Malaysian Journal of Computer Science, Vol. 10 No. 2, December 1997, pp. 51-59

A NEW FAST NAVIER-STOKES SOLVER AND ITS PARALLEL IMPLEMENTATION

Norhashidah Hj. Mohd. Ali
School of Mathematical Sciences
Universiti Sains Malaysia
11800 Penang
Malaysia
email: shidah@cs.usm.my

Abdul Rahman Abdullah
Faculty of Information Science and Technology
Universiti Kebangsaan Malaysia
43600 Bangi, Selangor Darul Ehsan
Malaysia
email: ara@pkrisc.cc.ukm.my

ABSTRACT

The Four-point Explicit Decoupled Group(EDG) iterative method was briefly introduced in solving a well known coupled system of elliptic partial differential equations in CFD (Computational Fluid Dynamics), in particular, the steady Navier-Stokes equations. This new method is compared with the Alternating Group Explicit (AGE) scheme. The numerical experiments carried out confirm the superiority of the former method over the latter in terms of execution time. The parallel implementation of the new method intended for a shared memory parallel computer is investigated and discussed.

Keywords: ***Explicit decoupled group (EDG) method, numerical methods, Navier-Stokes equations, parallel algorithm, Alternating Group Explicit (AGE) method, Computational Fluid Dynamics (CFD)***

1.0 INTRODUCTION

Consider the following coupled system of partial differential equations:

$$\nabla^2 y = -w \qquad (1.1)$$

$$\nabla^2 w + \mathrm{Re}(y_x w_y - y_y w_x) = -c \qquad (1.2)$$

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

The aim of this paper is to investigate the versatility of the four-point EDG method, sequentially and in parallel, 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 2.0. In Section 3.0, 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 4.0 describes the numerical algorithm for solving the coupled system (1.1)-(1.2) by incorporating the four-point EDG in its iteration scheme, followed by the numerical experiments and comparison with the AGE fractional scheme in Section 5.0. In Section 6.0, the parallel algorithm for the EDG scheme for solving the same problem is developed and investigated, and the concluding remarks are presented in Section 7.0.

2.0 FINITE DIFFERENCE APPROXIMATIONS

Let us assume that a rectangular grid in the (x,y)-plane with grid spacing h = L/n in both directions and $x_i = ih,\ y_j = jh,\ i,j = 0,1,\ldots,n$ is used. Observe that if w is known, then (1.1) is a linear *elliptic* equation in y, and if y is known, then (1.2) is a linear *elliptic* equation in w. Suppose $y^{(0)}$ and $w^{(0)}$ are the initial guesses, we can use the $w^{(0)}$ in (1.1) to produce $y^{(1)}$. Again we can use this $y^{(1)}$ in (1.2) to produce $w^{(1)}$. Then we can use this $w^{(1)}$ in (1.1) to produce $y^{(2)}$, and so on. This indicates that at the grid point (x_i, y_j) the following alternating sequences of outer iterates can be generated:

$$\begin{tikzcd}
y_{ij}^{(0)} & y_{ij}^{(1)} \arrow[d] & y_{ij}^{(2)} \arrow[d] & y_{ij}^{(3)} \arrow[d] & y_{ij}^{(4)} \arrow[d] & \ldots & y_{ij}^{(k)} \arrow[d] & \ldots \\
w_{ij}^{(0)} \arrow[ur] & w_{ij}^{(1)} \arrow[ur] & w_{ij}^{(2)} \arrow[ur] & w_{ij}^{(3)} \arrow[ur] & w_{ij}^{(4)} & \ldots & w_{ij}^{(k)} & \ldots
\end{tikzcd}$$

Fig. 1: Generation of outer iterates

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:

