AE332 – Modeling and Analysis Lab

A Report submitted

For internal assessment for Coarse

Modeling and Analysis Lab

in

Aerospace Engineering

by

Shingala Vaidik Pareshbhai

(SC21B054)

pursued in

Department of Aerospace Engineering

Indian Institute of Space Science and Technology

to

Dr. Manu

INDIAN INSTITUTE OF SPACE SCIENCE AND TECHNOLOGY

THIRUVANANTHAPURAM

November 15, 2023

Contents

Chapter 1

Poisson equation

1.1 Problem to solve

Consider the Poisson equation

$$u_{xx} + u_{yy} = -4(x^2 + y^2), \quad 0 < x < 1, \quad 0 < y < 1,$$

with Dirichlet boundary conditions $u(x, y) = xy$ on the boundary of the square.

1.2 Discretization

Let's discretize the domain using central differences with $\Delta x = \Delta y = 0.2$. Define $u_{i,j} \approx u(x_i, y_j)$ for $i, j = 1, 2, \ldots, N$, where N is the number of grid points in each direction.

The discretized Poisson equation becomes

$$\frac{u_{i+1,j} - 2u_{i,j} + u_{i-1,j}}{\Delta x^2} + \frac{u_{i,j+1} - 2u_{i,j} + u_{i,j-1}}{\Delta y^2} = -4(x_i^2 + y_j^2).$$

1.3 Solution Procedure

1. Initialize the grid with the boundary values: $u_{i,0} = x_i \cdot 0$, $u_{i,N+1} = x_i \cdot 1$, $u_{0,j} = 0 \cdot y_j$, $u_{N+1,j} = 1 \cdot y_j$.
2. Iterate until convergence using the discretized Poisson equation.

$$u_{i,j} = \frac{1}{4}\left(u_{i+1,j} + u_{i-1,j} + u_{i,j+1} + u_{i,j-1} + \Delta x^2 \cdot 4(x_i^2 + y_j^2)\right)$$

3. Repeat until the solution converges or reaches a specified number of iterations.

1.4 Matlab Code

```
% Define grid parameters
nx = 6;
ny = 6;
dx = 0.2;
```

```
dy = 0.2;

% Create a grid for u
u = zeros(nx, ny);

% Set boundary conditions
for i = 1:nx
    u(i, 1) =  0;
    u(i, ny) = (i - 1) * dx * 1;
end
for j = 1:ny
    u(1, j) = 0;
    u(nx, j) = 1*(j-1)*dy;
end

% Perform iterations to solve the Poisson equation
max_iterations = 1000;
tolerance = 1e-5;

for k = 1:max_iterations
    u_new = u;
    for i = 2:5
        for j = 2:5
            u_new(i, j) = (u(i+1, j) + u(i-1, j) + u(i, j+1) + u(i, j-1) + dx^4 *
                4 * ((i)^2+(j)^2)) / 4;
        end
    end

    max_diff = max(max(abs(u_new - u)));
    u = u_new;

    if max_diff < tolerance
        break;
    end
end

% Create a meshgrid for plotting
[X, Y] = meshgrid(0:dx:1, 0:dy:1);

% Plot the solution
figure;
surf(X, Y, u);
title('Poisson Equation Solution');
xlabel('x');
ylabel('y');
```

```
zlabel('u(x, y)');
colorbar;

% Print the solution in a custom format
disp('Solution:');
for j = 1:ny
    for i = 1:nx
        fprintf('u(%.2f, %.2f)=%.2f;\n ', (i — 1) * dx, (j — 1) * dy, u(i, j));
    end
end
```

