Numerical Solutions to Burgers’ Equations
(C.F.D.)

Utkarsh Bhardwaj
April 21, 2011
Numerical Solutions to Burgers’ Equations
(C.F.D.)

A report submitted by
Utkarsh Bhardwaj
(08DDCS215)

Under the guidance of
Mr. Vineet Kumar Srivastava

in partial fulfillment of the requirement of
B.tech programme

Faculty of Science and Technology ICFAI University, Dehradun.
Abstract

Burgers’ equation is a mathematical model to describe various kinds of phenomena such as turbulence and viscous fluid. In this work, numerical solutions to one-dimensional and two-dimensional Burgers’ equations are computed using fourth-order compact Crank-Nicolson scheme and second order Crank-Nicolson scheme, respectively and are analyzed by comparing with analytical solutions and previous available results. In case of one-dimensional Burgers equation, Hopf-Cole transformation is used to linearize burgers’ equation to heat equation which is then discretized using Crank-Nicolson scheme. The order of accuracy is improved to fourth order using Pade’ approximation and Richardson’s Extrapolation. Fourth-order Simpson’s rule is used for numerical integration to maintain overall accuracy. A direct method (TDMA) is employed for the solution of system of equations.

In case of two-dimensional coupled Burgers’ equations, Crank-Nicolson is used to discretize the set of equations, yielding a system of non-linear algebraic equations. Newton’s method is used to solve the system. A direct method has been developed for solving the sparsed linear set of equations occurring in Newton’s method. The efficiency analysis is then done based on the asymptotic analysis and running time of the program with different grid sizes.
Contents

1 Introduction 2

2 Mathematical Model 3
   2.1 One-dimensional Equation ........................................... 3
   2.2 Two-dimensional Equation ........................................... 3

3 Discretization Schemes 4
   3.1 Crank-Nicolson Scheme ............................................... 4
       3.1.1 One-Dimensional ............................................... 4
       3.1.2 Two-Dimensional ............................................... 6

4 Algorithms 7
   4.1 Newton’s Method .................................................... 7
   4.2 Gauss-Elimination with partial-pivoting .......................... 8
   4.3 Tridiagonal Matrix Algorithm ...................................... 8
   4.4 Algorithm for Sparse Matrix Solution ............................. 9
   4.5 Fourth Order Simpson’s Rule ...................................... 9

5 Solutions of Test Problems 10
   5.1 One-Dimensional Problems .......................................... 10
       5.1.1 Problem 1: .................................................... 10
       5.1.2 Problem 2: .................................................... 11
   5.2 Two-Dimensional Problems ......................................... 14
       5.2.1 Problem 1: .................................................... 14
       5.2.2 Problem 2: .................................................... 17

6 Analysis and Observations 20
   6.1 Time-complexity analysis ......................................... 20
   6.2 Accuracy Analysis .................................................. 22

Conclusion 24

References 25

Paper communicated based on this work 26
Acknowledgements

I wish to express my sincere thanks to my guide, Mr. Vineet Kumar Srivastava, Department of Mathematics, F.S.T. Icfai University for his precious encouragement and time throughout this work. I am also thankful to my family members, friends and Mansi for their love, motivation and support.

Utkarsh Bhardwaj
B.Tech(CSE)
ICFAI University
Dehradun
1 Introduction

Burgers’ equation is a special form of incompressible Navier-Stokes equation without having pressure term and continuity equation. Burgers’ equation is an important partial differential equation from fluid dynamics, and is widely used for various physical applications, such as modelling of gas dynamics and traffic flow, shock waves [5], investigating the shallow water waves[8], in examining the chemical reaction-diffusion model of Brusselator[6]. It is also used to test several numerical algorithms. The first attempt to solve Burgers’ equation analytically was given by Bateman [7], who derived the steady solution for a simple one-dimensional Burgers’ equation, which was used by J.M. Burger in to model turbulence[1]. For any initial condition, the one-dimensional Burgers’ equation can be reduced to a linear homogeneous heat equation that can be solved exactly, thus, the exact solution of the one-dimensional Burgers’ equation can be expressed in the form of a Fourier series. However, an efficient numerical method is still required, especially when the initial condition is not smooth or is only available on discrete locations. Another issue with the Fourier series solution is its slow convergence; therefore, a considerably long series is required to obtain an accurate approximation. In the past several years, numerical solution to one-dimensional Burgers’ equation and system of multidimensional Burgers’ equations have attracted a lot of attention, which has resulted in various finite-difference, finite-element and boundary element methods. To name a few, A. Refik Bahadır[4] proposed a fully implicit scheme and Arminjon P. Beauchamp C.[9] have developed a a finite-element method using rectangular elements. Finite difference methods can be classified in two broad categories, i.e. explicit and implicit. Explicit methods are efficient and easy to implement but are conditionally stable. Implicit methods are unconditionally stable but a system of equations needs to be solved at each time step of the solution. We use Crank-Nicolson finite difference scheme of second order and compact fourth order to solve one- and two-dimensional Burgers’ equation.
2 Mathematical Model

2.1 One-dimensional Equation

Governing Equation:

\[
\frac{\partial u}{\partial t} + u \frac{\partial u}{\partial x} - \frac{1}{Re} \left( \frac{\partial^2 u}{\partial x^2} \right) = 0
\]

Initial Condition:

\[u(x, 0) = \phi(x, 0)\]

Boundary Conditions:

\[u(a, t) = \psi(a, t)\]
\[u(b, t) = \psi(b, t)\]

where computational domain for \(x\) is \(a \leq x \leq b\) and for \(t\) is \(0 \leq t \leq t_{\text{max}}\).