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

$$\begin{aligned}&-\left[1 - s\left(y_{i,j-1}^{(k+1)} - y_{i,j+1}^{(k+1)}\right)\right]w_{i-1,j}^{(k+1)} - \left[1 + s\left(y_{i-1,j}^{(k+1)} - y_{i+1,j}^{(k+1)}\right)\right]w_{i,j-1}^{(k+1)} + 4w_{ij}^{(k+1)}\\ &-\left[1 - s\left(y_{i-1,j}^{(k+1)} - y_{i+1,j}^{(k+1)}\right)\right]w_{i,j+1}^{(k+1)} - \left[1 + s\left(y_{i,j-1}^{(k+1)} - y_{i,j+1}^{(k+1)}\right)\right]w_{i+1,j}^{(k+1)} = h^2 c_{ij}^{(k)},\end{aligned} \qquad (2.2)$$

here $s =$ 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 [6] which can be obtained by rotating the i-plane axis and the j-plane axis clockwise by 45^o. With this displacement, equations (2.1) and (2.2) become (2.3) and (2.4) respectively:

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

$$\begin{aligned}&-[1 - s(y_{i-1,j-1}^{(k+1)} - y_{i+1,j+1}^{(k+1)})]w_{i-1,j+1}^{(k+1)} - [1 + s(y_{i-1,j+1}^{(k+1)} - y_{i+1,j-1}^{(k+1)})]w_{i-1,j-1}^{(k+1)} + 4w_{ij}^{(k+1)}\\ &-[1 - s(y_{i-1,j+1}^{(k+1)} - y_{i+1,j-1}^{(k+1)})]w_{i+1,j+1}^{(k+1)} - [1 + s(y_{i-1,j-1}^{(k+1)} - y_{i+1,j+1}^{(k+1)})]w_{i+1,j-1}^{(k+1)} = 2h^2 c_{ij}^{(k)}.\end{aligned} \qquad (2.4)$$

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.

3.0 THE FOUR POINT EDG FORMULATION

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

$$\begin{bmatrix} 4 & -[1 - s(y_{i-1,j+1} - y_{i+1,j-1})] & 0 & 0 \\ -[1 + s(y_{i,j+2} - y_{i+2,j})] & 4 & 0 & 0 \\ 0 & 0 & 4 & -[1 - s(y_{i,j-1} - y_{i+2,j+1})] \\ 0 & 0 & -[1 + s(y_{i-1,j} - y_{i+1,j+2})] & 4 \end{bmatrix} \begin{bmatrix} w_{ij} \\ w_{i+1,j+1} \\ w_{i+1,j} \\ w_{i,j+1} \end{bmatrix}$$

$$= \begin{bmatrix} [1 - s(y_{i-1,j-1} - y_{i+1,j+1})]w_{i-1,j+1} + [1 + s(y_{i-1,j+1} - y_{i+1,j-1})]w_{i-1,j-1} + [1 + s(y_{i-1,j-1} - y_{i+1,j+1})]w_{i+1,j-1} + 2h^2 c_{ij} \\ [1 - s(y_{ij} - y_{i+2,j+2})]w_{i,j+2} + [1 - s(y_{i,j+2} - y_{i+2,j})]w_{i+2,j+2} + [1 + s(y_{ij} - y_{i+2,j+2})]w_{i+2,j} + 2h^2 c_{i+1,j+1} \\ [1 - s(y_{i,j+1} - y_{i+2,j-1})]w_{i+2,j+1} + [1 + s(y_{i,j+1} - y_{i+2,j-1})]w_{i,j-1} + [1 + s(y_{i,j-1} - y_{i+2,j+1})]w_{i+2,j-1} + 2h^2 c_{i+1,j} \\ [1 - s(y_{i-1,j+2} - y_{i+1,j})]w_{i+1,j+2} + [1 + s(y_{i-1,j+2} - y_{i+1,j})]w_{i-1,j} + [1 - s(y_{i-1,j} - y_{i+1,j+2})]w_{i-1,j+2} + 2h^2 c_{i,j+1} \end{bmatrix}$$