1.5 Output of Matlab code

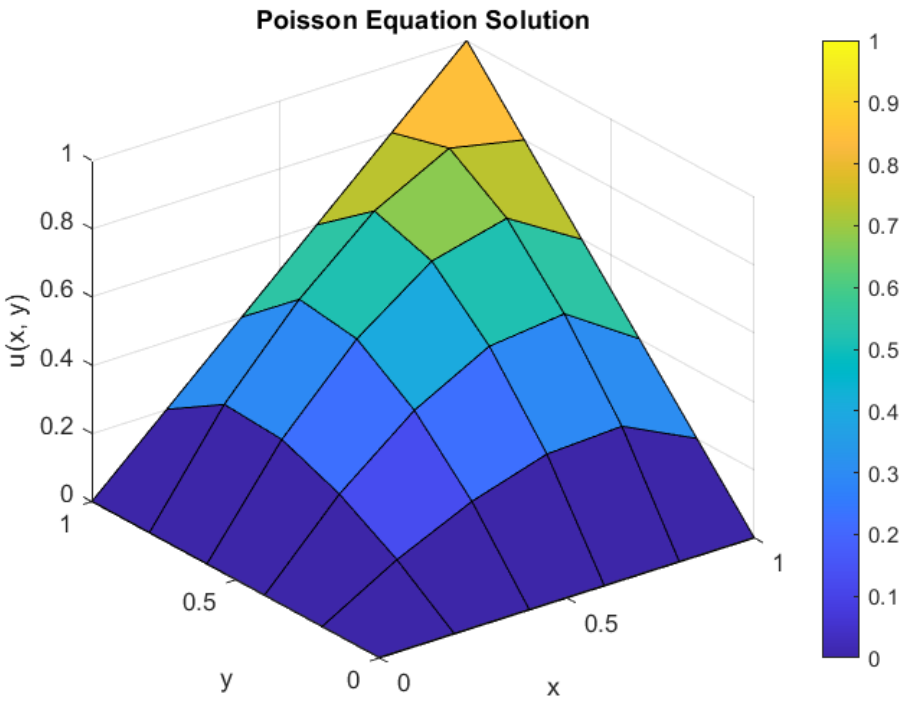


Figure 1.1: Poisson Equation Solution

Solution: u(x,y)

Y/X	0.00	0.20	0.40	0.60	0.80	1.00
0.00	0.00	0.00	0.00	0.00	0.00	0.00
0.20	0.00	0.13	0.23	0.30	0.31	0.20
0.40	0.00	0.23	0.40	0.52	0.54	0.40
0.60	0.00	0.30	0.52	0.68	0.73	0.60
0.80	0.00	0.31	0.54	0.73	0.85	0.80
1.00	0.00	0.20	0.40	0.60	0.80	1.00

Table 1.1: Values of $u(x, y)$ at different positions.

Chapter 2

1D Burgers' equation

2.1 Problem Statement

Consider the 1D Burgers' equation in the interval $0 < x < 1$:

$$\frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} = \nu\frac{\partial^2 u}{\partial x^2}$$

with the initial condition:

$$u(x, 0) = \sin(\pi x), \quad 0 < x < 1$$

2.2 Numerical Solution using FTBS Method

To solve the Burger's equation numerically, we use the FTBS (Forward-Time Backward-Space) method for the convective term and central differencing for the diffusive term.

The numerical solution is obtained by discretizing the equation in both space and time. The FTBS method for the convective term is given by:

$$u_i^{n+1} = u_i^n - \frac{\Delta t}{\Delta x}u_i^n(u_{i+1}^n - u_{i-1}^n)$$

and the central differencing for the diffusive term:

$$u_i^{n+1} = u_i^n + \nu\frac{\Delta t}{\Delta x^2}(u_{i+1}^n - 2u_i^n + u_{i-1}^n)$$

2.3 Analytical Solution

The analytical solution is given by:

$$u(x, t) = \frac{2\pi\nu\sum_{n=1}^{\infty} a_n e^{-n^2\pi^2\nu t}\sin(n\pi x)}{a_0 + \sum_{n=1}^{\infty} a_n e^{-n^2\pi^2\nu t}\cos(n\pi x)}$$

where

$$a_0 = \int_0^1 \exp\left(-\frac{1}{2\pi\nu}[1 - \cos(\pi x)]\right) dx$$

and

$$a_n = 2\int_0^1 \exp\left(-\frac{1}{2\pi\nu}[1 - \cos(\pi x)]\right)\cos(n\pi x)\, dx \quad \text{for } n = 1, 2, 3, \ldots$$

Perform a grid convergence and a time step convergence study for the problem to assess the accuracy of the numerical solution.