2.2 Two-dimensional Equation

Governing Equation:

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

Initial Condition:

\[u(x, y, 0) = \phi(x, y, 0)\]
\[v(x, y, 0) = \phi(x, y, 0)\]

Boundary Conditions:

\[u(a, t) = \psi(a, t)\]
\[u(b, t) = \psi(b, t)\]
\[u(a, t)v(a, t) = \psi(a, t)\]
\[v(b, t) = \psi(b, t)\]

where computational domain is \(a \leq x \leq b, c \leq y \leq d\) and \(0 \leq t \leq t_{\text{max}}\).
3 Discretization Schemes

3.1 Crank-Nicolson Scheme

Crank-Nicolson scheme can be seen as an average of explicit and implicit discretization. This scheme is unconditionally stable. The solution of an unknown grid point depends on the neighbouring points of the same time step as well as of the previous time step, this is shown by the following figure. The scheme leads to a system of equations to be solved at each time step. The errors are quadratic over space and time.

3.1.1 One-Dimensional

For one-dimensional case, the method is based on the Hopf-Cole transformation which transforms the original nonlinear Burgers’ equation into a linear heat equation. The linear heat equation is then solved by a Crank-Nicolson fourth-order compact finite difference scheme. The transformed boundary conditions and initial conditions are also approximated with fourth-order accuracy to maintain an overall fourth-order accuracy. The 1-dimensional Burger’s equation can be transformed into a heat equation using the Hopf-Cole transformation[10][5]:

\[ u(x, t) = \frac{-2 w_x(x, t)}{Re \ w(x, t)} \]
where \( w(x,t) \) satisfies the following heat equation:

\[
\frac{\partial w}{\partial t} = \frac{1}{Re} \frac{\partial^2 w}{\partial t^2}, \quad 0 \leq x \leq 1, \quad 0 \leq t \leq T_{\text{max}}
\]

with initial condition:

\[
w(x,0) = \exp \left( - \int_0^x \frac{Re}{2} \phi(s) ds \right), \quad 0 \leq x \leq 1
\]

and boundary conditions:

\[
w(-1,t) = w(1,t); \quad w(n+1,t) = w(n-1,t); \quad w(0,t) = w(1,t) = 0
\]

where \( w_{-1} \) and \( w_{n+1} \) are two virtual grid points.

The heat equation produced is discretized by second-order Crank-Nicolson method. For further increase to fourth order of accuracy in spatial and temporal dimension Padé’ approximation and Richardson’s extrapolation is used, respectively. Padé’ approximation is generally used to accelerate Taylor expansion or even turn it from divergent to convergent by being re-arranged into a ratio of two such expansions. Richardson’s extrapolation is based on the idea of cancelling out present order error terms by the use of the solution at the present step and half of the step. This in turn increases the accuracy to next order term in the error, which in this case is of fourth order.

For initial condition calculation fourth order accuracy is maintained by fourth-order Simpson’s Rule.

The discretized fourth order accurate heat equations thus produced are: for \( w(x,t) \):

\[
\left( 1 + \frac{1}{12} \delta^2_x + \frac{\Delta t}{2Re\Delta x^2} \delta^2_x \right) w_{i+1}^{n+1} = \left( 1 + \frac{1}{12} \delta^2_x + \frac{\Delta t}{2Re\Delta x^2} \delta^2_x \right) w_i^n
\]

for \( w_x(x,t) \):

\[
\left( 1 + \frac{1}{6} \delta^2_x \right) w_x(x,t) = \frac{\delta^2_x}{2\Delta x} w(x,t)
\]

where \( \delta^2_x \) is the second order central finite difference operator along \( x \).

The discretized burgers’ equation is:

\[
\frac{u_i^{n+1} - u_i^n}{\Delta t} + \frac{1}{2} \left[ u_i^{n+1} \left( \frac{u_{i+1}^{n+1} - u_{i-1}^{n+1}}{2\Delta x} \right) + u_i^n \left( \frac{u_{i+1}^n - u_{i-1}^n}{2\Delta x} \right) \right] - \frac{1}{2Re} \left( \frac{u_{i+1}^{n+1} - 2u_i^{n+1} + u_{i-1}^{n+1}}{\Delta x^2} + \frac{u_i^{n+1} - 2u_i^n + u_{i-1}^n}{\Delta x^2} \right) = 0
\]

where \((i,j)\) is a grid point, \( \Delta x \) is grid size in \( x \) direction and \( \Delta t \) is time step.
3.1.2 Two-Dimensional

In non-linear coupled two-dimensional Burgers’ equation, backward-difference approximation for time and central-difference approximation for space is used with Crank-Nicolson Scheme. The grid geometry is shown in the following figure.

![2-D Grid(x-y plane at some time-step)](image_url)

Figure 3.3: 2-D Grid(x-y plane at some time-step)

The discretized non-linear coupled burgers’ equations are:

\[
\frac{u_{i,j}^{n+1} - u_{i,j}^n}{\Delta t} + \frac{1}{2} \left[ u_{i+1}^{n+1} \left( \frac{u_{i+1}^{n+1} - u_{i-1}^{n+1}}{2\Delta x} \right) + u_{i}^{n} \left( \frac{u_{i+1}^{n} - u_{i-1}^{n}}{2\Delta x} \right) \right] + \frac{1}{2} \left[ v_{i}^{n+1} \left( \frac{v_{i+1}^{n+1} - v_{i-1}^{n+1}}{2\Delta y} \right) + v_{i}^{n} \left( \frac{v_{i+1}^{n} - v_{i-1}^{n}}{2\Delta y} \right) \right] - \frac{1}{2} \frac{Re}{\Delta x^2} \left( u_{i+1,j}^{n+1} - 2u_{i,j}^{n+1} + u_{i-1,j}^{n+1} + u_{i,j+1}^{n} - 2u_{i,j}^{n} + u_{i,j-1}^{n} \right) = 0
\]