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

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

$$\begin{bmatrix} \tilde{w}_{ij} \\ \tilde{w}_{i+1,j+1} \end{bmatrix} = \frac{1}{16 - [(1 - s(y_{i-1,j+1} - y_{i+1,j-1}))(1 + s(y_{i,j+2} - y_{i+2,j}))]} \text{X} \begin{bmatrix} 4 & 1 - s(y_{i-1,j+1} - y_{i+1,j-1}) \\ 1 + s(y_{i,j+2} - y_{i+2,j}) & 4 \end{bmatrix} \begin{bmatrix} rhs_{ij} \\ rhs_{i+1,j+1} \end{bmatrix} \qquad (3.1)$$

and

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

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

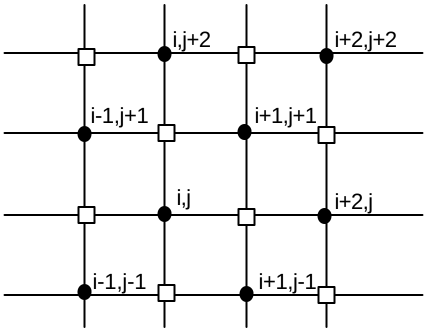

Fig. 2: Computational molecule of Eq. (3.1)

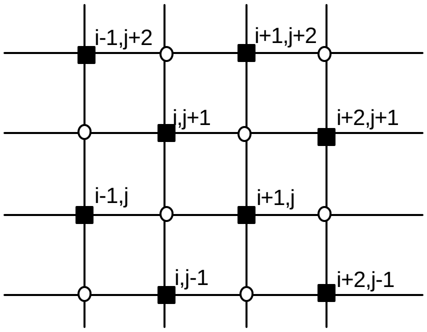

Fig. 3: Computational molecule of Eq. (3.2)

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 arose 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 types 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(SOR) scheme corresponds to iterating the solutions at approximately *half* of the points in the solution domain using either (3.1) or (3.2) by implementing the relaxation procedure $w^{(k+1)} = w^{(k)} + g(\tilde{w}^{(k+1)} - w^{(k)})$ until convergence is achieved, i.e., when $\left| w_{ij}^{(k+1)} - w_{ij}^{(k)} \right| \leq e$; here g is the optimum acceleration parameter and e is the convergence criterion used. If convergence is achieved, then the solutions are evaluated at the rest of the nodal points *(points of opposite type)* using the *centred difference* formula (2.2). Otherwise, the iteration cycle is repeated.

In the case of n even, the EDGR scheme is adopted, i.e., we assume the uncoupled points are on the rightmost and topmost grid lines. Suppose Eq. (3.1) is chosen to be used in the iterative evaluation of points, these uncoupled points must be calculated after the iterations on the points ● have converged. The uncoupled points of the same type are calculated using the *rotated* formula of Eq. (2.4). Only after these points have been calculated, the remaining points of the uncoupled group (of the opposite type) are evaluated using the *centred difference* formula (2.2).

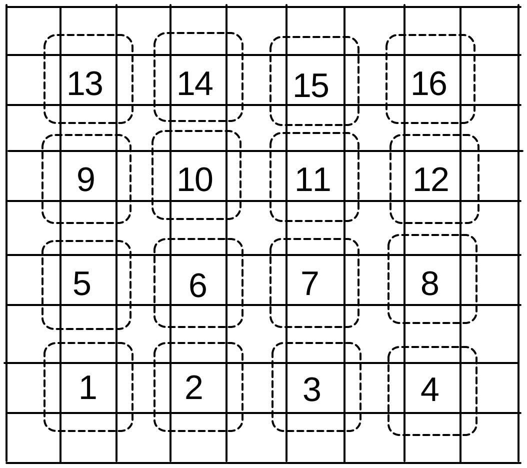

Fig. 4: Natural group ordering

In theory, if we apply Eq. (3.1) or (3.2) to each of the groups in natural ordering as in Fig. 4, then we shall obtain a system of equations

$$A\mathbf{u} = \mathbf{b} \tag{3.3}$$

whose coefficient matrix A is defined as

$$A = \begin{bmatrix} D_1 & V_1 & 0 & 0 \\ L_3 & D_3 & V_3 & 0 \\ 0 & L_5 & D_5 & V_5 \\ 0 & 0 & L_7 & D_7 \end{bmatrix} \tag{3.4}$$

