New Explicit Group Newton’s Iterative Methods for the Solutions of Burger’s Equation

Tan K. B. and Norhashidah Hj. M. Ali

Abstract—In this article, we aim to discuss the formulation of two explicit group iterative finite difference methods for time-dependent two dimensional Burger’s problem on a variable mesh. For the non-linear problems, the discretization leads to a non-linear system whose Jacobian is a tridiagonal matrix. We discuss the Newton’s explicit group iterative methods for a general Burger’s equation. The proposed explicit group methods are derived from the standard point and rotated point Crank-Nicolson finite difference schemes. Their computational complexity analysis is discussed. Numerical results are given to justify the feasibility of these two proposed iterative methods.

Keywords—Standard point Crank-Nicolson (CN), Rotated point Crank-Nicolson (RCN), Explicit Group (EG), Explicit Decoupled Group (EDG).

I. INTRODUCTION

This paper is an extension of our previous work in [1] on the explicit group methods for time-dependent two dimensional convection-diffusion equations. The explicit group (EG) iterative scheme was first introduced by Yousif and Evans [2] in solving the two dimensional elliptic problems by grouping the mesh points into smaller size groups of points. In 1991, Abdullah [3] developed the explicit decoupled group (EDG) iterative scheme by using the same ideas of explicit group on rotated grid. These explicit group methods have been extensively investigated over the years in solving various types of equations [4] – [7]. These methods are easy to implement and require lesser computational cost and they are suitable to be used on parallel computers. In [1] we have applied the EG and EDG iterative schemes to solve the two dimensional convection-diffusion problems. We also discussed the stability and consistency of EG and EDG schemes.

Burger’s equation is an important partial differential equation in fluid mechanics. It has been widely used for various applications, such as mathematical model of turbulence [8], shock wave [9] and traffic flow. There are many numerical methods for solving Burger’s equation, such as finite difference scheme, the Eulerian-Lagrangian method, variational iteration method, the lattice Boltzmann method and so on. In this paper, we introduce the explicit group Newton’s iterative methods for the solution of the time-dependent two dimensional Burger’s equation. In the next section, we will describe the formulation of these explicit group methods followed by the computational complexity analysis in Section III. The Numerical experiments and results are presented in Section IV. The concluding remark is given in Section V.

II. THE EXPLICIT GROUP NEWTON’S ITERATIVE METHODS

A. Problem statement Review Stage

In this paper, we introduce the explicit group Newton’s iterative methods for the Burger’s equation based on finite difference approximation. We consider the time-dependent two-dimensional Burger’s equation as follows:

\[
\frac{\partial u}{\partial t} + u \left( \frac{\partial u}{\partial x} + \frac{\partial u}{\partial y} \right) = \frac{1}{\text{Re}} \left( \frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} \right)
\]

subject to the initial conditions

\[
u(x,y,0) = u_0(x,y), \quad (x,y) \in D.
\]

The boundary conditions

\[
u(x,y,t) = f(x,y,t), (x,y,t) \in \partial D \times (0,T)
\]

where \(D = [a,b] \times [c,d]\) is a rectangular domain and \(\partial D\) is its boundary.

We define that

\[F = 0.5u^2.\]

Then, Equation (1) is equivalent to

\[
\frac{\partial u}{\partial t} + \left( \frac{\partial F}{\partial x} + \frac{\partial F}{\partial y} \right) = \frac{1}{\text{Re}} \left( \frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} \right)
\]

or

\[
\frac{\partial u}{\partial t} + 0.5 \left( \frac{\partial u}{\partial x} + \frac{\partial u}{\partial y} \right) = \frac{1}{\text{Re}} \left( \frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} \right)
\]