\[
\frac{v_{i,j}^{n+1} - v_{i,j}^n}{\Delta t} + \frac{1}{2} \left[ u_{i}^{n+1} \left( \frac{v_{i+1}^{n+1} - v_{i-1}^{n+1}}{2\Delta x} \right) + u_{i}^{n} \left( \frac{v_{i+1}^{n} - v_{i-1}^{n}}{2\Delta x} \right) \right] + \frac{1}{2} \left[ v_{i}^{n+1} \left( \frac{v_{i+1}^{n+1} - v_{i-1}^{n+1}}{2\Delta y} \right) + v_{i}^{n} \left( \frac{v_{i+1}^{n} - v_{i-1}^{n}}{2\Delta y} \right) \right] - \frac{1}{2} \frac{Re}{\Delta y^2} \left( v_{i+1,j}^{n+1} - 2v_{i,j}^{n+1} + v_{i-1,j}^{n+1} + v_{i,j+1}^{n} - 2v_{i,j}^{n} + v_{i,j-1}^{n} \right) = 0
\]

where \((i, j, n + 1)\) is a point in mesh grid, \(\Delta x\) is mesh size in x-direction, \(\Delta y\) is mesh size in y direction and \(\Delta t\) is time step.
4 Algorithms

The solutions of Burgers’ equation is computed by intervening various modules based on different algorithms. Some of the prominent algorithms used are cited and discussed in this section.

4.1 Newton’s Method

In two-dimensional case, the non-linear equations produced by discretization using Crank-Nicolson are solved using Newton’s Method.

Newton’s method for the given set of equations is stated stepwise:

1. set \( w_0 \) an initial guess.
2. while(not converge) do:
   - solve \( J(w_o) \times \Delta w = -R(\Delta w_o) \)
   - set \( w_{(k+1)} = w_o + \Delta w \).

Here, \( w_o \) is the initial guess or the previous iteration value, \( J \) is the Jacobian Matrix, \( R \) is the residual i.e. functional value of the discretized equations at \( w_o \). \( \Delta w \) is calculated at each step and added to the previous value of \( w \) to get new value of \( w \), then with this new value the convergence is checked and the steps are repeated until it has converge. Here, The convergence check is done using the r.m.s value at each iteration with proper tolerance limit. At each time step initial guess is retrieved from previous time step for all interior grid points and boundary values are updated from the given boundary value function.
4.2 Gauss-Elimination with partial-pivoting

The method used in solution for linear system of equations occuring in Newton’s method and in solution of heat equation is based on the Gauss-Elimination method. The method can be divided into two steps viz. forward elimination and back-substitution. To increase accuracy of the method and to avoid zero in the diagonal of the matrix, partial pivoting is done before the above two steps. Below are the steps listed in brief:

1. Form an augmented matrix $a$ of $n \times (n + 1)$ where $n$ is the no. of equations or unknowns.

2. Moving from top row to bottom compare the diagonal to other column entries and if the current diagonal entry is not greatest of all column entries, swap the current row and the row with greatest entry in the column.

3. Forward-Elimination: Moving from top to bottom, make the entries below the diagonal(lower triangular matrix) zero by performing subsequent row operations.

4. Back-Substitution: Moving from bottom row to top, the solution for $i^{th}$ row $x_i = a_{i,1}/b_i - (\sum_{j=i+1}^{n} x_j \times a_{i,j})$.

The method has time complexity of $O(n^3)$. The memory required for the set of equations is of order $n \times (n + 1)$ The linear system occurs in each timestep of fourth-order compact scheme twice and in each iteration of Newton’s method for every timestep in second-order compact scheme. This makes the use of this algorithm very inefficient.

4.3 Tridiagonal Matrix Algorithm

The solution using fourth-order compact scheme always produces a diagonally dominant tridiagonal matrix i.e. band matrix of bandwidth 1. So, tridiagonal matrix algorithm is used to account for efficiency. This algorithm is based on the Gauss-elimination algorithm. The steps in the algorithm are listed below:

1. Form an augmented matrix $a$ of $n \times 4$ (3 columns for coefficients and last one for constant).

2. Forward elimination: Moving from second row to bottom do following steps
   - $a_{i,1} = ai, 1/ai - 1, 2$.
   - $a_{i,2} = ai, 2/ai, 1 \times ai - 1, 3$.
   - $a_{i,4} = ai, 4/ai, 1 \times ai - 1, 4$.

3. Back-Substitution: Moving from bottom row to top, the solution for $i^{th}$ row $x_i = (a_{i,4} - a_{i,3} \times a_{i+1,4})/a_{i,2}$.

The time complexity of the algorithm is $O(n)$ and memory required is of order $n \times 4$. That makes it very efficient with respect to normal Gauss-Elimination.
4.4 Algorithm for Sparse Matrix Solution

The solution using second-order scheme for two-dimensional Burgers’ equation produces a sparse matrix. There are many sparse matrix storing and solving techniques available for example frontal solver. A variant of the Gauss-elimination is developed by observing the pattern of the sparse matrix. This sparse matrix is a jacobian matrix. The only non-zero entries on \( i^{th} \) row are those on which \( i^{th} \) mesh point depends upon. For two-dimensional problem the scheme produces six non-zero entries in one row and also in a column(for \( d \)-dimensional \( d \times 3 \) entries) which follow a particular pattern. The steps in the algorithm developed are listed below:

1. Form an augmented matrix \( a \) of \( n \times d \times 3 + 2 \) (\( d \times 3/2 \) columns for left \( d \times 3/2 \) columns for right of diagonal entries for coefficients and last one for constant).

2. Forward elimination: Follow the pattern to trace down the non-zero entry of the current column and apply subsequent row-operation to make the entry zero.

3. Back-Substitution: Moving from bottom row to top, find non-zero entries according to the pattern and apply back-substitution accordingly.