with

$$D_j = \begin{bmatrix} R_{1j} & Q_{1j} & 0 & 0 \\ U_{3j} & R_{3j} & Q_{3j} & 0 \\ 0 & U_{5j} & R_{5j} & Q_{5j} \\ 0 & 0 & U_{7j} & R_{7j} \end{bmatrix}, \; L_j = \begin{bmatrix} S_{1j} & 0 & 0 & 0 \\ E_{3j} & S_{3j} & 0 & 0 \\ 0 & E_{5j} & S_{5j} & 0 \\ 0 & 0 & E_{7j} & S_{7j} \end{bmatrix},$$

$$V_j = \begin{bmatrix} F_{1j} & G_{1j} & 0 & 0 \\ 0 & F_{3j} & G_{3j} & 0 \\ 0 & 0 & F_{5j} & G_{5j} \\ 0 & 0 & 0 & F_{7j} \end{bmatrix}$$

where

$$R_{ij} = \begin{bmatrix} 4 & -[1-s(y_{i-1,j+1} - y_{i+1,j-1})] \\ -[1+s(y_{i,j+2} - y_{i+2,j})] & 4 \end{bmatrix},$$

$$S_{ij} = \begin{bmatrix} 0 & -[1+s(y_{i-1,j-1} - y_{i+1,j+1})] \\ 0 & 0 \end{bmatrix},$$

$$Q_{ij} = \begin{bmatrix} 0 & 0 \\ -[1+s(y_{ij} - y_{i+2,j+2})] & 0 \end{bmatrix},$$

$$E_{ij} = \begin{bmatrix} 0 & -[1+s(y_{i-1,j+1} - y_{i+1,j-1})] \\ 0 & 0 \end{bmatrix},$$

$$F_{ij} = \begin{bmatrix} 0 & 0 \\ -[1-s(y_{ij} - y_{i+2,j+2})] & 0 \end{bmatrix},$$

$$U_{ij} = \begin{bmatrix} 0 & -[1-s(y_{i-1,j-1} - y_{i+1,j+1})] \\ 0 & 0 \end{bmatrix}$$

$$\text{and } G_{ij} = \begin{bmatrix} 0 & 0 \\ -[1-s(y_{i,j+2} - y_{i+2,j})] & 0 \end{bmatrix}, \quad i,j = 1(2)(n\text{-}2). \tag{3.5}$$

The coefficient matrix A can be written as the sum of its diagonal block element D, its strictly lower triangular block element L, and its strictly upper triangular block element U, i.e.,

$$A = D + L + U. \tag{3.6}$$

Thus, the solution of (3.3) can be iteratively computed via Gauss-Seidel scheme as

$$(D + L)\mathbf{u}^{(m+1)} = \mathbf{b} - U\mathbf{u}^{(m)}. \tag{3.7}$$

This scheme can be further improved by introducing the over-relaxation parameter g .

4.0 NUMERICAL ALGORITHM

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

Step 1 Choose h and construct the number of nodal points as usual for an elliptic problem. Set $y_{ij}^{(0)} = w_{ij}^{(0)} = 0 = \text{outer\_}y_{ij}^{(0)} = \text{outer\_}w_{ij}^{(0)}$ as initial approximations for the outer iteration.

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

Generate $y_{ij}^{(k+1)}$ of (2.3) using the four-point EDG inner iterative procedure described in Section 3.0 for a prescribed tolerance e. (Use the same Eq. (2.4) but replace w with y, c_{ij} with w_{ij}, and $s = 0$).

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

Store the converged values $y_{ij}^{(k+1)}$ in outer_ $y_{ij}^{(m)}$, and $w_{ij}^{(k+1)}$ in outer_ $w_{ij}^{(m)}$.

Step 3 Check the convergence of the outer iteration process over the whole mesh points for a prescribed convergence criterion d, i.e., check whether the following condition is achieved,

$$\max\left\{\left|\text{outer\_}y_{ij}^{(r+1)} - \text{outer\_}y_{ij}^{(r)}\right|, \left|\text{outer\_}w_{ij}^{(r+1)} - \text{outer\_}w_{ij}^{(r)}\right|\right\} \le d .$$

If convergence is achieved, then the numerical solution of the given problem is given by the generated outer_ $y_{ij}^{(m+1)}$ and outer_ $w_{ij}^{(m+1)}$. Otherwise, go back to ***Step 2***.

