MATEMATIKA, 2005, Jilid 21, Bil. 1, hlm. 61–77

Rotated Outer-Inner Iterative Schemes for The Solution of Burgers' Equation

[1]**Norhashidah Hj. Mohd. Ali &** [2]**Abdul Rahman Abdullah**

[1]School of Mathematical Sciences, Universiti Sains Malaysia
11800 Penang, Malaysia. E-mail: shidah@cs.usm.my

[2]Department of Industrial Computing, FTSM, Universiti Kebangsaan Malaysia
43600 Bangi, Selangor, Malaysia. E-mail: ara@ftsm.ukm.my

Abstract In their paper, Ali and Abdullah introduced new iterative methods based on *rotated(cross)* five-point finite difference discretisation in solving a coupled system of *elliptic* partial differential equations (p.d.e.'s) where these methods were found to be more superior than common existing methods based on the *centred* five-point difference schemes. In this paper, the application of a new iterative schemes derived from the *rotated* finite difference discretisation to the numerical solution of the nonlinear steady two dimensional Burgers' equation is considered. Some numerical experiments are presented and we will show that the new methods are accurate and comparable to the existing finite difference method.

Keywords Numerical methods; Burgers' equation; *rotated* finite difference; explicit decoupled group (EDG) method

Abstrak *Dalam kertas kerja mereka, Ali dan Abdullah telah memperkenalkan kaedah-kaedah lelaran baru berdasarkan pendiskretan beza terhingga lima titik putaran dalam menyelesaikan sistem berpasangan persamaan pembezaan separa eliptik di mana kaedah-kaedah ini telah didapati lebih baik daripada kaedah sedia ada yang berdasarkan skema beza terhingga ke tengah. Dalam kertas ini, aplikasi skema lelaran baru yang diterbitkan daripada pendiskretan beza terhingga putaran ini kepada penyelesaian berangka persamaan Burgers dua dimensi tak linear telah ditinjau. Beberapa ujikaji berangka akan dibentangkan dan kami akan menunjukkan bahawa kaedah-kaedah baru ini adalah jitu dan setanding dengan kaedah beza terhingga yang sedia ada.*

Katakunci *Kaedah berangka; persamaan Burgers; beza terhingga putaran; kaedah kumpulan nyah pasangan tak tersirat (KNPTT)*

1 Introduction

The algebraic equations produced by discretising fluid dynamics governing equations are usually nonlinear due to the nature of the convective terms. To cope with the nonlinearity of the discretised equations, some form of outer-inner iterative procedure is consequently inevitable. This type of procedure has long been investigated by researchers to solve such problems particularly the *Navier-Stokes* and *biharmonic* equations ([6], [7], [9], [10], [11]). Recently a new outer-inner iterative procedure derived from the *rotated* (skewed) finite

difference formulae ([4], [8], [12]) was successfully formulated in solving the two dimensional steady Navier-Stokes equation ([2], [3]). In this paper, we extend the application of this type of outer-inner iterative procedure to the Burgers' equation; one of the most fundamental nonlinear problem in computational fluid dynamics. We proceed as follows. In Section 2, a point iterative scheme based on the *rotated* five-point formula in solving this equation is developed. In Section 3, its group-wise counterpart, i.e. the four-point Explicit Decoupled Group (EDG) scheme, which was firstly introduced in solving the Poisson equation [1], will be described. The numerical experiments of these methods are discussed in Section 4, followed by the theoretical complexity analysis and discussion of results in Sections 5 and 6 respectively.

2 Rotated Point Iterative Algorithm

The problem to be considered will be formulated as follows. Consider the steady two-dimensional Burgers' equations

$$u\frac{\partial u}{\partial x}+v\frac{\partial u}{\partial y}-\frac{1}{Re}\left(\frac{\partial^2 u}{\partial x^2}+\frac{\partial^2 u}{\partial y^2}\right)=0 \qquad [2.1]$$

$$u\frac{\partial v}{\partial x}+v\frac{\partial v}{\partial y}-\frac{1}{Re}\left(\frac{\partial^2 v}{\partial x^2}+\frac{\partial^2 v}{\partial y^2}\right)=0 \qquad [2.2]$$

with Dirichlet boundary conditions on u and v. This equation is considered to be a simplified form of the Navier-Stokes equation, where the pressure term is neglected. Here, Re is the Reynolds number. For our discussion, the solutions u and v that satisfy these equations are sought in the interior region S as shown in Figure 1.

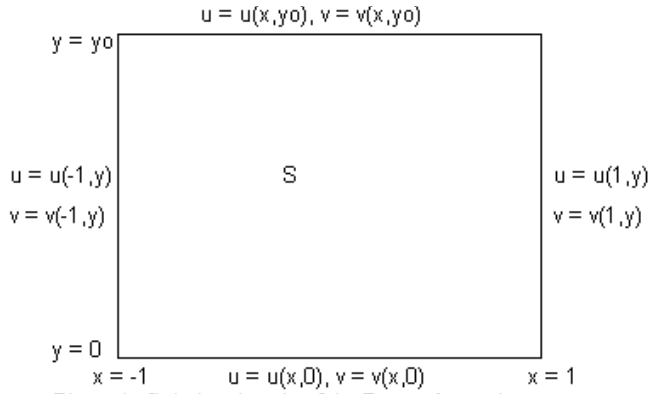

Figure 1 Solution domain of the Burgers' equations

Let n be a fixed positive integer. Determine the grid size h = 2/n so that a uniformly spaced square network ($\Delta x = \Delta y = h$) with $x_i = -1 + ih$, $y_j = jh$, i,j = 0,1,2,...,n, is imposed on S. Using the *centred* difference approximation, Equations [2.1] and [2.2] can now be discretised at the grid point (x_i, y_j) by the following finite difference equations:

$$u_{ij}\left(\frac{u_{i+1,j}-u_{i-1,j}}{2h}\right)+v_{ij}\left(\frac{u_{i,j+1}-u_{i,j-1}}{2h}\right)-\frac{1}{Re}\left(\frac{u_{i-1,j}+u_{i+1,j}+u_{i,j-1}+u_{i,j+1}-4u_{ij}}{h^2}\right)+ \frac{h^2}{3!}\left(\frac{1}{2Re}\frac{\partial^4 u}{\partial x^4}+\frac{1}{2Re}\frac{\partial^4 u}{\partial y^4}-u_{ij}\frac{\partial^3 u}{\partial x^3}-v_{ij}\frac{\partial^3 u}{\partial y^3}-...\right)=0 \quad [2.3]$$

$$u_{ij}\left(\frac{v_{i+1,j}-v_{i-1,j}}{2h}\right)+v_{ij}\left(\frac{v_{i,j+1}-v_{i,j-1}}{2h}\right)-\frac{1}{Re}\left(\frac{v_{i-1,j}+v_{i+1,j}+v_{i,j-1}+v_{i,j+1}-4v_{ij}}{h^2}\right)+ \frac{h^2}{3!}\left(\frac{1}{2Re}\frac{\partial^4 v}{\partial x^4}+\frac{1}{2Re}\frac{\partial^4 v}{\partial y^4}-u_{ij}\frac{\partial^3 v}{\partial x^3}-v_{ij}\frac{\partial^3 v}{\partial y^3}-...\right)=0 \quad [2.4]$$