B. Standard Point Crank-Nicolson (CN) Newton’s Iterative Method

The common Crank-Nicolson finite difference formula in discretizing (6) is:
This equation is equivalent to

\[
(1 + \Delta x + \Delta y) u_{i,j}^{k+1} - \frac{\Delta x}{2} (u_{i+1,j}^{k+1} + u_{i,j}^{k+1}) - \frac{\Delta y}{2} (u_{i,j+1}^{k+1} + u_{i,j}^{k+1}) + \frac{\Delta x}{8} \left[ (u_{i+1,j}^{k+1})^2 - (u_{i,j}^{k+1})^2 \right] + \frac{\Delta y}{8} \left[ (u_{i,j+1}^{k+1})^2 - (u_{i,j}^{k+1})^2 \right] = (1 - \Delta x - \Delta y) u_{i,j}^k + \frac{\Delta x}{2} (u_{i+1,j}^k + u_{i,j}^k) + \frac{\Delta y}{2} (u_{i,j+1}^k + u_{i,j}^k) - \frac{\Delta x}{8} \left[ (u_{i+1,j}^k)^2 - (u_{i,j}^k)^2 \right] - \frac{\Delta y}{8} \left[ (u_{i,j+1}^k)^2 - (u_{i,j}^k)^2 \right]
\]  

(8)

where

\[
S_x = \frac{1}{\text{Re} \Delta x}, \quad S_y = \frac{1}{\text{Re} \Delta y}, \quad C_x = \frac{\Delta t}{\Delta x} \quad \text{and} \quad C_y = \frac{\Delta t}{\Delta y}.
\]

Using the classic Newton’s iteration method, equation (8) can be written as the root finding problem for the function of \( g \) as

\[
g_{i,j}^k = (1 + \Delta x + \Delta y) u_{i,j}^{k+1} - \frac{\Delta x}{2} (u_{i+1,j}^{k+1} + u_{i,j}^{k+1}) - \frac{\Delta y}{2} (u_{i,j+1}^{k+1} + u_{i,j}^{k+1}) + \frac{\Delta x}{8} \left[ (u_{i+1,j}^{k+1})^2 - (u_{i,j}^{k+1})^2 \right] + \frac{\Delta y}{8} \left[ (u_{i,j+1}^{k+1})^2 - (u_{i,j}^{k+1})^2 \right]
\]

(9)

The Newton’s iteration [10] at each time step, is given by

\[
\left( \frac{\partial g_{i,j}^k}{\partial u_{i,j}^{k+1}} \right) (w_{i,j}) = -g_{i,j}^k
\]  

(10)

where

\[
w_{i,j} = u_{i,j}^{k+1} - u_{i,j}^k, \quad p = 0,1,2,\
\]

and \( \left( \frac{\partial g_{i,j}^k}{\partial u_{i,j}^{k+1}} \right) \) = Jacobian matrix, evaluated using term by term discretization of (9). The standard point Crank-Nicolson Newton’s iterative formula can be shown to be the following form:

\[
(1 + \Delta x + \Delta y) w_{i,j} = \left( \frac{\Delta x}{2} C_x u_{i,j}^{k+1} \right) w_{i,j} - \left( \frac{\Delta x}{2} C_x u_{i,j}^{k+1} \right) w_{i,j} \]

(11)

The computational molecule of CN is shown in Fig. 1 (a). The Newton’s Iterative Algorithm for standard point Crank-Nicolson (CN) method at each time step is:

1. Do until convergence
2. compute \( g \) function
3. Do until convergence
4. solve for \( w \) in system \( Aw=b \)
5. End Do
6. \( w^{k+1} = w + u^k \)
7. End Do

C. Rotated Point Crank-Nicolson (RCN) Newton’s Iterative Method

Using rotated point finite difference approximation for (6), the difference equation becomes

\[
u_{i,j}^{k+1} + \frac{\Delta t}{2h^2} \left[ u_{i+1,j+1}^{k+1} + u_{i-1,j+1}^{k+1} - 4u_{i,j}^{k+1} \right]
\]

(12)

\[
+ \frac{\Delta t}{2h^2} \left[ u_{i+1,j-1}^{k+1} + u_{i-1,j-1}^{k+1} - 4u_{i,j}^{k+1} \right]
\]

The computational molecule of (12) and (16) is shown in Fig. 1 (b).

\[
(w^{k+1} + u^k)
\]

Fig. 1 (a) Computational Molecule of (12); (b) Computational Molecule of (16)