The time complexity of the algorithm is \( O(n) \) and memory required is of order \( n \times (d \times 3 + 2) \) or \( n \times (d \times 3 + 2) \). That makes it very efficient with respect to normal Gauss-Elimination for the sparse matrix and Other techniques like frontal solver and link-structured implementation.

4.5 Fourth Order Simpson’s Rule

The integrals for exact solutions and initial conditions for the heat equation in fourth-order compact scheme can not be expressed explictely for all test problems so to maintain overall fourth-order accuracy of the scheme, we use fourth order Simpson’s rule. The algorithm for numerical integration of \( f(x) \) over \((a, b)\) is as follows:

1. set m to some preferred value which is multiple of three and generally in the range of 12-18.

2. set \( h = (b - a)/(3m) \) and \( \text{res}=0 \).

3. for m iterations do:
   - \( \text{res} = \text{res} + f(h(3i - 3)) \)
   - \( \text{res} = \text{res} + 2f(h(3i - 2)) \)
   - \( \text{res} = \text{res} + 2f(h(3i - 1)) \)
   - \( \text{res} = \text{res} + f(h(3i)) \)

4. \( \text{res} = 3h \times \text{res}/8 \) is the result for numerical integration of \( f(x) \) over \((a, b)\).

The method is sufficiently accurate for small values of \( m \). The small value of \( m \) also makes it an efficient approach to use in the scheme.
5 Solutions of Test Problems

5.1 One-Dimensional Problems

To validate the fourth order Crank-Nicolson scheme two test problems are taken and the results computed are tabulated, plotted and compared to analytical solutions for typical mesh-points with different grid-sizes. Maximum error and run time for the solution set computed is also recorded for further analysis.

5.1.1 Problem 1:
Computational Domain:

\[ 0 \leq x \leq 1 \]
\[ 0 \leq t \leq t_{\text{max}} \]

Initial Condition:

\[ u(x, 0) = \frac{2}{Re} \left( \frac{\pi \sin(\pi x)}{2 + \cos(\pi x)} \right) \]

Boundary Conditions:

\[ u(0, t) = 0 \]
\[ u(1, t) = 0 \]

Exact Solution: Exact solution to compare numerical results

\[ u(x, t) = \frac{2\pi \exp(-\pi^2 t/Re) \sin(\pi x)}{2 + \exp(-\pi^2 t/Re) \cos(\pi x)} \]

step size: \[ \Delta t = 0.00001 \]
Tabulated Results and Observation

The table shows numerical results of fourth-order compact scheme against exact solutions. The numerical results obtained are in good agreement with the analytical solution.

Table 5.1: Problem 1 numerical solutions for respective grid size at t=1.0

<table>
<thead>
<tr>
<th>T.M.P.</th>
<th>$\Delta x = 1/20$</th>
<th>$\Delta x = 1/40$</th>
<th>Exact Solution</th>
</tr>
</thead>
<tbody>
<tr>
<td>(0.1)</td>
<td>0.03073532056</td>
<td>0.03073535698</td>
<td>0.03073535939</td>
</tr>
<tr>
<td>(0.2)</td>
<td>0.05980678869</td>
<td>0.05980685681</td>
<td>0.05980686132</td>
</tr>
<tr>
<td>(0.3)</td>
<td>0.08537566915</td>
<td>0.08537575991</td>
<td>0.08537576593</td>
</tr>
<tr>
<td>(0.4)</td>
<td>0.1052951509</td>
<td>0.1052952518</td>
<td>0.1052952585</td>
</tr>
<tr>
<td>(0.5)</td>
<td>0.1170895171</td>
<td>0.1170896144</td>
<td>0.1170896208</td>
</tr>
<tr>
<td>(0.6)</td>
<td>0.1181633843</td>
<td>0.1181634656</td>
<td>0.1181634709</td>
</tr>
<tr>
<td>(0.7)</td>
<td>0.1063798733</td>
<td>0.1063799310</td>
<td>0.1063799348</td>
</tr>
<tr>
<td>(0.8)</td>
<td>0.08104164267</td>
<td>0.08104167622</td>
<td>0.08104167843</td>
</tr>
<tr>
<td>(0.9)</td>
<td>0.04397681845</td>
<td>0.04397683275</td>
<td>0.04397683368</td>
</tr>
</tbody>
</table>

Max. Error 1.076747840e-007 6.715570683e-009
Run Time(s) 0.643 1.291

The maximum error for different grid size is also recorded and is reducing with refinement in grid. Further analysis on the basis of the results shown is done later in the accuracy analysis section.

5.1.2 Problem 2:
Computational Domain:

$$0 \leq x \leq 1$$
$$0 \leq t \leq t_{max}$$

Initial Condition:

$$u(x, 0) = \exp \left( \frac{1 - \cos(\pi x)}{2\pi/Re} \right)$$

Boundary Conditions:

$$u(0, t) = 0$$
$$u(1, t) = 0$$

Exact Solution: Exact solution to compare numerical results

$$u(x, t) = \frac{2\pi}{Re} \sum_{n=1}^{\infty} c_n \exp(-n^2\pi^2 t/Re) \sin(n\pi x)$$
\[ c_o = \int_0^1 \exp \left( -\frac{1 - \cos(\pi x)}{2\pi/Re} \right) dx \]

\[ c_n = 2 \int_0^1 \exp \left( -\frac{1 - \cos(\pi x)}{2\pi/Re} \right) \cos(n\pi x) dx, n = 1, 2, 3... \]

**Tabulated Results and Observation**

The table shows numerical results of fourth-order compact scheme against exact solutions.