Neglecting the error terms, the following forms can be obtained from [2.3] and [2.4]

$$\left[\left(\frac{u_{i+1,j}-u_{i-1,j}}{2h}\right)+\frac{4}{Reh^2}\right]u_{ij} = \frac{1}{Reh^2}\left(u_{i-1,j}+u_{i+1,j}+u_{i,j-1}+u_{i,j+1}\right)-v_{ij}\left(\frac{u_{i,j+1}-u_{i,j-1}}{2h}\right), \quad [2.5]$$

$$\left[\left(\frac{v_{i,j+1}-v_{i,j-1}}{2h}\right)+\frac{4}{Reh^2}\right]v_{ij} = \frac{1}{Reh^2}\left(v_{i-1,j}+v_{i+1,j}+v_{i,j-1}+v_{i,j+1}\right)-u_{ij}\left(\frac{v_{i+1,j}-v_{i-1,j}}{2h}\right), \quad [2.6]$$

i,j = 1,2,3, ..., n.

Observe that if v is known, then we can solve [2.5] iteratively for u, while if u is known, we can solve [2.6] iteratively for v, and vice versa. Analogous to the schemes presented for the Navier-Stokes problem ([2], [3]), we can devise a similar algorithm by first making initial guesses $u_{ij}^{(0)}$ and $v_{ij}^{(0)}$, and then generate an alternating sequence of outer iterates:

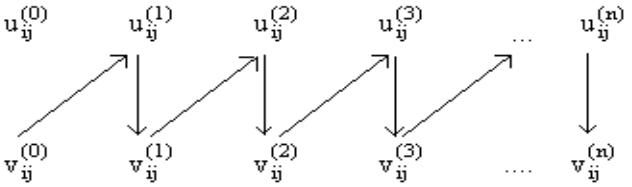

Figure 2 Alternating sequence of outer iterates

The iteration is continued until for some k, $\left|u_{ij}^{(k+1)}-u_{ij}^{(k)}\right|<\delta$ and $\left|v_{ij}^{(k+1)}-v_{ij}^{(k)}\right|<\delta$ for some given tolerance δ. The solutions $u_{ij}^{(k+1)}$ and $v_{ij}^{(k+1)}$ generated are then taken to be the numerical solutions of the given problem.

Another approximation may be obtained by exploiting the four points (i±1, j±1) and (i∓1, j±1) located on the skewed (*rotated*) stencil ([4], [8], [12]). The resulting difference approximations for the Burgers' equations can thus be shown to be as the following:

$$\left[\frac{Reh(u_{i+1,j-1}-u_{i-1,j+1}+u_{i+1,j+1}-u_{i-1,j-1})+8}{4Reh^2}\right]u_{ij}+\left(\frac{v_{ij}}{4h}-\frac{1}{2Reh^2}\right)u_{i-1,j+1}+ \left(\frac{-v_{ij}}{4h}-\frac{1}{2Reh^2}\right)u_{i+1,j-1}+\left(\frac{-v_{ij}}{4h}-\frac{1}{2Reh^2}\right)u_{i-1,j-1}+\left(\frac{v_{ij}}{4h}-\frac{1}{2Reh^2}\right)u_{i+1,j+1}=0 \quad [2.7]$$

$$\left[\frac{\mathrm{Reh}(v_{i+1,j+1}-v_{i-1,j-1}-v_{i+1,j-1}+v_{i-1,j+1})+8}{4\mathrm{Reh}^2}\right] v_{ij} + \left(\frac{-u_{ij}}{4h} - \frac{1}{2\mathrm{Reh}^2}\right) v_{i-1,j+1} + \left(\frac{u_{ij}}{4h} - \frac{1}{2\mathrm{Reh}^2}\right) v_{i+1,j-1} + \left(\frac{-u_{ij}}{4h} - \frac{1}{2\mathrm{Reh}^2}\right) v_{i-1,j-1} + \left(\frac{u_{ij}}{4h} - \frac{1}{2\mathrm{Reh}^2}\right) v_{i+1,j+1} = 0 \quad [2.8]$$

A point iterative scheme based on the *rotated* difference equations [2.7] and [2.8] can now be constructed for the solution of the given problem. Lets consider all alternate points on the x-y plane as shown in Figure 3. Because the evaluations in [2.7] (and also [2.8]) involve points of type ◯ only, iterations can be carried out using the points of this type only.

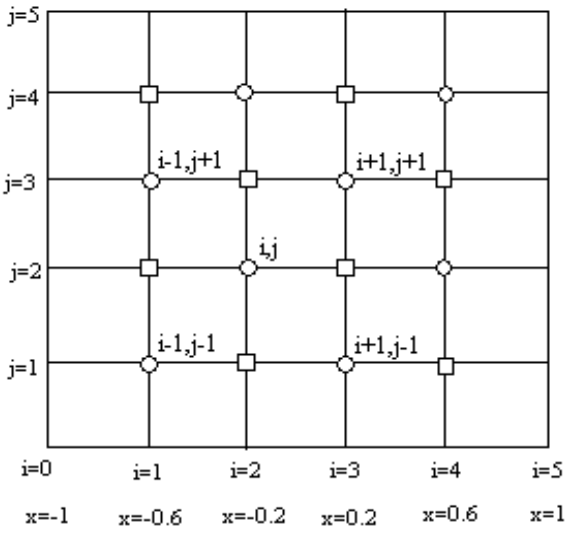

Figure 3 *Rotated* point iterative scheme for n=5

There are 16 interior grid points to be computed. The iterations on u (assuming v is known) using Equation [2.7] are done only on points of type ◯. Iterations on v using Equation [2.8] are also done in a similar way using the values of u just computed. The iterations can also be done in reverse order, i.e. do the iterations on v first using Equation [2.8], then use these most recent values of v to iterate on u using Equation [2.7]. These points undergo iterations until a chosen convergence criteria is met. Only after the iterations have converged, the values of the solutions at the remaining points (of the type □) will be computed directly once using formula [2.5] (or [2.6]) of *centred* difference approximations. In our example, the computation of the eight points of type ◯ using Equation [2.7] can be represented in matrix form [2.9]-[2.10]. The computation of the eight points using Equation [2.8] can also be represented in a similar way. Here, we let $c = \frac{1}{4h}$ and $d = \frac{1}{2\mathrm{Reh}^2}$. We can now formulate the *rotated* point outer-inner iterative method in solving the problem [2.1]-[2.2] as follows:

Algorithm 1: Rotated point outer-inner iterative method

(i) Divide the solution domain into grid lines by choosing $h = \frac{2}{n}$. Define the group of points which will be involved in the iterative computations (i.e. points of type ◯ and □).