International Scholarly and Scientific Research & Innovation 5(12) 2011 1824 scholar.waset.org/1999.7/6670
approximation is:
\[
g_{ij}^{p+1} = \left(1 + S\right)u_{ij}^{p+1} - S\left(u_{ij}^{p+1} + u_{i+1,j}^{p+1} + u_{i,j+1}^{p+1} + u_{i-1,j+1}^{p+1}\right)
+ \frac{C}{8}\left(u_{i+1,j+1}^{p+1} - u_{i,j+1}^{p+1} - u_{i+1,j}^{p+1} + u_{i,j}^{p+1}\right) - \frac{(1 - S)u_{ij}^p + S(u_{i+1,j+1}^p + u_{i+1,j}^p + u_{i,j+1}^p + u_{i,j}^p)}{4} - \frac{C}{8}\left(u_{i+1,j+1}^p - u_{i,j+1}^p - u_{i+1,j}^p + u_{i,j}^p\right)
\]
\[\text{(15)}\]

The RCN Newton’s iterative equation is as below
\[
\left(1 + S\right)w_{ij} - \left(\frac{S}{4} - \frac{C}{4}u_{ij}^{p+1}\right)w_{ij} - \left(\frac{S}{4} + \frac{C}{4}u_{ij}^{p+1}\right)w_{ij} = -g_{ij}^p
\]
\[\text{(16)}\]

The computational molecule of RCN is shown in Fig. 1 (b). Fig. 2 represents the mesh points on solution domain of RCN Newton’s iterative method for the case n = 9. We can see that the evaluations at the points • involve only the same type of points; similarly for the points of type O. Thus, the iterations can be generated involving one type of points only. For the numerical experiment in IV, we chose the points of type • to be involved in the iterations. After a certain convergence criteria is achieved, the solutions at the points of O will be evaluated directly using (9) and (12).

The Newton’s Iterative Algorithm of rotated point Crank-Nicolson (RCN) scheme at each time step is as follows:
1. Do until convergence
2. compute \(g\) function at the points (•) only
3. Do until convergence
4. solve for \(w\) in system \(Aw=b\) at the points (•) only
5. End Do
6. compute \(u^{p+1} = w + u^p\) at the points (•) only
7. End Do
8. compute \(g\) function using CN at the points (O)
9. compute \(w\) at points (O)
10. compute \(u^{p+1}\) at points (O)

D. Explicit Group (EG) Newton’s Iterative Method
To formulate the EG Newton’s iterative scheme for the Burger’s equation, we apply (12) to any group of four points on the solution domain in the inner iteration at each time step. This will result in a (4x4) system of equations as follows:
\[
\begin{align*}
\begin{bmatrix}
a_1 & a_2 & 0 & a_3 \\
a_4 & a_1 & a_5 & 0 \\
0 & a_6 & a_1 & a_7 \\
a_8 & 0 & a_6 & a_1
\end{bmatrix}
\begin{bmatrix}
w_{1,j} \\
w_{1+1,j} \\
w_{j+1,j} \\
w_{j+1,j+1}
\end{bmatrix}
&=
\begin{bmatrix}
RHS_{1,j} \\
RHS_{1+1,j} \\
RHS_{j+1,j} \\
RHS_{j+1,j+1}
\end{bmatrix}
\end{align*}
\]
\[\text{(17)}\]

where
\[
\begin{align*}
a_1 &= Sx + Sy, \\
a_2 &= -\left(Sx + Cy\right)u_{ij}^{p+1}, \\
a_3 &= -\left(Sy + Cx\right)u_{ij}^{p+1}, \\
a_4 &= -\left(Sx + Cy\right)u_{ij}^{p+1}, \\
a_5 &= -\left(Sy + Cx\right)u_{ij}^{p+1}, \\
a_6 &= -\left(Sx + Cy\right)u_{ij}^{p+1}, \\
a_7 &= -\left(Sy + Cx\right)u_{ij}^{p+1}, \\
a_8 &= -\left(Sx + Cy\right)u_{ij}^{p+1}
\end{align*}
\]
\[\text{(18)}\]

and
\[
\begin{align*}
RHS_{1,j} &= \left(Sx + Cy\right)u_{ij}^{p+1}, \\
RHS_{1+1,j} &= \left(Sy + Cx\right)u_{ij}^{p+1}, \\
RHS_{j+1,j} &= \left(Sx + Cy\right)u_{ij}^{p+1}, \\
RHS_{j+1,j+1} &= \left(Sy + Cx\right)u_{ij}^{p+1}
\end{align*}
\]
\[\text{(19)}\]