<table>
<thead>
<tr>
<th>T.M.P. ( x )</th>
<th>Numerical Solution</th>
<th>Exact Solution</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \triangle x = 1/20 )</td>
<td>0.0754110180</td>
<td>0.0753819089</td>
</tr>
<tr>
<td>( \triangle x = 1/80 )</td>
<td>0.1506561788</td>
<td>0.1506451797</td>
</tr>
<tr>
<td>( \triangle x = 1/320 )</td>
<td>0.2256096930</td>
<td>0.2256658887</td>
</tr>
<tr>
<td>( \triangle t = 1/16 )</td>
<td>0.3002197939</td>
<td>0.3003088472</td>
</tr>
<tr>
<td>( \triangle t = 1/64 )</td>
<td>0.4487326429</td>
<td>0.4478160933</td>
</tr>
<tr>
<td>( \triangle t = 1/256 )</td>
<td>0.5231517985</td>
<td>0.5202683874</td>
</tr>
</tbody>
</table>

The table shows results for short time steps as well as for large time-step that shows the robustness of the scheme against other schemes such as explicit schemes that require very small time steps for stability issues. As the mesh is refined the values get closer to the analytical solution. Further analysis of accuracy and run time is done later in the analysis section.
Plots

The plots of the example show numerical and analytical solutions for the domain at different time which almost overlap. The curves also demonstrate the shocks produced in the solution of the Burgers’ Equation.

**Figure 5.1:** Numerical and Analytical solution of example for Re=100, $\Delta x = 0.0025$ and $\Delta t = 0.0001$
5.2 Two-Dimensional Problems

To verify the second-order Crank-Nicolson scheme for two-dimensional Burgers’ equation two test problems are taken. The results computed are listed, plotted and compared to analytical and other numerical results to account the behaviour of the method.

5.2.1 Problem 1:
Computational Domain:

\[
0 \leq x \leq 1 \\
0 \leq y \leq 1 \\
0 \leq t \leq t_{\text{max}}
\]

Initial Conditions:

\[
u(x, y, 0) = \frac{3}{4} - \frac{1}{4[1 + exp((-4x + 4y) \frac{Re}{32})]}
\]

\[
v(x, y, 0) = \frac{3}{4} + \frac{1}{4[1 + exp((-4x + 4y) \frac{Re}{32})]}
\]

Boundary Conditions:

\[
u(0, y, t) = \frac{3}{4} - \frac{1}{4[1 + exp((4y - t) \frac{Re}{32})]}
\]

\[
v(0, y, t) = \frac{3}{4} + \frac{1}{4[1 + exp((4y - t) \frac{Re}{32})]}
\]

\[
u(1, y, t) = \frac{3}{4} - \frac{1}{4[1 + exp((-4 + 4y - t) \frac{Re}{32})]}
\]

\[
v(1, y, t) = \frac{3}{4} + \frac{1}{4[1 + exp((-4 + 4y - t) \frac{Re}{32})]}
\]

\[
u(x, 0, t) = \frac{3}{4} - \frac{1}{4[1 + exp((-4x - t) \frac{Re}{32})]}
\]

\[
v(x, 0, t) = \frac{3}{4} + \frac{1}{4[1 + exp((-4x - t) \frac{Re}{32})]}
\]

\[
u(x, 1, t) = \frac{3}{4} - \frac{1}{4[1 + exp((-4x + 4 - t) \frac{Re}{32})]}
\]

\[
v(x, 1, t) = \frac{3}{4} + \frac{1}{4[1 + exp((-4x + 4 - t) \frac{Re}{32})]}
\]
Exact Solution: \[ u(x, y, t) = \frac{3}{4} \left( 1 - \frac{1}{4[1 + \exp((-4x + 4y - t)\frac{Re}{32})]} \right) \]

\[ v(x, y, t) = \frac{3}{4} + \frac{1}{4[1 + \exp((-4x + 4y - t)\frac{Re}{32})]} \]

step size: test set 1: \( \Delta x = 0.05 \)
\( \Delta t = 0.0001 \)
\( Re = 100 \)

**Tabulated Results and Observation**

The table shows numerical and analytical results of the problem with \( Re = 100 \) at some typical mesh points.

| Table 5.3: Problem:1 solutions for \( u \) at some typical mesh points at different final times |
|---|---|---|---|
| T.M.P. | time=0.01 | time=0.5 | time=2.0 |
| x | Numerical | Exact | Numerical | Exact | Numerical | Exact |
| (.1,.1) | 0.623106 | 0.623047 | 0.543002 | 0.543322 | 0.500470 | 0.500482 |
| (.5,.1) | 0.501617 | 0.501622 | 0.500341 | 0.500353 | 0.500003 | 0.500003 |
| (.9,.1) | 0.623106 | 0.623047 | 0.542692 | 0.543322 | 0.500003 | 0.500003 |
| (.3,.3) | 0.500011 | 0.500011 | 0.500002 | 0.500002 | 0.500000 | 0.500000 |
| (.7,.3) | 0.623106 | 0.623047 | 0.542692 | 0.543322 | 0.500003 | 0.500003 |
| (.1,.5) | 0.748272 | 0.748274 | 0.742150 | 0.742214 | 0.555153 | 0.555675 |
| (.5,.5) | 0.623106 | 0.623047 | 0.542509 | 0.543322 | 0.500414 | 0.500482 |
| (.9,.5) | 0.748272 | 0.748274 | 0.742114 | 0.742214 | 0.554816 | 0.555675 |
| (.3,.7) | 0.501617 | 0.501622 | 0.500317 | 0.500353 | 0.500003 | 0.500003 |
| (.7,.7) | 0.623106 | 0.623047 | 0.542692 | 0.543322 | 0.500003 | 0.500003 |
| (.1,.9) | 0.749988 | 0.749988 | 0.749945 | 0.749946 | 0.744196 | 0.744256 |
| (.5,.9) | 0.748272 | 0.748274 | 0.742103 | 0.742214 | 0.554504 | 0.555675 |
| (.9,.9) | 0.623106 | 0.623047 | 0.542282 | 0.543322 | 0.500525 | 0.500482 |
Table 5.4: Problem:1 solutions for v at some typical mesh points at different final times