(ii) Set $u_{ij}^{(0)} = v_{ij}^{(0)} = 0$ as initial approximation for the outer iterates.

(iii) Generate approximations $v_{ij}^{(k+1)}$ and $u_{ij}^{(k+1)}$ for k = 0,1,2,... as follows:

(a) Generate $v_{ij}^{(k+1)}$ (or $u_{ij}^{(k+1)}$) of Equation [2.8] (of Equation [2.7]) using the *rotated* point iterative scheme described previously for a prescribed tolerance ε.

(b) Generate $u_{ij}^{(k+1)}$ (or $v_{ij}^{(k+1)}$) of Equation [2.7] (of Equation [2.8]) using the *rotated* point iterative scheme for the prescribed tolerance ε.
(Here, use the recently obtained $v_{ij}^{(k+1)}$ (or $u_{ij}^{(k+1)}$) in (a) for the values v_{ij}(or u_{ij}) in the formulae).

(c) Store the converged values of both inner iterative schemes $u_{ij}^{(k+1)} \rightarrow \text{outer\_}u_{ij}^{(r)}$ and$v_{ij}^{(k+1)} \rightarrow \text{outer_}v_{ij}^{(r)}$.

(iv) Check the convergence of the outer iteration over the whole interior mesh points, i.e. check whether

$$\max\left\{ \left|\text{outer\_}u_{ij}^{(r+1)} - \text{outer\_}u_{ij}^{(r)}\right|, \left|\text{outer\_}v_{ij}^{(r+1)} - \text{outer\_}v_{ij}^{(r)}\right| \right\} \leq \delta,$$

for a prescribed termination criteria δ. If yes, the numerical solution of the given problem is given by $\text{outer_}v_{ij}^{(r+1)}$and $\text{outer\_}u_{ij}^{(r+1)}$. Otherwise go back to Step (iii).

$$\mathbf{A} \times \begin{bmatrix} u_{11} \\ u_{31} \\ u_{22} \\ u_{42} \\ u_{13} \\ u_{33} \\ u_{24} \\ u_{44} \end{bmatrix} = \begin{bmatrix} b_{11} \\ b_{31} \\ b_{22} \\ b_{42} \\ b_{13} \\ b_{33} \\ b_{24} \\ b_{44} \end{bmatrix}, \qquad [2.9]$$

where

$$\begin{aligned}
b_{11} &= (cv_{11}+d)u_{20} - (cv_{11}-d)u_{02} + (cv_{11}+d)u_{00} \\
b_{31} &= (cv_{31}+d)u_{40} + (cv_{31}+d)u_{20} \\
b_{22} &= 0 \\
b_{42} &= (cv_{42}+d)u_{51} - (cv_{42}-d)u_{53} \\
b_{13} &= (cv_{13}+d)u_{02} - (cv_{13}-d)u_{04} \\
b_{33} &= 0 \\
b_{24} &= -(cv_{24}-d)u_{15} - (cv_{24}-d)u_{35} \\
b_{44} &= (cv_{44}+d)u_{53} - (cv_{44}-d)u_{55} - (cv_{44}-d)u_{35}
\end{aligned} \qquad [2.10]$$

$$\mathbf{A} = \begin{bmatrix} c(u_{20}-u_{02}+u_{22}-u_{00})+4d & 0 & cv_{11}-d & 0 \\ 0 & c(u_{40}-u_{22}+u_{42}-u_{20})+4d & cv_{31}-d & cv_{31}-d \\ -cv_{22}-d & -cv_{22}-d & c(u_{31}-u_{13}+u_{33}-u_{11})+4d & 0 \\ 0 & -cv_{42}-d & 0 & c(u_{51}-u_{33}+u_{53}-u_{31})+4d \\ 0 & 0 & -cv_{13}-d & 0 \\ 0 & 0 & -cv_{33}-d & -cv_{33}-d \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

$$\begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ cv_{23}-d & cv_{22}-d & 0 & 0 \\ 0 & cv_{42}-d & 0 & 0 \\ c(u_{22}-u_{04}+u_{24}-u_{02})+4d & 0 & cv_{13}-d & 0 \\ 0 & c(u_{42}-u_{24}+u_{44}-u_{22})+4d & cv_{33}-d & cv_{33}-d \\ -cv_{24}-d & -cv_{24}-d & c(u_{33}-u_{15}+u_{35}-u_{13})+4d & 0 \\ 0 & -cv_{44}-d & 0 & c(u_{53}-u_{35}+u_{55}-u_{33})+4d \end{bmatrix}$$

3 Four-Point Edg Iterative Algorithm

Adopting the same idea as the four-point EDG in [2] and [3], the four-point EDG for solving the problem [2.1]-[2.2] can now be formulated. Without loss of generality, assume the generation of $v_{ij}^{(k+1)}$ is done first using Equation [2.8] followed by the generation of $u_{ij}^{(k+1)}$ using Equation [2.7]. Using the *rotated* finite difference approximation [2.8] for v_{ij}, any group of four points on a discretised solution domain can be solved resulting in a (4x4) system of equations as shown in [3.1]-[3.2].