5.0 NUMERICAL EXPERIMENTATION

Numerical experiments have been carried out on the Sequent Balance 8000 multiprocessor at Loughborough University of Technology, United Kingdom, using the algorithm described previously to solve the following Navier-Stokes equations [9],

$$\nabla^2 y = -w \qquad (5.1)$$

$$\nabla^2 w + \text{Re}\left(y_x w_y - y_y w_x\right) = -1 \qquad (5.2)$$

with the boundary conditions

$$y(x,0) = y(x,1) = w(x,0) = w(x,1) = 0, \quad 0 \le x \le 1,$$
$$y(0,y) = y(1,y) = w(0,y) = w(1,y) = 0, \quad 0 \le y \le 1. \qquad (5.3)$$

The grid spacing used was h = 0.1 and the problem was solved for various values of Reynolds number Re ≥ 1. For each case, the experimental optimum relaxation parameter g was chosen to within ±0.01 which gives the most rapid convergence. Throughout the experiment, a tolerance of $d = e = 10^{-11}$ was used as the termination criteria for both the outer and inner iterations.

Table 1 lists the iteration counts and timings for the EDG method for selected Re ranging from 1 to 6954. The same problem was also solved using the AGE fractional scheme [9]. In ***Step 2*** of the numerical algorithm presented in Section 4.0, we replaced the EDG inner iterative process with AGE inner iterative scheme. The iteration counts and timings obtained from this method are also displayed in Table 1 for selected values of Re. Here, r represents the optimum acceleration parameter for the AGE scheme which was chosen experimentally the same way as in the EDG case. The final computed values of y and w using the EDG and AGE methods for selected values of x and y for Re = 1 and 1000 are shown in Tables 2-3 (a and b). For comparison purposes, the numerical solutions obtained by the conventional *centred difference* scheme is also included.

Table 1: Iteration counts and timings of EDG and AGE for various Reynolds number

Iteration numbers for the Navier-Stokes problem using the four point EDG scheme						Iteration numbers for the Navier-Stokes problem using the AGE fractional scheme					
Re	Time (secs)	Value of g	Number of outer iteration m	Number of inner iteration for y	Number of inner iteration for w	Re	Time (secs)	Value of r	Number of outer iteration m	Number of inner iteration for y	Number of inner iteration for w
1	1.50	1.37	1	1	23	1	12.99	1.26	1	1	58
			2	21	14				2	50	20
			3	9	4				3	7	3
			4	1	1				4	1	1
10	1.78	1.37	1	1	23	10	14.88	1.26	1	1	58
			2	21	16				2	50	28
			3	10	8				3	11	9
			4	4	1				4	2	1
			5	1	1				5	1	1
100	2.27	1.37	1	1	23	100	19.14	1.26	1	1	58
			2	21	19				2	50	34
			3	13	13				3	19	20
			4	8	7				4	13	6
			5	2	2				5	2	2
			6	1	1				6	1	1
1000	4.31	1.29	1	1	40	1000	30.30	1.26	1	1	58
			2	35	25				2	50	41
			3	14	21				3	27	41
			4	16	17				4	25	28
			5	9	14				5	16	15
			6	6	10				6	6	9
			7	3	6				7	2	4
			8	1	2				8	1	2
			9	1	1				9	1	1
5000	13.63	0.82	1	1	121	5000	62.01	1.70	1	1	88
			2	104	74				2	76	79
			3	53	66				3	45	70
			4	53	52				4	41	46
			5	18	47				5	33	50
			6	30	38				6	24	41
			7	21	29				7	14	29
			8	11	17				8	6	16
			9	3	14				9	1	7
			10	1	10				10	1	3
			11	1	7				11	1	1
			12	1	4	7334	94.84	1.77	1	1	92
			13	1	1				2	79	150
			14	1	1				3	48	129
6954	23.62	0.72	1	1	149				4	43	106
			2	128	144				5	35	90
			3	65	150				6	26	72
			4	65	136				7	17	52
			5	35	119				8	8	37
			6	37	101				9	2	16
			7	27	83				10	1	4
			8	15	63				11	1	2
			9	4	40				12	1	1
			10	1	11	>7334		Diverge			
			11	1	6						
			12	1	6						
			13	1	3						
			14	1	2						
>6954		Diverge									