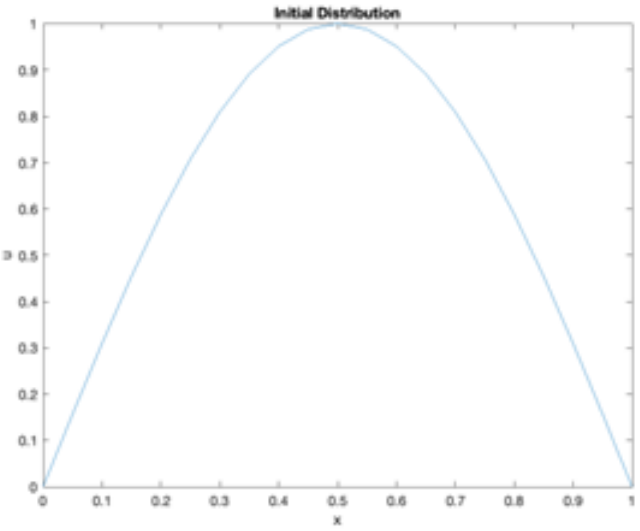
2.4 Matlab code

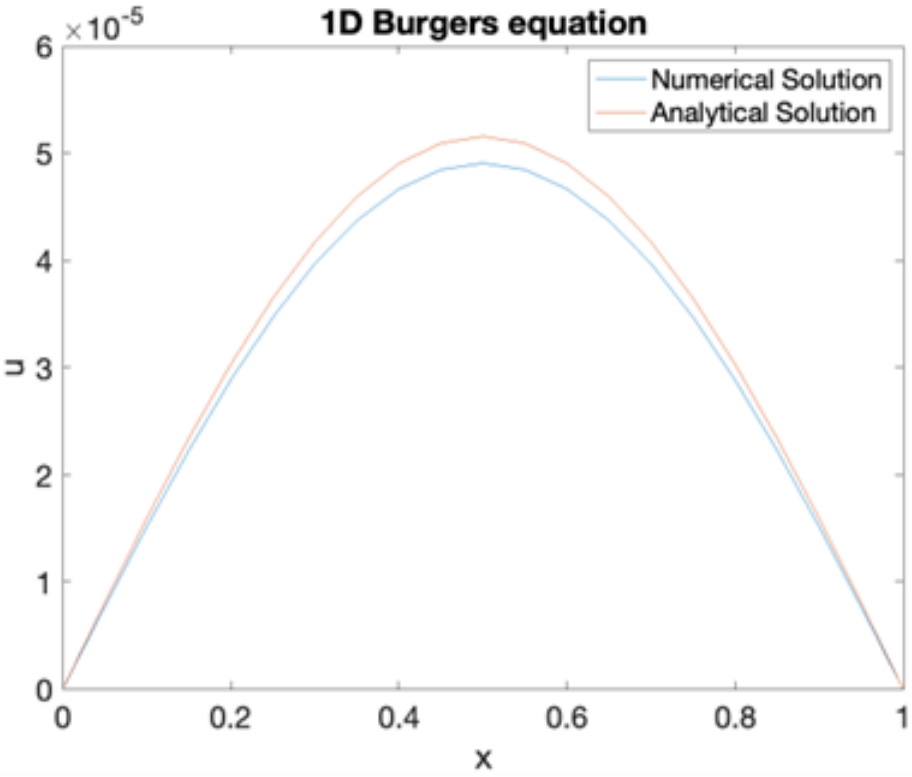
```
% 1 D  Burgers' equation

clear all

clc

% grid convergence
[u_n_1,u_t_1,e1]=burger(0.5,0.002);
[u_n_2,u_t_2,e2]=burger(0.2,0.002);
[u_n_3,u_t_3,e3]=burger(0.10,0.002);
[u_n_4,u_t_4,e4]=burger(0.05,0.002);
[u_n_5,u_t_5,e5]=burger(0.04,0.002);
[u_n_6,u_t_6,e6]=burger(0.02,0.002);

e1
e2
e3
e4
e5
e6
scatter([1/0.5 1/0.2 1/0.1 1/0.05 1/0.04 1/0.02],[e1 e2 e3 e4 e5 e6],"filled")
hold on
plot([1/0.5 1/0.2 1/0.1 1/0.05 1/0.04 1/0.02],[e1 e2 e3 e4 e5 e6])
xlabel("Number of spatial grid points")
ylabel("Error")
hold off

% time step convergence
[u_n_7,u_t_7,e7]=burger(0.05,0.0012);
[u_n_8,u_t_8,e8]=burger(0.05,0.0010);
[u_n_9,u_t_9,e9]=burger(0.05,0.0008);
[u_n_10,u_t_10,e10]=burger(0.05,0.0006);
[u_n_11,u_t_11,e11]=burger(0.05,0.0004);
[u_n_12,u_t_12,e12]=burger(0.05,0.0002);

e7
e8
e9
e10
e11
```

```
e12
scatter([1/0.0012 1/0.001 1/0.0008 1/0.0006 1/0.0004 1/0.0002],[e7 e8 e9 e10 e11
    e12],"filled")
hold on
plot([1/0.0012 1/0.001 1/0.0008 1/0.0006 1/0.0004 1/0.0002],[e7 e8 e9 e10 e11 e12
    ])
xlabel("Number of time grid points")
ylabel("Error")