$$\mathbf{B} \times \begin{bmatrix} v_{ij} \\ v_{i+1,j+1} \\ v_{i+1,j} \\ v_{i,j+1} \end{bmatrix} = \begin{bmatrix} rhs_{ij} \\ rhs_{i+1,j+1} \\ rhs_{i+1,j} \\ rhs_{i,j+1} \end{bmatrix} \qquad [3.1]$$

where

$$\begin{aligned} rhs_{ij} &= (cu_{ij}+d)v_{i-1,j+1}+(-cu_{ij}+d)v_{i+1,j-1}+(cu_{ij}+d)v_{i-1,j-1} \\ rhs_{i+1,j+1} &= (cu_{i+1,j+1}+d)v_{i,j+2}+(-cu_{i+1,j+1}+d)v_{i+2,j}+(-cu_{i+1,j+1}+d)v_{i+2,j+2} \\ rhs_{i+1,j} &= (-cu_{i+1,j}+d)v_{i+2,j+1}+(-cu_{i+1,j}+d)v_{i+2,j-1}+(cu_{i+1,j}+d)v_{i,j-1} \\ rhs_{i,j+1} &= (cu_{i,j+1}+d)v_{i-1,j+2}+(-cu_{i,j+1}+d)v_{i+1,j+2}+(cu_{i,j+1}+d)v_{i-1,j}. \end{aligned} \qquad [3.2]$$

$$\mathbf{B} = \begin{bmatrix} \dfrac{\mathrm{Reh}(v_{i+1,j+1} - v_{i-1,j-1} - v_{i+1,j-1} + v_{i-1,j+1}) + 8}{4\mathrm{Reh}^2} & cu_{ij} - d \\ -cu_{i+1,j+1} - d & \dfrac{\mathrm{Reh}(v_{i+2,j+2} - v_{ij} - v_{i+2j} + v_{i,j+2}) + 8}{4\mathrm{Reh}^2} \end{bmatrix}$$

$$\begin{bmatrix} \dfrac{\mathrm{Reh}(v_{i+2,j+1} - v_{i,j-1} - v_{i+2,j-1} + v_{i,j+1}) + 8}{4\mathrm{Reh}^2} & -cu_{i+1,j} - d \\ cu_{i,j+1} - d & \dfrac{\mathrm{Reh}(v_{i+1,j+2} - v_{i-1,j} - v_{i+1,j} + v_{i-1,j+2}) + 8}{4\mathrm{Reh}^2} \end{bmatrix}$$

The system [3.1]-[3.2] leads to a *decoupled* system of (2x2) equations which can be made explicit as follows:

$$\begin{bmatrix} v_{ij} \\ v_{i+1,j+1} \end{bmatrix}^{(k+1)} =$$

$$\frac{(4\mathrm{Reh}^2)^2}{\left[\mathrm{Reh}(v^{(k)}_{i+1,j+1} - v_{i-1,j-1} - v_{i+1,j-1} + v_{i-1,j+1}) + 8\right]\left[\mathrm{Reh}(v_{i+2,j+2} - v^{(k)}_{i,j} - v_{i+2,j} + v_{i,j+2}) + 8\right] + (\mathrm{Rehu}_{i+1,j+1} + 2)(\mathrm{Rehu}_{ij} - 2)}$$

$$\times \begin{bmatrix} \dfrac{\mathrm{Reh}(v_{i+2,j+2} - v^{(k)}_{ij} - v_{i+2,j} + v_{i,j+2}) + 8}{4\mathrm{Reh}^2} & -cu_{ij} + d \\ cu_{i+1,j+1} + d & \dfrac{\mathrm{Reh}(v^{(k)}_{i+1,j+1} - v_{i-1,j-1} - v_{i+1,j-1} + v_{i-1,j+1}) + 8}{4\mathrm{Reh}^2} \end{bmatrix} \begin{bmatrix} rhs_{ij} \\ rhs_{i+1,j+1} \end{bmatrix} \quad [3.3]$$

and

$$\begin{bmatrix} v_{ij} \\ v_{i+1,j+1} \end{bmatrix}^{(k+1)} =$$

$$\frac{(4\mathrm{Reh}^2)^2}{\left[\mathrm{Reh}(v^{(k)}_{i+1,j+1} - v_{i-1,j-1} - v_{i+1,j-1} + v_{i-1,j+1}) + 8\right]\left[\mathrm{Reh}(v_{i+2,j+2} - v^{(k)}_{i,j} - v_{i+2,j} + v_{i,j+2}) + 8\right] + (\mathrm{Rehu}_{i+1,j+1} + 2)(\mathrm{Rehu}_{ij} - 2)}$$

$$\times \begin{bmatrix} \dfrac{\mathrm{Reh}(v_{i+1,j+2} - v^{(k)}_{i+1,j} - v_{i-1,j} + v_{i-1,j+2}) + 8}{4\mathrm{Reh}^2} & cu_{i+1,j} + d \\ -cu_{i,j+1} + d & \dfrac{\mathrm{Reh}(v^{(k)}_{i,j+1} - v_{i,j-1} - v_{i+2,j-1} + v_{i+2,j+1}) + 8}{4\mathrm{Reh}^2} \end{bmatrix} \begin{bmatrix} rhs_{i+1,j} \\ rhs_{i,j+1} \end{bmatrix}. \quad [3.4]$$

Similarly, from the generation of u_{ij} using Equation [2.7], a (4x4) system of equations can be formed as shown in [3.5]-[3.6]:

$$\mathbf{C} \times \begin{bmatrix} u_{ij} \\ u_{i+1,j+1} \\ u_{i+1,j} \\ u_{i,j+1} \end{bmatrix} = \begin{bmatrix} rhs_{ij}^{*} \\ rhs_{i+1,j+1}^{*} \\ rhs_{i+1,j}^{*} \\ rhs_{i,j+1}^{*} \end{bmatrix} \qquad [3.5]$$

where

$$\mathbf{C} = \begin{bmatrix} \frac{Reh(u_{i+1,j-1} - u_{i-1,j+1} + u_{i+1,j+1} - u_{i-1,j-1}) + 8}{4Reh^2} & cv_{ij} - d \\ -cv_{i+1,j+1} - d & \frac{Reh(u_{i+2,j} - u_{i,j+2} + u_{i+2,j+2} - u_{ij}) + 8}{4Reh^2} \end{bmatrix}$$

$$\begin{bmatrix} \frac{Reh(u_{i+2,j-1} - u_{i,j+1} + u_{i+2,j+1} - u_{i,j-1}) + 8}{4Reh^2} & cv_{i+1,j} - d \\ -cv_{i,j+1} - d & \frac{Reh(u_{i+1,j} - u_{i-1,j+2} + u_{i+1,j+2} - u_{i-1,j}) + 8}{4Reh^2} \end{bmatrix}$$

$$\begin{aligned} rhs_{ij}^{*} &= (-cv_{ij}+d)u_{i-1,j+1}+(cv_{ij}+d)u_{i+1,j-1}+(cv_{ij}+d)u_{i-1,j-1} \\ rhs_{i+1,j+1}^{*} &= (-cv_{i+1,j+1}+d)u_{i,j+2}+(cv_{i+1,j+1}+d)u_{i+2,j}+(-cv_{i+1,j+1}+d)u_{i+2,j+2} \\ rhs_{i+1,j}^{*} &= (-cv_{i+1,j}+d)u_{i+2,j+1}+(cv_{i+1,j}+d)u_{i+2,j-1}+(cv_{i+1,j}+d)u_{i,j-1} \\ rhs_{i,j+1}^{*} &= (-cu_{i,j+1}+d)u_{i-1,j+2}+(-cv_{i,j+1}+d)u_{i+1,j+2}+(cv_{i,j+1}+d)u_{i-1,j} \end{aligned} \qquad [3.6]$$