Table 2a: Numerical solutions obtained for y when x = 0.2 and 0.7 (Re = 1)

y	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
x = 0.2									
Centred Difference	.0008093	.0015056	.0020305	.0023544	.0024636	.0023544	.0020305	.0015056	.0008093
AGE	.0008093	.0015056	.0020305	.0023544	.0024636	.0023544	.0020305	.0015056	.0008093
EDG	.0008539	.0015889	.0021345	.0024753	.0025862	.0024754	.0021345	.0015889	.0008539
x = 0.7									
Centred Difference	.0010885	.0020305	.0027439	.0031854	.0033345	.0031854	.0027439	.0020305	.0010885
AGE	.0010885	.0020305	.0027439	.0031854	.0033345	.0031854	.0027439	.0020305	.0010885
EDG	.0011466	.0021345	.0028863	.0033444	.0035043	.0033444	.0028863	.0021345	.0011466

Table 2b: Numerical solutions obtained for w when x = 0.2 and 0.7 (Re = 1)

y	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
x = 0.2									
Centred Difference	.0206265	.0342946	.0429625	.0477719	.0493143	.0477726	.0429633	.0342946	.0206258
AGE	.0206265	.0342946	.0429625	.0477719	.0493143	.0477726	.0429633	.0342946	.0206258
EDG	.0213641	.0354767	.0440904	.0489574	.0504545	.0489583	.0440914	.0354767	.0213632
x = 0.7									
Centred Difference	.0253966	.0429625	.0543877	.0608126	.0628844	.0608123	.0543877	.0429633	.0253977
AGE	.0253966	.0429625	.0543877	.0608126	.0628844	.0608123	.0543877	.0429633	.0253977
EDG	.0261079	.0440904	.0558204	.0622729	.0644235	.0622726	.0558204	.0440914	.0261094

Table 3a: Numerical solutions obtained for y when x = 0.2 and 0.7 (Re = 1000)

y	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
x = 0.2									
Centred Difference	.0008098	.0015043	.0020279	.0023530	.0024647	.0023566	.0020316	.0015043	.0008070
AGE	.0008098	.0015043	.0020279	.0023530	.0024647	.0023566	.0020316	.0015043	.0008070
EDG	.0008546	.0015865	.0021302	.0024726	.0025872	.0024787	.0021359	.0015865	.0008503
x = 0.7									
Centred Difference	.0010857	.0020279	.0027434	.0031863	.0033351	.0031849	.0027434	.0020316	.0010907
AGE	.0010857	.0020279	.0027434	.0031863	.0033351	.0031849	.0027434	.0020316	.0010907
EDG	.0011414	.0021302	.0028853	.0033458	.0035053	.0033436	.0028853	.0021359	.0011499

Table 3b : Numerical solutions obtained for w when x = 0.2 and 0.7 (Re = 1000)

y	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
x = 0.2									
Centred Difference	.0208293	.0340898	.0425407	.0475625	.0494725	.0481154	.0431501	.0340898	.0201854
AGE	.0208293	.0340898	.0425407	.0475625	.0494725	.0481154	.0431501	.0340898	.0201854
EDG	.0216508	.0351961	.0434897	.0486152	.0506435	.0494292	.0443268	.0351961	.0207396
x = 0.7									
Centred Difference	.0248889	.0425407	.0542726	.0608966	.0629378	.0607102	.0542726	.0431501	.0258641
AGE	.0248889	.0425407	.0542726	.0608966	.0629378	.0607102	.0542726	.0431501	.0258641
EDG	.0253344	.0434897	.0556858	.0624054	.0645339	.0621464	.0556858	.0443268	.0267076

6.0 PARALLEL IMPLEMENTATION

Recall that in the numerical algorithm presented, for each outer iteration there will be two EDG inner iterative processes involved; one for y and another one for w. 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 y for the first inner iteration, and a linear elliptic equation in w for the second inner iteration, we will adopt the *best* strategy in parallelising EDG for the two dimensional elliptic problem [2] in each inner iterative process. This means that for each inner iteration there are four stages of computations such that an approximately equal number of blocks of points are assigned to each processor in each stage.