<table>
<thead>
<tr>
<th>T.M.P. x,y</th>
<th>time=.01</th>
<th></th>
<th>time=0.5</th>
<th></th>
<th>time=2.0</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Numerical</td>
<td>Exact</td>
<td>Numerical</td>
<td>Exact</td>
<td>Numerical</td>
<td>Exact</td>
</tr>
<tr>
<td>(.1,.1)</td>
<td>0.876894</td>
<td>0.876953</td>
<td>0.956998</td>
<td>0.956678</td>
<td>0.999530</td>
<td>0.999518</td>
</tr>
<tr>
<td>(.5,.1)</td>
<td>0.998383</td>
<td>0.998378</td>
<td>0.999659</td>
<td>0.999647</td>
<td>0.99997</td>
<td>0.999997</td>
</tr>
<tr>
<td>(.9,.1)</td>
<td>0.999989</td>
<td>0.999989</td>
<td>0.999998</td>
<td>0.999998</td>
<td>1.000000</td>
<td>1.000000</td>
</tr>
<tr>
<td>(.3,.3)</td>
<td>0.876894</td>
<td>0.876953</td>
<td>0.957308</td>
<td>0.956678</td>
<td>0.999559</td>
<td>0.999518</td>
</tr>
<tr>
<td>(.7,.3)</td>
<td>0.998383</td>
<td>0.998378</td>
<td>0.999683</td>
<td>0.999647</td>
<td>0.999997</td>
<td>0.999997</td>
</tr>
<tr>
<td>(.1,.5)</td>
<td>0.751728</td>
<td>0.751726</td>
<td>0.757850</td>
<td>0.757786</td>
<td>0.944847</td>
<td>0.944325</td>
</tr>
<tr>
<td>(.5,.5)</td>
<td>0.876894</td>
<td>0.876953</td>
<td>0.957491</td>
<td>0.956678</td>
<td>0.999586</td>
<td>0.999518</td>
</tr>
<tr>
<td>(.9,.5)</td>
<td>0.998383</td>
<td>0.998378</td>
<td>0.999696</td>
<td>0.999647</td>
<td>0.999997</td>
<td>0.999997</td>
</tr>
<tr>
<td>(.3,.7)</td>
<td>0.751728</td>
<td>0.751726</td>
<td>0.757886</td>
<td>0.757786</td>
<td>0.945184</td>
<td>0.944325</td>
</tr>
<tr>
<td>(.7,.7)</td>
<td>0.876894</td>
<td>0.876953</td>
<td>0.957537</td>
<td>0.956678</td>
<td>0.999961</td>
<td>0.999518</td>
</tr>
<tr>
<td>(.1,.9)</td>
<td>0.750012</td>
<td>0.750012</td>
<td>0.750055</td>
<td>0.750054</td>
<td>0.755804</td>
<td>0.755744</td>
</tr>
<tr>
<td>(.5,.9)</td>
<td>0.751728</td>
<td>0.751726</td>
<td>0.757897</td>
<td>0.757786</td>
<td>0.945496</td>
<td>0.944325</td>
</tr>
<tr>
<td>(.9,.9)</td>
<td>0.876894</td>
<td>0.876953</td>
<td>0.957718</td>
<td>0.956678</td>
<td>0.999475</td>
<td>0.999518</td>
</tr>
</tbody>
</table>

Plots

The solution set of u and v is plotted as perspective surface graph.

![Perspective Surface Graph](image)

**Figure 5.2:** The plot numerical value of u for Re = 500 at time level t = 0.5
5.2.2 Problem 2:

Computational Domain:

\[ 0 \leq x \leq 0.5 \]
\[ 0 \leq y \leq 0.5 \]
\[ 0 \leq t \leq t_{\text{max}} \]

Initial Conditions:

\[ u(x, y, 0) = \sin(\pi x) + \cos(\pi y) \]
\[ v(x, y, 0) = x + y \]

Boundary Conditions:

\[ u(0, y, t) = \cos(\pi y) \]
\[ v(0, y, t) = y \]
\[ u(0.5, y, t) = 1 + \cos(\pi y) \]
\[ v(0.5, y, t) = 0.5 + y \]
\[ u(x, 0, t) = 1 + \sin(\pi x) \]
\[ v(x, 0, t) = x \]
\[ u(x, 0.5, t) = \sin(\pi x) \]
\[ v(x, 0.5, t) = x + 0.5 \]

Exact Solution: Not Available

step size: test set 1: \( \Delta x = 0.025 \)
\( \Delta t = 0.0001 \)

Re=50
**Tabulated Results and Observation**

The table shows numerical results of current scheme against other numerical solutions, the problem with $Re = 50$ at some typical mesh points.