The system [3.5]-[3.6] leads to a decoupled system of (2x2) equations whose explicit forms are given by:

$$\begin{bmatrix} u_{ij} \\ u_{i+1,j+1} \end{bmatrix}^{(k+1)} =$$

$$\frac{(4Reh^2)^2}{\left[Reh(u_{i+1,j-1}-u_{i-1,j+1}+u_{i+1,j+1}^{(k)}-u_{i-1,j-1}) + 8\right]\left[Reh(u_{i+2,j}-u_{i,j+2}+u_{i+2,j+2}-u_{ij}^{(k)}) + 8\right] + (Rehv_{i+1,j+1} + 2)(Rehv_{ij} - 2)}$$

$$\times \begin{bmatrix} \frac{Reh(u_{i+2,j}-u_{i,j+2}+u_{i+2,j+2}-u_{ij})+8}{4Reh^2} & -cv_{ij} + d \\ cv_{i+1,j+1}+d & \frac{Reh(u_{i+1,j-1}-u_{i-1,j+1}+u_{i+1,j+1}-u_{i-1,j-1})+8}{4Reh^2} \end{bmatrix} \begin{bmatrix} rhs_{ij}* \\ rhs_{i+1,j+1}* \end{bmatrix} \qquad [3.7]$$

and

$$\begin{bmatrix} u_{i+1,j} \\ u_{i,j+1} \end{bmatrix}^{(k+1)} =$$

$$\frac{(4\mathrm{Reh}^2)^2}{\left[\mathrm{Reh}(u_{i+2,j-1}+u_{i+2,j+1}-u^{(k)}_{i,j+1}-u_{i,j-1})+8\right]\left[\mathrm{Reh}(u^{(k)}_{i+1,j}-u_{i-1,j+2}-u_{i-1,j}+u_{i+1,j+2})+8\right]+(\mathrm{Rehv}_{i,j+1}+2)(\mathrm{Rehv}_{i+1,j}-2)}$$

$$\times \begin{bmatrix} \frac{\mathrm{Reh}(u_{i+1,j}-u_{i-1,j+2}+u_{i+1,j+2}-u_{i-1,j})+8}{4\mathrm{Reh}^2} & -cv_{i+1,j}+d \\ cv_{i,j+1}+d & \frac{\mathrm{Reh}(u_{i+2,j-1}-u_{i,j+1}+u_{i+2,j+1}-u_{i,j-1})+8}{4\mathrm{Reh}^2} \end{bmatrix} \begin{bmatrix} rhs_{i+1,j}* \\ rhs_{i,j+1}* \end{bmatrix} \quad [3.8]$$

Suppose we choose to do the iterations at *half* of the points on the solution domain using [3.3] and [3.7]. Hence, we can define the four-point EDG outer-inner scheme to solve the Burgers' Equations [2.1]-[2.2] as follows:

Algorithm 2: Four-point EDG outer-inner iterative method

(i) Divide the solution domain into grids with even number of lines. Define the groups of points ◯ and □.

(ii) Choose h and set $u_{ij}^{(0)} = v_{ij}^{(0)} = 0$ as initial approximations for the outer iterates.

(iii) Generate sequences $v_{ij}^{(k+1)}$ and $u_{ij}^{(k+1)}$ for k = 0,1,2,... as follows:

(a) Generate $v_{ij}^{(k+1)}$ by the four-point EDG inner iterative scheme using Equation [3.3] on *half* of the total nodal points (namely the points of type ◯), for a prescribed termination criterion ε. After the inner iteration converges, compute the v's on the remaining points (of the type □) directly using Equation [2.6]. (For the term u_{ij} in [3.3], use the most recent value available for u_{ij}'s.)

(b) Generate $u_{ij}^{(k+1)}$ by the four-point EDG inner iterative scheme using Equation [3.7] also on *half* of the total nodal points (type ◯) for a prescribed tolerance ε. Similarly, compute the u_{ij}'s on the points of type □ directly once using Equation [2.5] after the inner iteration converges. (Here, use the recently obtained $v_{ij}^{(k+1)}$ in (a) for the values in Equation [3.7]).

(c) Store the converged values of both inner iterative schemes

$$u_{ij}^{(k+1)} \rightarrow \mathrm{outer\_u}_{ij}^{(r)} \mathrm{and} v_{ij}^{(k+1)} \rightarrow \mathrm{outer\_v}_{ij}^{(r)}.$$

(iv) Check the convergence of the outer iteration over the whole interior mesh points checking whether the following condition is satisfied:

$$\max\left\{\left|\mathrm{outer\_u}_{ij}^{(r+1)} - \mathrm{outer\_u}_{ij}^{(r)}\right|, \left|\mathrm{outer\_v}_{ij}^{(r+1)} - \mathrm{outer\_v}_{ij}^{(r)}\right|\right\} \leq \delta,$$

for a specific tolerance δ. If its converges, stop the outer iterative process and the solutions to the problem are given by $\mathrm{outer\_v}_{ij}^{(r+1)}$ and $\mathrm{outer\_u}_{ij}^{(r+1)}$. Otherwise, go back to Step (iii).

Observe that the computational molecule for this scheme is similar to the ones described in the EDG method for problems in [1]. The iterative evaluation of [3.3] and [3.7] involve points of type ◯ only, while Equations [3.4] and [3.8] can be evaluated using points of type □ only. As a result, the iteration for each case can be carried out on either one of the two types of points. After convergence is achieved, the solution at the other half of the points is evaluated directly once using the centred difference formulae [2.5] or [2.6].