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

```
     ( Declaration section and initialization of
     data/constants to be used in the program )
     call initialize_index  ( to initialize the points to
     the blocks )
     call calculate_num_blocks   ( to calculate the
     number of blocks in each stage )
     create multiple processes
10   do k = 0 to 1      ( when k = 0, do inner iteration
         for $y$, when k = 1, do inner iteration for $w$ )
        if k = 0 then $S$ = 0
          else $S$ = Re/4
        end if
        m_fork(EDG)
        m_fork(Max_Error)
        if k = 1 then
           check outer iteration convergence ( if
               converge, go to 20, else go to 10 )
        end if
     end do
20   kill child processes and summing up results
     ( Print results )
     end
```

Here, subroutine *EDG* implements the EDG inner iterative process in parallel. In this procedure, caution should be taken in keeping track of the variables evaluated since only one equation is being used to evaluate y and w, one after the other. Also, the initialization of the values to be used in each inner iteration are all done in parallel here such that synchronizing points have to be placed in appropriate places in the program to ensure the right answers are obtained. Subroutine *Max_Error* uses the same number of processors available as in *EDG* to find the maximum error in checking the convergence of the outer iteration.

To test the parallel algorithm just described, we use the same problem (5.1)-(5.2) as before. The experiment runs on the same Balance multiprocessor at LUT, United Kingdom. The maximum number of iterations was set to be 500, and p processors (p = 1,2,...,9) were used for n = 61, 91 and 121 with Reynolds number Re = 1 and 1000. The elapsed times against the number of processors were plotted for Re = 1 and 1000 and are shown in Figs. 5 and 6 respectively. Speedup values are shown in Table 4.

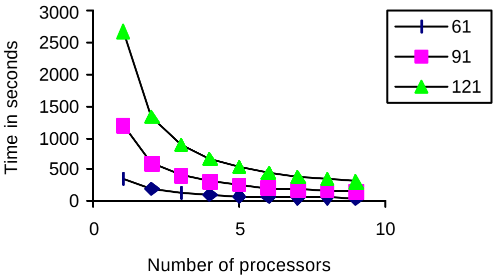

Fig. 5: Time in seconds when Re = 1 with sizes of n as shown

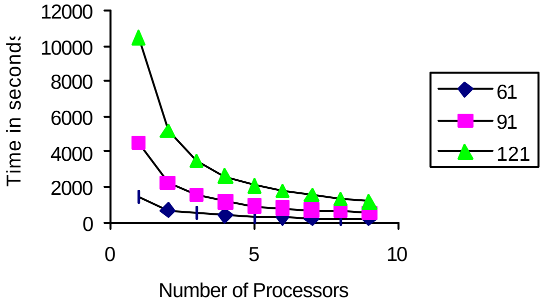

Fig. 6: Time in seconds when Re = 1000 with sizes of n as shown.

7.0 CONCLUSIONS

For the sequential algorithm, the symmetry of the EDG computed values indicate that they are good approximations to the exact solution. But as Re gets larger, the computed values seem to lose their symmetrical properties slightly, indicating a small decrease in accuracy. Furthermore, if we set 150 as the maximum number of iterations, the method gets to be divergent for Re > 6954. This is due to the terms $\frac{\mathrm{Re}}{4}\left(y^{(k+1)}_{i-1,\,j-1} - y^{(k+1)}_{i+1,\,j+1}\right)$ and $\frac{\mathrm{Re}}{4}\left(y^{(k+1)}_{i-1,\,j+1} - y^{(k+1)}_{i+1,\,j-1}\right)$ in the vorticity Eq. (2.4) which have become so large that the coefficient matrix A has lost its diagonal dominance. This also explains why the method is diverging in its attempt to generate the vorticity solutions $w^{(k+1)}_{ij}$ in the second inner iteration (see Table 1). The results from AGE scheme portray similar behaviour as the four-point EDG. Between these two methods, it can be seen that AGE is better in accuracy compared to EDG, with its numerical solutions being identical with the ones obtained from the well known *centred difference* scheme in all of the cases attempted.