The computational molecule of EG is shown in Fig. 3. The explicit form of (17) is obtained as
\[
\begin{align*}
\begin{bmatrix}
w_{1,j} \\
w_{1+1,j} \\
w_{j+1,j} \\
w_{j+1,j+1}
\end{bmatrix}
&=
\begin{bmatrix}
b_1 & b_2 & b_3 & b_4 \\
b_5 & b_6 & b_7 & b_8 \\
b_9 & b_{10} & b_{11} & b_{12} \\
b_{13} & b_{14} & b_{15} & b_{16}
\end{bmatrix}
\begin{bmatrix}
RHS_{1,j} \\
RHS_{1+1,j} \\
RHS_{j+1,j} \\
RHS_{j+1,j+1}
\end{bmatrix}
\end{align*}
\]
\[\text{(20)}\]

where
\[
\begin{align*}
b_1 &= a_1 \left(a_1^2 + a_4 a_5 - a_5 a_1\right), \\
b_2 &= a_1 \left(a_4 a_5 + a_9\right), \\
b_3 &= a_1 \left(a_1^2 + a_4 a_5 - a_5 a_1\right), \\
b_4 &= a_1 \left(a_5^2 + a_5 a_1 - a_1 a_5\right), \\
b_5 &= -a_1 \left(a_1^2 - a_5 a_1\right) - a_6 a_1 a_5, \\
b_6 &= a_1 \left(a_5^2 + a_5 a_1 - a_1 a_5\right), \\
b_7 &= -a_1 \left(a_1^2 - a_5 a_1\right) - a_6 a_1 a_5, \\
b_8 &= a_1 \left(a_5^2 + a_5 a_1 - a_1 a_5\right), \\
b_9 &= a_1 \left(a_1^2 - a_5 a_1\right) - a_6 a_1 a_5, \\
b_{10} &= a_1 \left(a_5^2 + a_5 a_1 - a_1 a_5\right), \\
b_{11} &= a_1 \left(a_1^2 - a_5 a_1\right) - a_6 a_1 a_5, \\
b_{12} &= a_1 \left(a_5^2 + a_5 a_1 - a_1 a_5\right), \\
b_{13} &= a_1 \left(a_1^2 - a_5 a_1\right) - a_6 a_1 a_5, \\
b_{14} &= a_1 \left(a_5^2 + a_5 a_1 - a_1 a_5\right), \\
b_{15} &= a_1 \left(a_1^2 - a_5 a_1\right) - a_6 a_1 a_5, \\
b_{16} &= a_1 \left(a_5^2 + a_5 a_1 - a_1 a_5\right), \\
\text{const} &= a_1^2 \left(a_1^2 - a_5 a_1\right) - a_6 a_1 a_5, \\
+ a_1 a_5 a_1 a_6 - a_1 a_4 a_5 a_6 + a_1 a_5 a_1 a_6
\end{align*}
\]
\[\text{(21)}\]
Method group of four points on the solution domain in the inner (4x4) system of equation:

\[
\begin{bmatrix}
  a_1 & -a_2 & 0 & 0 \\
  -a_3 & a_4 & 0 & 0 \\
  0 & 0 & a_5 & -a_6 \\
  0 & 0 & -a_7 & a_8 \\
\end{bmatrix}
\begin{bmatrix}
  w_{1,j} \\
  w_{i+1,j} \\
  w_{i,j+1} \\
  w_{i+1,j+1} \\
\end{bmatrix} =
\begin{bmatrix}
  \text{RHS}_{i,j} \\
  \text{RHS}_{i+1,j} \\
  \text{RHS}_{i,j+1} \\
  \text{RHS}_{i+1,j+1} \\
\end{bmatrix}
\]  

where

\[
a_i = 1 + S, a_2 = \frac{S}{4} C_{i,j+1}, a_3 = \frac{S}{4} C_{i+1,j}, a_4 = -\frac{S}{4}
\]  

