducted to test the parallel strategy on the coupled p.d.e.'s. Our studies show that the parallel group scheme has the advantage of delivering fast solutions on a LINUX cluster.

6. ACKNOWLEDGEMENT

The authors acknowledge the Fundamental Research Grant Scheme (304/PMATHS/670020) that has resulted in this article.

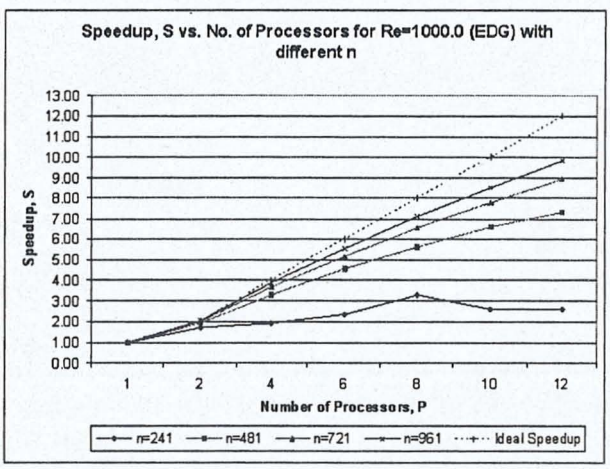

_Fig. 6: Speedups for the parallel method for different mesh sizes

7. REFERENCES

[1] A.R. Abdullah and N.H.M. Ali, The Comparative Study of Parallel Strategies for the Solution of Elliptic PDE's, *Parallel Algorithms and Applications,* 10, 1996, 93-103.

[2] N.H.M. Ali and A.R. Abdullah, Parallel Four Point Explicit Decoupled Group (EDG) Method For Elliptic PDE's. *Proceedings of the Seventh IASTED/ISMM International Conference on Parallel and Distributed Computing and Systems, Washington D.C.*, 1995, 302-304.

[3] N.H.M.Ali and A.R.Abdullah, "Four point EDG: A Fast Solver for the Navier-Stokes Equation", in *Proceedings of the International Conference on Modelling Simulation and Optimization, Gold Coast, Australia*, 1996, CD Rom-File 242-65.pdf, ISBN: 0-88986-197-8.

[4] N.H.M.Ali & A.R. Abdullah, New Rotated Iterative Algorithms For The Solution of A Coupled System of Elliptic Equations, *International Journal of Computer Mathematics*, Vol. 74, 1999, pp. 223-251.

[5] G. Dahlquist and A. Bjorck, *Numerical Methods,* Prentice-Hall, Englewood Cliffs, NJ., 1974.

[6] D. Greenspan, *Discrete Numerical Methods in Physics and Engineering,* Academic Press Inc., New York, 1974.

[7] M. S. Sahimi & D. J. Evans, The numerical solution of a coupled system of elliptic equations using the AGE fractional scheme. *Int. Journal Computer Maths.* 50, 1994, 65-87.