4 Numerical Results

For our experimental work, we consider the Burgers' Equations [2.1]-[2.2] with the exact solutions [5]

$$\begin{aligned} \mathrm{u} &= \frac{-2(a_2 + a_4 y + \lambda a_5 \cos \lambda y (e^{\lambda(x-x_0)} - e^{-\lambda(x-x_0)}))}{\mathrm{Re}(a_1 + a_2 x + a_3 y + a_4 xy + a_5 (e^{\lambda(x-x_0)} + e^{-\lambda(x-x_0)}) \cos \lambda y)} \\ \mathrm{v} &= \frac{-2(a_3 + a_4 x - \lambda a_5 \sin \lambda y (e^{\lambda(x-x_0)} + e^{-\lambda(x-x_0)}))}{\mathrm{Re}(a_1 + a_2 x + a_3 y + a_4 xy + a_5 (e^{\lambda(x-x_0)} + e^{-\lambda(x-x_0)}) \cos \lambda y)}, \; -1 \le x \le 1, 0 \le y \le 2 \end{aligned} \qquad [4.1]$$

with the boundary conditions satisfying the exact solutions. Here, $a_1, a_2, a_3, a_4, a_5, \lambda$ and x_o can be chosen to produce different behaviour of the exact solutions. For our experiments, we randomly chose $a_1 = a_2 = 1.0, a_3 = a_4 = 0.0, a_5 = x_\mathrm{o} = 1.0$, and $\lambda = 0.3$ for Re = 10, 100 and 1000. The numerical results obtained from the *rotated* point outer-inner scheme and four-point EDG are compared with the *centred* difference outer-inner scheme, i.e. the inner iterative procedure is based on the centred difference formula [2.5]-[2.6] where iterations are done using all the nodal points. The programming language used was Fortran. Throughout the experiment, $\delta = \varepsilon = 10^{-11}$ was used as the termination criteria for both the outer and inner iterations. Also, for all the methods, the relaxation parameter, ω, was obtained experimentally to within ± 0.01 which gives the most rapid convergence. The results obtained from the three methods with various sizes of n and for Re = 10, 100 and 1000 are shown in TABLES 1 - 3.

5 Computational Complexity

In order to measure the computational complexity of the three methods, we shall obtain an estimate of the amount of computational work required from each inner iteration process on v and u. Assume that there are m^2 internal mesh points (where m = n−1) and the execution times for the *adds, mults* and *divs* operations are roughly the same. We estimate the computational work in terms of arithmetic operations performed per inner iteration on v and u.

The Point *Centred* Difference Scheme

Consider Equation [2.6] for the inner iteration on v. From the simplification of this equation, we obtain

$$\begin{aligned} &\mathrm{s}_1 = \mathrm{a}_1/(8 + \mathrm{a}_2^*(\mathrm{v}_{\mathrm{i,j+1}}^{(\mathrm{k})} - \mathrm{v}_{\mathrm{i,j-1}}^{(\mathrm{k+1})})) \\ &\mathrm{s}_2 = \mathrm{a}_3 + \mathrm{a}_4^* \mathrm{u}_{\mathrm{ij}} \qquad\qquad\qquad\qquad \mathrm{s}_3 = \mathrm{a}_3 - \mathrm{a}_4^* \mathrm{u}_{\mathrm{ij}} \end{aligned} \qquad [5.1]$$

where $a_1 = 2Reh^2$, $a_2 = Reh$, $a_3 = 1/Reh^2$ and $a_4 = 1/(2h)$ are assumed stored beforehand. The Gauss-Seidel form of [2.6] is given by

$$\tilde{v}_{ij}^{(k+1)} = s_1 * (a_3 * (v_{i,j-1}^{(k+1)} + v_{i,j+1}^{(k)}) + s_2 * v_{i-1,j}^{(k+1)} + s_3 * v_{i+1,j}^{(k)}). \quad [5.2]$$

Therefore, the number of operations required for m2 internal mesh points including the over-relaxation process

$$v_{ij}^{(k+1)} = \omega * (\tilde{v}_{ij}^{(k+1)} - v_{ij}^{(k)}) + v_{ij}^{(k)}, \ (\ i,j = 1,2,...,m) \quad [5.3]$$

(but excluding the convergence test) is

$$18m^2 \text{ operations per inner iteration,} \quad [5.4]$$

since there are 9 adds, 8 mults and 1 div per point per inner iteration. In a similar manner, we can estimate the operation counts for the inner iteration on u from Equation [2.5], which was found to be the same as for v.

The Rotated Point Difference Scheme

The rotated formula [2.8] for the iteration on v, can be written in explicit form as

$$\begin{aligned} v_{ij}^{(k+1)} = & \frac{4Reh^2}{[Re(v_{i+1,j+1}^{(k)} - v_{i-1,j-1}^{(k+1)} - v_{i+1,j-1}^{(k+1)} + v_{i-1,j+1}^{(k)}) + 8]} \times \\ & \left[\left(\frac{1}{2Reh^2} - \frac{u_{ij}}{4h} \right) \left(v_{i+1,j-1}^{(k+1)} + v_{i+1,j+1}^{(k)} \right) + \left(\frac{1}{2Reh^2} + \frac{u_{ij}}{4h} \right) \left(v_{i-1,j+1}^{(k)} + v_{i-1,j-1}^{(k+1)} \right) \right]. \end{aligned} \quad [5.5]$$

Recall that the iteration is carried out on half of the mesh points, the other $m^2/2$ are solved directly once using the centred difference formula [2.6].

Let

$$\begin{aligned} & a_1 = 4Reh^2, a_2 = Reh, a_3 = 1/2Reh^2, a_4 = 1/(4h), \\ & b_1 = a_1/(a2^*(v_{i+1,j+1}^{(k)} - v_{i-1,j-1}^{(k+1)} - v_{i+1,j-1}^{(k+1)} + v_{i-1,j+1}^{(k)}) + 8) \\ & b_2 = a_3 - a_4^* u_{ij} \\ & b_3 = a_3 + a_4^* u_{ij}. \end{aligned} \quad [5.6]$$

Thus, the (k+1)th iterate of the SOR iterative scheme is defined by

$$v_{ij}^{(k+1)} = \omega * (\tilde{v}_{ij}^{(k+1)} - v_{ij}^{(k)}) + v_{ij}^{(k)}, \quad (i,j = 1,(2),n\text{-}1) \quad [5.7]$$

$\tilde{v}_{ij}^{(k+1)}$ represents the components of the (k+1)th Gauss-Seidel iteration defined by

$$\tilde{v}_{ij}^{(k+1)} = b_1 * (b_2 * (v_{i+1,j-1}^{(k+1)} + v_{i+1,j+1}^{(k)}) + b_3 * (v_{i-1,j+1}^{(k)} + v_{i-1,j-1}^{(k+1)})). \quad [5.8]$$

This process requires 11 *adds,* 7 *mults* and 1 *divs* (19 operations) for one mesh point per inner iteration, assuming the constants a1, a2, a3 and a4 are stored beforehand. Referring to [5.1]-[5.2], we can also estimate the number of operations after convergence is achieved. Specifically, 15 operations per point are needed to calculate the remaining points after