and

\[
\begin{align*}
\text{RHS}_{i,j} &= \frac{S}{4} C_{i,j+1} w_{i+1,j} + \frac{S}{4} (w_{i+1,j} + w_{i+1,j+1}) - E_{i,j} \\
\text{RHS}_{i+1,j} &= \frac{S}{4} C_{i+1,j} w_{i,j+1} + \frac{S}{4} (w_{i,j+1} + w_{i+1,j+1}) - E_{i+1,j} \\
\text{RHS}_{i,j+1} &= \frac{S}{4} C_{i,j+1} w_{i+1,j} + \frac{S}{4} (w_{i+1,j} + w_{i,j+1}) - E_{i,j+1} \\
\text{RHS}_{i+1,j+1} &= \frac{S}{4} C_{i+1,j} w_{i,j+1} + \frac{S}{4} (w_{i,j+1} + w_{i+1,j+1}) - E_{i+1,j+1}
\end{align*}
\]

This system leads to two explicit decoupled systems of 2x2 equations

\[
\begin{bmatrix}
  w_{j,i} \\
  w_{i+1,j+1} \\
\end{bmatrix} =
\begin{bmatrix}
  a_1 & a_2 \\
  -a_3 & a_4 \\
\end{bmatrix}
\begin{bmatrix}
  w_{1,j} \\
  w_{i+1,j} \\
\end{bmatrix} +
\begin{bmatrix}
  \text{RHS}_{1,j} \\
  \text{RHS}_{i+1,j} \\
\end{bmatrix}
\]  

and

\[
\begin{bmatrix}
  w_{i+1,j} \\
  w_{i,j+1} \\
\end{bmatrix} =
\begin{bmatrix}
  a_5 & a_6 \\
  -a_7 & a_8 \\
\end{bmatrix}
\begin{bmatrix}
  w_{1,j} \\
  w_{i+1,j} \\
\end{bmatrix} +
\begin{bmatrix}
  \text{RHS}_{i,j} \\
  \text{RHS}_{i,j+1} \\
\end{bmatrix}
\]

E. Explicit Decoupled Group (EDG) Newton’s Iterative Method

For EDG Newton’s iterative scheme, we apply (17) to any group of four points on the solution domain in the inner iteration at each time step. Similarly, this will also result in a type 1 only, while (26) involves points of type 2. The iterative evaluation of (25) involves points of type 1, while (26) involves points of type 2. Therefore, the implementation of (25) and (26) can be carried out independently. We can save the execution time if the iteration over the solution domain is only carried out on either type of points (● or ■). Suppose we choose to iterate on points of type ●. The EDG Newton’s iterative scheme corresponds to iterations on these points using the group formula (25) until convergence test is satisfied. After a convergence criterion is achieved, the solutions at the remaining points of type ■ are evaluated directly once using the formulas of CN (9) and (12).

![Fig. 3 Computational Molecule of (17)](image)

![Fig. 4 (a) Computational Molecule of (25); (b) Computational Molecule of (26)](image)

III. COMPUTATIONAL COMPLEXITY ANALYSIS

In this section, we analyze the computational complexity of the four Newton’s iterative methods. The estimation is based on the arithmetic operations performed at every iteration. Assume that the solution domain is discretized with mesh size \( n \), then the number of internal mesh points is given by \( m^2 \) where \( m = n - 1 \). There are two types of internal mesh points, which are the iterative and direct points. Iterative points are the points which are involved in the iteration process, while direct points are the points that are computed directly once after the iteration convergence criteria is achieved. Table 1 shows the number of arithmetic operations required for every iteration of each method (excluding the convergence test and direct solutions).

To further study the complexity analysis of the explicit group methods, please refer to [2] – [3].

<table>
<thead>
<tr>
<th>Methods</th>
<th>Additional Multiplication</th>
</tr>
</thead>
<tbody>
<tr>
<td>Standard Point CN</td>
<td>9 m²</td>
</tr>
<tr>
<td>Rotated Point CN</td>
<td>3 m²</td>
</tr>
<tr>
<td>EG</td>
<td>21.75 m²</td>
</tr>
<tr>
<td>EDG</td>
<td>4 m²</td>
</tr>
</tbody>
</table>

IV. NUMERICAL EXPERIMENTS AND DISCUSSION OF RESULTS

In order to compare the performances of these Newton’s iterative methods described in Section II, we carried out the numerical experiments on a PC with Core 2 Duo 2.8 GHz, 2 GB of RAM with Window XP SP3 operating system using Cygwin C. The convergence criteria used was the maximum absolute error test with the error tolerance equal to \( \varepsilon = 10^{-10} \).