function [u,sol_x,error]=burger(dx,dt)
dx=0.05; %grid size
nx=1+1/dx; % number of points in x direction
x=0:dx:1;
dt = 0.001;
nt =1/dt;% number of time steps
mu =1 ;
u=zeros(1,nx);
un=zeros(1,nx);

for j=1:nx
    u(j)=sin(pi*x(j));
end

plot(x,u)
xlabel('x')
ylabel('u')
title ('Initial Distribution')


for it =1:nt
    un=u;
        for i=2:nx-1

            u(i) = un(i) - un(i)*(dt/dx)*(un(i)-un(i-1))+mu*(dt/dx^2)*(un(i+1)-2*
                un(i)+un(i-1));

        end


end
 plot(x,u);
    xlabel('x')
    ylabel('u')
    set(gca,'FontSize',18)
    title('1D Burgers equation')
```

```
    legend(['time=' num2str(it*dt)])


hold on

syms z
a_n_arr=zeros;
for j=1:16
    integrand=2*exp(-1*(1-cos(pi*z))/(2*pi))*cos((j-1)*pi*z);
    a_n_arr(j)=vpa(int(integrand,z,0,1));
end
a_n_arr(1)=a_n_arr(1)/2;


time=nt*dt;
sol_x=zeros(1,nx);
for i=1:nx
    x1=x(i);
    num=0;
    den=0;
    for n=1:j-1
        num=num+(a_n_arr(n+1))*exp(-n^2*pi^2*time)*n*sin(n*pi*x1);
        den=den+(a_n_arr(n+1))*exp(-n^2*pi^2*time)*cos(n*pi*x1);
    end
    sol_x(i)=2*pi*num/(a_n_arr(1)+den);
end
plot(x,sol_x)
legend ('Numerical Solution','Analytical Solution')
error=abs(max(u)-max(sol_x))
end
```