TABLE 1: Iterative count and CPU measure of the *centred* difference outer-inner scheme

n	Re	w	Ave-Abs Error for u	Ave-Abs Error for v	Number of outer iterates	Number of inner iter. for v	Number of inner iter. for u	Time (secs)
25	10	1.78	1.46E-07	6.65E-08	1 2 3 4	102 83 49 1	105 62 25 1	68.33
	100	1.78	1.46E-08	6.65E-09	1 2 3 4	100 70 35 1	101 54 6 1	58.12
	1000	1.78	1.46E-09	6.58E-10	1 2 3 4	78 58 24 1	92 50 1 1	48.83
37	10	1.85	6.87E-08	3.12E-08	1 2 3 4	152 120 68 1	154 90 33 1	218.54
	100	1.85	6.85E-09	3.12E-09	1 2 3 4	149 107 48 1	150 76 3 1	191.09
	1000	1.84	6.75E-10	2.94E-10	1 2 3 4	127 99 32 1	138 70 1 1	165.46
49	10	1.88	3.97E-08	1.80E-08	1 2 3 4	199 146 81 1	201 120 39 1	492.66
	100	1.88	3.93E-09	1.79E-09	1 2 3 4	196 136 63 1	198 102 1 1	436.40
	1000	1.88	3.84E-10	1.55E-10	1 2 3 4	150 114 38 1	176 87 1 1	354.95
61	10	1.91	2.58E-08	1.18E-08	1 2 3 4	248 196 109 1	258 147 45 1	983.65
	100	1.91	2.56E-09	1.18E-09	1 2 3 4	246 169 73 1	248 123 1 1	850.07
	1000	1.90	2.35E-10	7.95E-11	1 2 3 4	201 155 43 1	222 108 1 1	712.68

TABLE 2: Iterative count and CPU measure of the *rotated* point outer-inner scheme

n	Re	w	Ave-Abs Error for u	Ave-Abs Error for v	Number of outer iterates	Number of inner iter. for v	Number of inner iter. for u	Time (secs)
25	10	1.71	1.78E-07	1.09E-07	1 2 3 4	75 58 31 1	74 45 21 1	24.48
	100	1.71	1.78E-08	1.09E-08	1 2 3 4	65 53 26 1	71 34 7 1	22.51
	1000	1.70	1.77E-09	1.09E-09	1 2 3 4	58 47 21 1	62 34 1 1	19.62
37	10	1.79	8.34E-08	5.14E-08	1 2 3 4	111 85 44 1	112 64 29 1	84.62
	100	1.79	8.32E-09	5.15E-09	1 2 3 4	97 75 34 2	101 55 7 1	71.01
	1000	1.79	8.27E-10	5.19E-10	1 2 3 4	88 64 28 1	90 47 1 1	61.64
49	10	1.84	4.82E-08	2.98E-08	1 2 3 4	148 110 56 1	150 85 35 1	195.09
	100	1.84	4.80E-09	2.98E-09	1 2 3 4	127 99 47 1	133 68 5 1	141.94
	1000	1.84	4.74E-10	3.04E-10	1 2 3 4	115 84 34 1	118 58 1 1	138.40
61	10	1.87	3.13E-08	1.94E-08	1 2 3 4	184 134 67 1	187 105 41 1	373.48
	100	1.87	3.11E-09	1.94E-09	1 2 3 4	141 123 57 1	148 83 3 1	308.70
	1000	1.87	3.05E-10	2.03E-10	1 2 3 4	142 107 39 1	151 69 1 1	245.03

convergence. Hence, the number of operations required (excluding the convergence test) for the rotated point method is

$$\begin{gathered} 9.5\mathrm{m}^2 \text{ operations per inner iteration} \\ + \\ 7.5\mathrm{m}^2 \text{ operations after convergence.} \end{gathered} \qquad [5.9]$$

Once again, the estimates for u using Equation [2.7] coincides with v.

The Four-point EDG Scheme

Referring to Equation [3.3] for the iteration on v, let

$$\begin{aligned} &a_1 = 4\mathrm{Reh}^2, a_2 = \mathrm{Reh}, a_3 = 1/2\mathrm{Reh}^2, a_4 = 1/(4h), a_5 = 2\mathrm{Reh}^2, a_6 = 1/\mathrm{Reh}^2, a_7 = 1/(2h), \\ &b_1 = a_2 * (v^{(k)}_{i+1,j+1} - v^{(k+1)}_{i-1,j-1} - v^{(k+1)}_{i+1,j-1} + v^{(k+1)}_{i-1,j+1}) + 8 \\ &b_2 = a_2 * (v^{(k)}_{i+2,j+2} - v^{(k)}_{i,j} - v^{(k)}_{i+2,j} + v^{(k)}_{i,j+2}) + 8 \\ &b_3 = a^*_2 u_{i+1,j+1} + 2 \\ &b_4 = a^*_2 u_{ij} - 2 \\ &b_5 = a_1/(b^*_1 b_2 + b^*_3 b_4) \\ &c_1 = a^*_4 u_{ij} \\ &c_2 = a^*_4 u_{i+1,j+1} \\ &s_1 = (c_1 + a_3) * (v^{(k+1)}_{i-1,j+1} + v^{(k+1)}_{i-1,j-1}) + (a_3 - c_1) * v^{(k+1)}_{i+1,j-1} \\ &s_2 = (a_3 - c_2) * (v^{(k)}_{i+2,j} + v^{(k)}_{i+2,j+2}) + (a_3 + c_2) * v^{(k)}_{i,j+2}. \end{aligned} \qquad [5.10]$$

Here, we assume all the a_i's (i = 1,2,...7) are stored beforehand. Then, Equation [3.3] becomes

$$\begin{bmatrix} \tilde{v}^{(k+1)}_{ij} \\ \tilde{v}^{(k+1)}_{i+1,j+1} \end{bmatrix} = \begin{bmatrix} b_5 * (b_2 * s_1 - b_4 * s_2) \\ b_5 * (b_3 * s_1 + b_1 * s_2 \end{bmatrix}, \quad i,j = 1,(2),n-1. \qquad [5.11]$$

The (k+1)th iterate of the SOR variant is then given by

$$\begin{aligned} &v^{(k+1)}_{ij} = \omega * (\tilde{v}^{(k+1)}_{ij} - v^{(k)}_{ij}) + v^{(k)}_{ij}, \\ &v^{(k+1)}_{i+1,j+1} = \omega * \left(\tilde{v}^{(k+1)}_{i+1,j+1} - v^{(k)}_{i+1,j+1} \right) + v^{(k)}_{i+1,j+1}. \end{aligned} \qquad [5.12]$$

There are 25 *adds,* 20 *mults* and 1 *div* per block per inner iteration. Hence, the number of operations required (excluding the convergence test) for the four-point EDG inner iterative scheme is

$$\begin{gathered} 11.5\mathrm{m}^2 \text{ operations per inner iteration} \\ + \\ 7.5\mathrm{m}^2 \text{ operations after convergence.} \end{gathered} \qquad [5.13]$$

This estimate also holds for the inner iteration on u.