However, one big drawback is that AGE is far more expensive in terms of execution time. In all cases observed, it can be seen that the EDG method requires only about 11-28 % of the time required by the AGE scheme. This is due to the latter's higher computational complexity in its iteration formula, while for EDG, it's iterative formula is less complicated and only *half* of the total nodal points are involved in the iteration. The iteration count for AGE increases at a slower rate than EDG as Re gets larger so that AGE converges for Re $\leq$ 7334 compared to 6954 for EDG.

Table 4 : Speedup values for the parallel 4-point EDG

Size of n	Number of Procs. (p)	Speedup values when Re = 1	Speedup values when Re = 1000
61	1	1.000	1.000
	2	1.968	1.976
	3	2.907	2.932
	4	3.789	3.845
	5	4.804	4.864
	6	5.670	5.749
	7	6.519	6.569
	8	7.391	7.498
	9	8.414	8.652
91	1	1.000	1.000
	2	2.016	2.022
	3	2.986	3.018
	4	3.946	3.980
	5	4.891	4.954
	6	5.822	5.912
	7	6.709	6.822
	8	7.606	7.752
	9	8.524	8.722
121	1	1.000	1.000
	2	2.027	2.021
	3	3.026	3.024
	4	4.020	4.020
	5	5.009	5.015
	6	5.972	6.006
	7	6.914	6.963
	8	7.863	7.975
	9	8.804	8.921

For the parallel implementations, nearly linear speedups were obtained in all cases attempted, with the performance being slightly better when Re = 1000 compared to when Re = 1. This indicates that the problem is able to benefit most from the parallelisation as Re gets larger and as the amount of work increases. Overall, the best speedup values were achieved when n = 121 and Re = 1000. In conclusion, the four-point EDG proves to be a viable alternative as a fast solver for the steady Navier-Stokes problem sequentially and in parallel.

REFERENCES

[1] A. R. Abdullah, "The Four Point Explicit Decoupled Group (EDG) Method: A Fast Poisson Solver", *International Journal of Computer Mathematics*, Vol. 38, 1991, pp. 61-70.

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

[4] I. Arsmah, *The Study of the Iterative Solution of the Boundary Value Problems by the Finite Difference Methods*, Ph.D. Thesis, Universiti Kebangsaan Malaysia, 1993.

[5] A. J. Chorin, "Numerical Solution of the Navier-Stokes Equations", *Mathematics of Comp*utation, Vol. 22, 1969, pp. 745-762.

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

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

[8] C. E. Pearson, "A Computational Method for Viscous Flow Problems", *Journal of Fluid Mechanics*, Vol. 21, No. 4, 1965, pp. 611-622.

[9] M. S. Sahimi and D. J. Evans, "The Numerical Solution of a Coupled System of Elliptic Equations Using The AGE Fractional Scheme", *International Journal of Computer Mathematics*, Vol. 50, 1994, pp. 65-87.

BIOGRAPHY

Norhashidah Hj. Mohd. Ali obtained her Master of Science in Applied Mathematics from Virginia Polytech. Institute and State University, USA, in 1986 and is currently attached to the School of Mathematical Sciences, Universiti Sains Malaysia. Her area of research is in designing parallel numerical algorithms, particularly in the numerical solutions of partial differential equations.

Professor Abdul Rahman Abdullah received his Ph.D. from Loughborough University of Technology, UK, in 1983. Currently, he is attached to the Department of Industrial Computing, Faculty of Information Science and Technology, Universiti Kebangsaan Malaysia. His research areas include Computed-Assisted Design(CAD) and parallel processing.