We solve the following two-dimensional Burger’s equation

\[
\frac{\partial u}{\partial t} + u \left( \frac{\partial u}{\partial x} + \frac{\partial u}{\partial y} \right) = \frac{1}{\text{Re}} \left( \frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} \right)
\]
where $D: [0,2] \times [0,2]$. The initial and boundary conditions are defined so that they satisfy the exact solution [11]:

$$u(x,y,t) = \frac{1}{1 + \exp(\text{Re}(x+y-t)/2)} \in D, t \geq 0$$ (28)

Tables II and III in APPENDIX show the numerical results of all four methods described in Section II when $1/\text{RE} = 1.0$. Amongst the point methods, the rotated point scheme (RCN) is faster than the traditional CN point scheme as the number of grid points increases due to its lower computational complexities. It can be observed that the accuracies of EG and EDG are as good as the CN and RCN respectively, but they require less execution times to achieve the results. The computing timing of EG is only 56 – 84 % of CN. The computing timing of EDG is only 62 – 91 % of RCN. The results of these four methods when $1/\text{RE} = 0.1$ are shown in Tables 4 and 5 in APPENDIX. The EDG scheme is the fastest method. Figs. 5 and 6 display the execution times of these four methods when $1/\text{RE}=1.0$ and 0.1 respectively.

![Fig. 5 Execution Times of standard point CN, rotated point CN, EG and EDG Newton’s iterative methods when $t = 2, 1/\text{RE} = 1, \Delta t=1/1000$](image)

![Fig. 6 Execution Times of standard point CN, rotated point CN, EG and EDG Newton’s iterative methods when $t = 2, 1/\text{RE} = 0.10, \Delta t=1/1000$](image)

The experimental results of these methods with various number of $1/\text{RE}$ (0.1, 0.01 and 0.005) are shown in Table 6 on APPENDIX. When $1/\text{RE}$ is decreased, the shock of the solution is sharper and steeper. Our Newton’s iterative methods are still accurate and non-oscillatory sinc the maximum error increases only at a small part localized to the middle part of the domain along the line $x+y=2$.

![Fig. 7 Surface plot of the difference between the numerical results of EG and exact solutions at $t = 2, 1/\text{RE} = 0.005, \Delta t = 0.005, \Delta x = 2/401$](image)

![Fig. 8 Surface plot of the difference between the numerical results of EDG and exact solutions at $t = 2, 1/\text{RE} = 0.005, \Delta t = 0.005, \Delta x = 2/401$](image)

### APPENDIX

#### TABLE II

<table>
<thead>
<tr>
<th>Space step</th>
<th>Maximum Error</th>
<th>Time (seconds)</th>
<th>Maximum Error</th>
<th>Time (seconds)</th>
</tr>
</thead>
<tbody>
<tr>
<td>2/21</td>
<td>0.0000001543</td>
<td>0.406</td>
<td>0.0000001329</td>
<td>0.217</td>
</tr>
<tr>
<td>2/41</td>
<td>0.0000001677</td>
<td>2.310</td>
<td>0.00000016577</td>
<td>6.198</td>
</tr>
<tr>
<td>2/81</td>
<td>0.000000153</td>
<td>17.777</td>
<td>0.00000016577</td>
<td>6.198</td>
</tr>
<tr>
<td>2/161</td>
<td>0.00000016577</td>
<td>198.231</td>
<td>0.00000016577</td>
<td>6.198</td>
</tr>
<tr>
<td>2/321</td>
<td>0.00000016577</td>
<td>3104.663</td>
<td>0.00000016577</td>
<td>6.198</td>
</tr>
</tbody>
</table>
The authors acknowledge the School of Mathematical Sciences KPI Incentive Grant Scheme (1002/PMATHS/ARSP13000) for the support of this research.

REFERENCES