TABLE 3: Iterative count and CPU measure of the 4-pt EDG outer-inner scheme

n	Re	w	Ave-Abs Error for u	Ave-Abs Error for v	Number of outer iterates	Number of inner iter. for v	Number of inner iter. for u	Time (secs)
25	10	1.68	1.78E-07	1.09E-07	1 2 3 4	58 47 25 1	41 34 15 1	24.98
	100	1.68	1.78E-08	1.09E-08	1 2 3 4	52 42 19 1	35 29 7 1	21.12
	1000	1.68	1.78E-09	1.09E-09	1 2 3 4	44 34 14 1	49 25 1 1	17.94
37	10	1.77	8.34E-08	5.14E-08	1 2 3 4	85 47 35 1	89 50 19 1	78.95
	100	1.77	8.33E-09	5.15E-09	1 2 3 4	74 59 27 1	80 43 8 1	66.73
	1000	1.77	8.28E-10	5.20E-10	1 2 3 4	47 45 19 1	71 34 1 1	54.84
49	10	1.82	4.82E-08	2.98E-08	1 2 3 4	112 87 45 1	114 47 24 1	181.75
	100	1.82	4.80E-09	2.99E-09	1 2 3 4	101 72 34 1	105 54 8 1	151.47
	1000	1.82	4.72E-10	3.09E-10	1 2 3 4	88 62 23 1	93 47 1 1	124.48
61	10	1.84	3.13E-08	1.94E-08	1 2 3 4	148 109 44 1	153 82 27 1	349.80
	100	1.84	3.11E-09	1.94E-09	1 2 3 4	133 94 38 1	138 68 1 1	293.19
	1000	1.84	3.17E-10	2.04E-10	1 2 3 4	117 81 27 1	123 48 1 1	250.13

6 Discussion and Conclusion

In this paper, two types of iterative schemes based on the *rotated* finite difference discretisation were presented in solving the two dimensional steady Burgers' equation. By combining the results for the experimental number of iterations (inner and outer) shown in TABLES 1- 3 with the number of operations required in each inner iteration by each method, we can get the total number of arithmetic operations for the three methods in order to get the solution of the problem. The estimations are tabulated in TABLE 4.

TABLE 4: Arithmetic operations estimates for the *centred* difference, *rotated* point and 4-pt EDG schemes

n	Re	*Centred* Difference Scheme	*Rotated* Point Scheme	4-pt EDG Scheme
25	10	$7704m^2$	$2986m^2$	$2843m^2$
	100	$6624m^2$	$2530m^2$	$2429m^2$
	1000	$5490m^2$	$2197.5m^2$	$2026.5m^2$
37	10	$11142m^2$	$4326m^2$	$4051m^2$
	100	$9630m^2$	$3613m^2$	$3452.5m^2$
	1000	$8442m^2$	$3100m^2$	$2831.5m^2$
49	10	$14184m^2$	$5627m^2$	$5269.5m^2$
	100	$12564m^2$	$4629.5m^2$	$4407m^2$
	1000	$10224m^2$	$3993m^2$	$3694m^2$
61	10	$18090m^2$	$6919m^2$	$6580.5m^2$
	100	$15516m^2$	$5731.5m^2$	$5511m^2$
	1000	$13176m^2$	$4914.5m^2$	$4648.5m^2$

From the results obtained, it is apparent that both *rotated* schemes are faster than the scheme based on the *centred* difference formula since their iterations involve only half of the total nodal points in the solution domain. In terms of accuracy, both *rotated* schemes are relatively as good as the latter method since they are of second order accuracies. It can also be observed that as Re gets larger, the convergence gets to be faster with the number of iterations in each inner iteration process tends to decrease. One possible explaination for this could be that as Re gets larger, the matrices resulted from [2.7] and [2.8], which contain variable elements u_{ij} and vij, may have become more diagonally dominant with the computed u_{ij} and v_{ij} in the inner iteration processes.

The best results were obtained when the model problem was solved using the four-point EDG inner iterative scheme. From TABLE 4, clearly it can be observed that the four-point EDG method requires the *least* amount of computational work and the total computational operations in the *rotated* point scheme is slightly higher than the four-point EDG which coincides with the pattern of timing results obtained in our experiments. In conclusion, the new iterative schemes serve as viable alternatives in solving the two dimensional Burger's equation.

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 & A.R. Abdullah, *A New Fast Navier-Stokes Solver and Its Parallel Implementation,* Malaysian Journal of Computer Science, Vol. 10(2)(1997) pp. 51-59.

[3] 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.

[4] G. Dahlquist and A. Bjorck, *Numerical Methods,* Prentice-Hall, Englewood Cliffs, N.Jersey, 1974.

[5] C.A.J. Fletcher, *Computational Techniques for Fluid Dynamics 1,* Springer-Verlag, Berlin, 1991.

[6] G.A. Gravvanis, M.P. Bekakos and O.B. Efremides, *Parallel Implicit Preconditioned Conjugate Gradient Methods for Solving Biharmonic Equations,* to appear in the Proceeding of the 6th Hellenic European Conference On Computer Mathematics and its Applications (HERCMA 2003), Athens, Greece, Sep. 2003, 25-27

[7] D. Greenspan, *Numerical Studies of Steady, Viscous, Incompressible Flow in a Channel with a Step,* Journal of Engineering Mathematics, Vol. 3 No. 1(1969), Wolters-Noordhoff Publishing-Groningen, Netherlands

[8] Y. Saad, *Iterative Methods for Sparse Linear Systems,* PWS, New York, 1996.

[9] M.S. Sahimi, & D.J. Evans, *The AGE Solution of the Biharmonic Equation for the Deflection of A Uniformly-loaded Square Plate,* Report 748(1994), Department of CS, Loughborough University of Technology.

[10] M.S. Sahimi & 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.

[11] J. Smith, *The Coupled Equation Approach to the Numerical Solution of the Biharmonic Equation by Finite Differences,* SIAM Journal Numerical Analysis, Vol. 5(2)(1968), pp. 323-339.

[12] R. Vichnevetsky, *Computer Methods for Partial Differential Equations, Volume 1,* Prentice-Hall, Englewood Cliffs, N Jersey, 1981.