**Table 5.5:** Problem:2 comparison of numerical solutions for $u$ at some typical mesh points at $t=0.625$

<table>
<thead>
<tr>
<th>T.M.P.</th>
<th>Present Scheme</th>
<th>A. R. Bhadir</th>
<th>Jain and Holla</th>
</tr>
</thead>
<tbody>
<tr>
<td>(0.1,0.1)</td>
<td>0.97146</td>
<td>0.96688</td>
<td>0.97258</td>
</tr>
<tr>
<td>(0.3,0.1)</td>
<td>1.15280</td>
<td>1.14827</td>
<td>1.16214</td>
</tr>
<tr>
<td>(0.2,0.2)</td>
<td>0.86307</td>
<td>0.85911</td>
<td>0.86281</td>
</tr>
<tr>
<td>(0.4,0.2)</td>
<td>0.97981</td>
<td>0.97637</td>
<td>0.96483</td>
</tr>
<tr>
<td>(0.1,0.3)</td>
<td>0.66316</td>
<td>0.66019</td>
<td>0.66318</td>
</tr>
<tr>
<td>(0.3,0.3)</td>
<td>0.77230</td>
<td>0.76932</td>
<td>0.77030</td>
</tr>
<tr>
<td>(0.2,0.4)</td>
<td>0.58180</td>
<td>0.57966</td>
<td>0.58070</td>
</tr>
<tr>
<td>(0.4,0.4)</td>
<td>0.75856</td>
<td>0.75678</td>
<td>0.74435</td>
</tr>
</tbody>
</table>

**Table 5.6:** Problem:2 comparison of numerical solutions for $u$ at some typical mesh points at $t=0.625$

<table>
<thead>
<tr>
<th>T.M.P.</th>
<th>Present Scheme</th>
<th>A. R. Bhadir</th>
<th>Jain and Holla</th>
</tr>
</thead>
<tbody>
<tr>
<td>(0.1,0.1)</td>
<td>0.09869</td>
<td>0.09824</td>
<td>0.09773</td>
</tr>
<tr>
<td>(0.3,0.1)</td>
<td>0.14158</td>
<td>0.14112</td>
<td>0.14039</td>
</tr>
<tr>
<td>(0.2,0.2)</td>
<td>0.16754</td>
<td>0.16681</td>
<td>0.16660</td>
</tr>
<tr>
<td>(0.4,0.2)</td>
<td>0.17110</td>
<td>0.17065</td>
<td>0.17397</td>
</tr>
<tr>
<td>(0.1,0.3)</td>
<td>0.26378</td>
<td>0.26261</td>
<td>0.26294</td>
</tr>
<tr>
<td>(0.3,0.3)</td>
<td>0.22654</td>
<td>0.22576</td>
<td>0.22463</td>
</tr>
<tr>
<td>(0.2,0.4)</td>
<td>0.32851</td>
<td>0.32745</td>
<td>0.32402</td>
</tr>
<tr>
<td>(0.4,0.4)</td>
<td>0.32500</td>
<td>0.32441</td>
<td>0.31822</td>
</tr>
</tbody>
</table>
Plots

The solution set of $u$ and $v$ is plotted as perspective surface graph. The two-dimensional burgers’ equations also show shock in the surface plot.

Figure 5.4: The plot for numerical value of $u$ for $Re = 50$ at time level $t = 0.625$

Figure 5.5: The plot for numerical value of $v$ for $Re = 50$ at time level $t = 0.625$
6 Analysis and Observations

The Methods employed are unconditionally stable and have good order of accuracy. The methods show different behaviour for different choice of grid-sizes and time-steps. Accuracy and efficiency are co-related by the fact that more refined the grid is more the accuracy but the running time and memory requirement will also increase. If the solution scheme is efficient than the refinement of grid will not change the running time by large factors and we can get accurate results in lesser time.

6.1 Time-complexity analysis

The Crank-Nicolson method gives a system of non-linear algebraic equations solved by Newton’s Method for each time step and is converted to a set of linear equations to be solved by direct method for each iteration of Newton’s Method. The direct method used can be Gauss Elimination. The matrix obtained in each iteration of Newton’s Method is highly sparsed so a modified version of Gauss Elimination can be used to improve time complexity from $O(n^3)$ to $O(n)$. The solutions obtained by the program varies from some seconds to hours of running time with small changes in grid size and time step. Some points about the growth rate of the method:

1. The solution for every interior grid point is found until it converges.
2. The converged solution is found for each time step.
3. For one iteration of solution set for the mesh, the time growth rate is $O(n)$ where $n$ is number of interior grid points.

So, the growth rate depends on number of time steps ($nt$), convergence ($cf$) and number of interior grid points or unknowns ($n$). A small value of time step also ensures a good initial guess for next time step and solution converges in very less number of iterations. For a good choice of time step which is not generally very large solution gives good accuracy and also converges fast. So, in most of the cases, the spatial grid is the bottleneck for the growth rate. $T(n, nt) = O(nt) \times O(n)$

Now, $n$, the no. of interior grid points, grows exponentially for d-dimensional problem as $n = d \times (nx-2)^d$, where $nx$ is the total number of grid points taken in any dimension, $(nx - 2)$ is the no. of grid points excluding two boundaries where the solution is known by boundary conditions.

For each iteration of solution, the growth rate can be stated as:

- for d-dimension $T(nx) = ((nx)^d)$
- for 1-dimension $T(nx) = (nx)$
• for 2-dimension $T(nx) = ((nx)^2)$
• for 3-dimension $T(nx) = ((nx)^3)$

This asymptotic analysis suggests that a small change of grid size can increase the running-time considerably for a 2-dimension and even more for 3-dimension problem but is of little significance for 1-dimension problem.

The fourth order accurate compact scheme gives a tridiagonal matrix to be solved by direct method at each time step and does not depend on dimensions involved. The direct method for a tridiagonal matrix is of $O(n)$ which is a modified form of Gauss elimination. The Richardson’s Extrapolation employed to increase order of accuracy in temporal dimension makes the system of the equation to be calculated twice for each time step which although doesn not contribute much to the time-complexity. so the overall complexity of the problem can be seen as $T(n, nt) = O(nt) \times O(n)$.

Now we can conclude that the time complexity of the two methods is comparable for one-dimensional problem but for multi-dimensional problems second method is more efficient.