2.5 Output of Matlab code



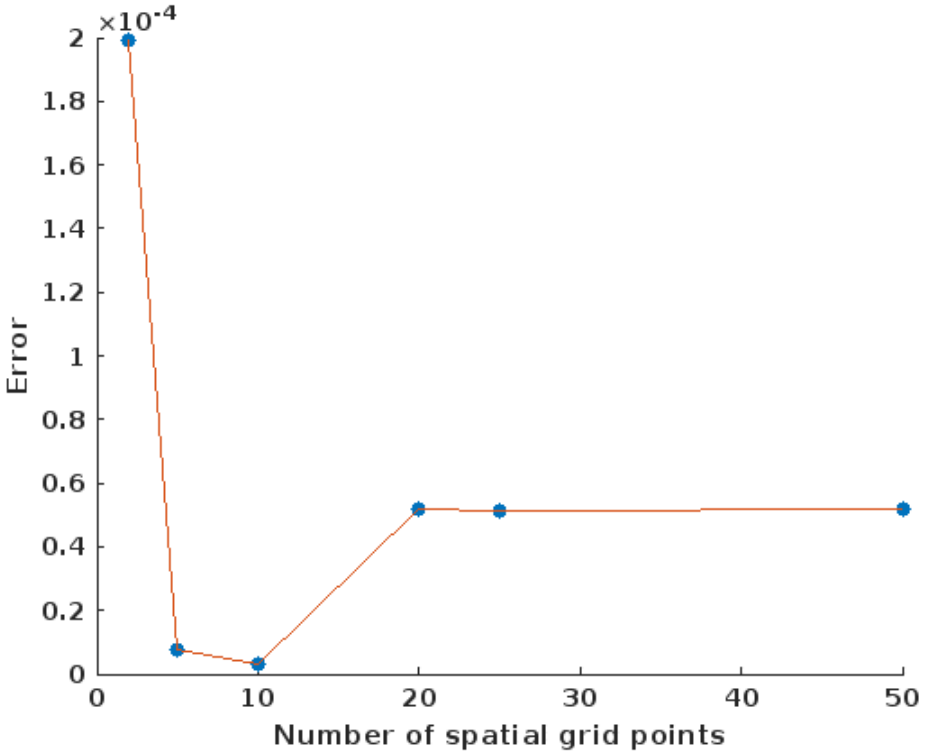


Grid convergence: Time step: 0.002 s

Spatial steps: $0.5,\ 0.2,\ 0.1,\ 0.05,\ 0.04,\ 0.02$

Errors: $e1,\ e2,\ e3,\ e4,\ e5,\ e6$

```
e1 = 1.9910e-04
e2 = 7.7258e-06
e3 = 2.9129e-06
e4 = 5.1560e-05
e5 = 5.1458e-05
e6 = 5.1560e-05
```