The recorded running time values for single time step for one-dimensional problem using both the schemes show approximately same value with .002-.007sec of running time for nx value from 100-1000.
6.2 Accuracy Analysis

The accuracy of the methods depends on the order of accuracy of the scheme and grid-size chosen for the solution. More refined the grid is, lesser will be the truncation error and the results will be more close to the analytical values. The Crank-Nicolson scheme is second order accurate in time and space. The following table gives comparison for different values of grid size and time step for the second order Crank-Nicolson scheme for one-dimensional problem.

<table>
<thead>
<tr>
<th>T.M.P. x</th>
<th>Numerical Solutions</th>
<th>Exact Solution</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>nt=45 nx=20</td>
<td>nt=45 nx=100</td>
</tr>
<tr>
<td>(0.500000)</td>
<td>0.0074401</td>
<td>0.0074401</td>
</tr>
<tr>
<td>(0.600000)</td>
<td>0.0089561</td>
<td>0.0089561</td>
</tr>
<tr>
<td>(0.700000)</td>
<td>0.0104883</td>
<td>0.0104883</td>
</tr>
<tr>
<td>(0.800000)</td>
<td>0.0120402</td>
<td>0.0120402</td>
</tr>
<tr>
<td>(0.900000)</td>
<td>0.0136158</td>
<td>0.0136158</td>
</tr>
<tr>
<td>(1.000000)</td>
<td>0.0152199</td>
<td>0.0152198</td>
</tr>
<tr>
<td>(1.100000)</td>
<td>0.0168576</td>
<td>0.0168576</td>
</tr>
<tr>
<td>(1.200000)</td>
<td>0.0185354</td>
<td>0.0185353</td>
</tr>
<tr>
<td>(1.300000)</td>
<td>0.0202607</td>
<td>0.0202607</td>
</tr>
<tr>
<td>(1.400000)</td>
<td>0.0220427</td>
<td>0.0220427</td>
</tr>
<tr>
<td>(1.500000)</td>
<td>0.0238925</td>
<td>0.0238925</td>
</tr>
<tr>
<td>Total Absolute Error</td>
<td>4.46785e – 008</td>
<td>1.24365e – 008</td>
</tr>
</tbody>
</table>

Table 6.1: Comparing Accuracy of C-N scheme

The Compact Scheme is fourth order in both the dimensions viz. spatial and temporal. It utilises C-N scheme to solve linearized heat equations and then reducing them to the solution of Burgers’ equation. As already known, C-N scheme is second order in time and space, certain methods are employed to increase order of accuracy. In saptial dimension fourth order accuracy is attained by Pade’ approximation while in temporal dimension richardson’s extrapolation is used to account for fourth order accuracy. The overall fourth order accuracy is maintained by using fourth order accurate integration rule wherever necessary. The following tables verifies the statements made about the accuracy of the scheme.

<table>
<thead>
<tr>
<th>h</th>
<th>Max. Error=E(h)</th>
<th>1/10</th>
<th>1/20</th>
<th>1/40</th>
<th>1/80</th>
<th>1/160</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>r=E(h)/E(h/2)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>log2r</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

| r   | 1.76E-006 | 1.08E-007 | 6.72E-009 | 4.19E-010 | 2.62E-011 |
| log2r | 4.03238 | 4.00287 | 4.00211 | 3.99988 |

The table verifies that the order of accuracy for the compact scheme implemented is fourth order in spatial dimension.
Table 6.3: Order of accuracy with temporal dimension without Richardson’s extrapolation (One-dimensional: Problem 1)

<table>
<thead>
<tr>
<th></th>
<th>1/16</th>
<th>1/32</th>
<th>1/64</th>
</tr>
</thead>
<tbody>
<tr>
<td>( E(t) )</td>
<td>3.929842318e-005</td>
<td>9.821715264e-006</td>
<td>2.455248585e-006</td>
</tr>
<tr>
<td>( r = \frac{E(t)}{E(t/2)} )</td>
<td>-</td>
<td>4.001177200</td>
<td>4.000293625</td>
</tr>
<tr>
<td>( \log_2 r )</td>
<td>-</td>
<td>2.000424523</td>
<td>2.000105899</td>
</tr>
</tbody>
</table>

The table verifies that the order of accuracy for the compact scheme without implementing Richardson’s extrapolation is second order in temporal dimension.

Table 6.4: Order of accuracy in temporal and spatial dimension with Richardson’s extrapolation (One-dimensional: Problem 2)

<table>
<thead>
<tr>
<th></th>
<th>20</th>
<th>40</th>
<th>80</th>
</tr>
</thead>
<tbody>
<tr>
<td>( h )</td>
<td>0.002172966411</td>
<td>0.0001352021349</td>
<td>8.650748243e-006</td>
</tr>
<tr>
<td>( t )</td>
<td>0.002172966411</td>
<td>0.0001352021349</td>
<td>8.650748243e-006</td>
</tr>
<tr>
<td>( E(h,t) )</td>
<td>-</td>
<td>16.07198297</td>
<td>15.62895268</td>
</tr>
<tr>
<td>( \log_2 r )</td>
<td>-</td>
<td>4.006476036</td>
<td>3.966149199</td>
</tr>
</tbody>
</table>

The table verifies that the order of accuracy for the compact scheme with Richardson’s extrapolation is fourth order in temporal dimension.
Conclusion

Unconditionally stable and efficient finite difference schemes of second and fourth order accuracy have been employed to solve the Burger’s equation. The Crank-Nicolson scheme of second order is implemented using Newton’s Method and a modified form of Gaussian-elimination developed for sparse matrices. The fourth-order scheme based on Hopf-cole transformation, Pade’ approximation and Richardson’s Expolation is implemented. The schemes are implemented with high efficiency and accuracy. Various Numerical examples are taken that demonstrate the robustness, efficiency and accuracy of the schemes. The schemes are then analysed for time-complexity and accuracy. The analysis agrees with the experiment problem’s accuracy and efficiency.
References


Paper communicated based on this work