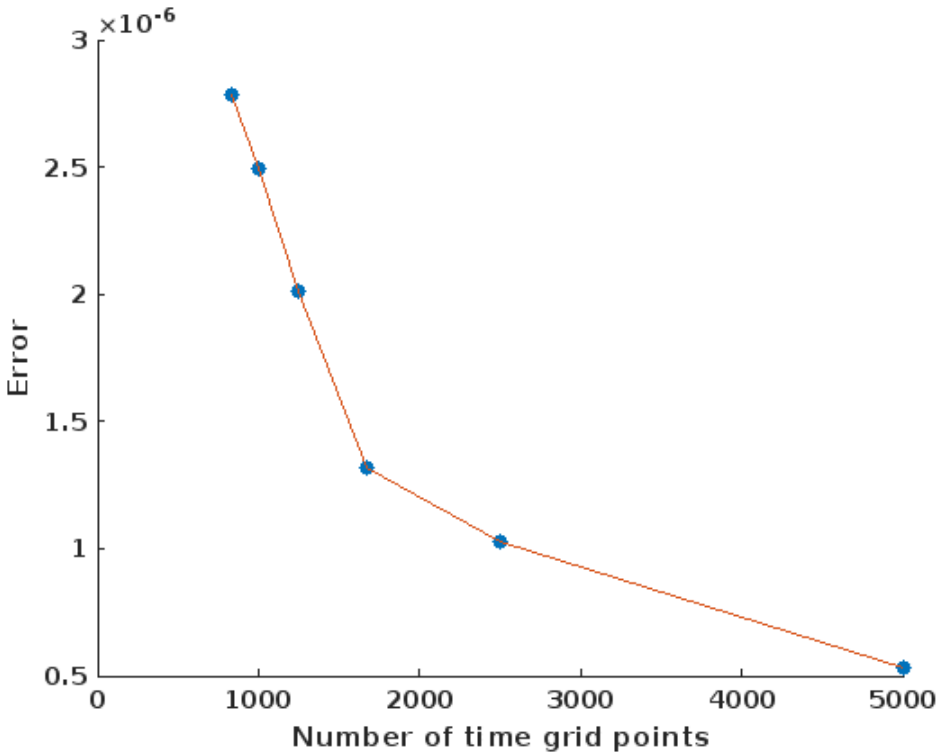


Time step convergence: Grid step: 0.05 s

Temporal steps: $0.0012,\ 0.001,\ 0.0008,\ 0.0006,\ 0.0004,\ 0.0002$

Errors: $e7,\ e8,\ e9,\ e10,\ e11,\ e12$

```
e7 = 2.7860e-06
e8 = 2.4958e-06
e9 = 2.0093e-06
e10 = 1.3211e-06
e11 = 1.0256e-06
e12 = 5.2842e-07
```


×10-6
3
2.5
2
1.5
1
0.5
Error
0
1000
2000
3000
4000
5000
Number of time grid points

Chapter 3

2D linear convection

3.1 Governing Equations

The governing equations for the 2D linear convection problem are given by:

$$\frac{\partial u}{\partial t} + c\frac{\partial u}{\partial x} + c\frac{\partial v}{\partial y} = 0$$

$$\frac{\partial v}{\partial t} + c\frac{\partial u}{\partial x} + c\frac{\partial v}{\partial y} = 0$$

where c is the convection speed.

3.2 Initial and Boundary Conditions

The initial conditions are given by:

$$u(x, y, 0) = f(x, y) \quad \text{and} \quad v(x, y, 0) = g(x, y)$$

The boundary conditions are typically specified on the boundaries of the computational domain.

3.3 Numerical Solution using FTBS Method

To solve the 2D linear convection problem numerically, we use the FTBS method. The FTBS method for the convection terms is given by:

$$u_{i,j}^{n+1} = u_{i,j}^n - c\frac{\Delta t}{\Delta x}(u_{i,j}^n - u_{i-1,j}^n) - c\frac{\Delta t}{\Delta y}(u_{i,j}^n - u_{i,j-1}^n)$$

$$v_{i,j}^{n+1} = v_{i,j}^n - c\frac{\Delta t}{\Delta x}(v_{i,j}^n - v_{i-1,j}^n) - c\frac{\Delta t}{\Delta y}(v_{i,j}^n - v_{i,j-1}^n)$$

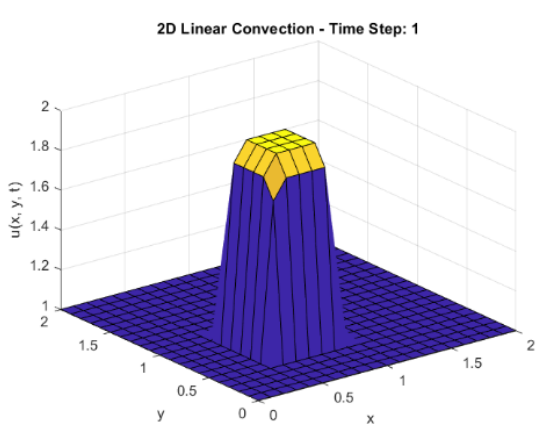
The numerical solution at each grid point (x_i, y_j) provides the solution to the 2D linear convection problem.

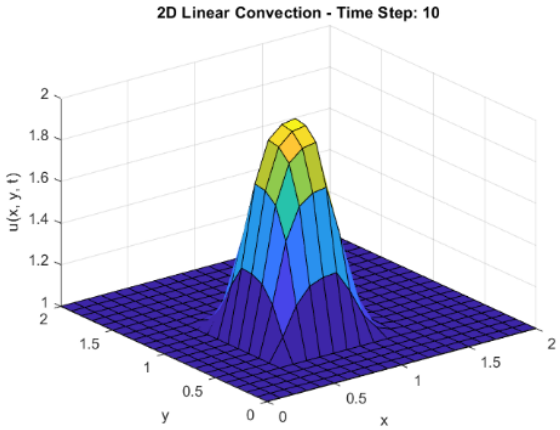
3.4 Matlab code

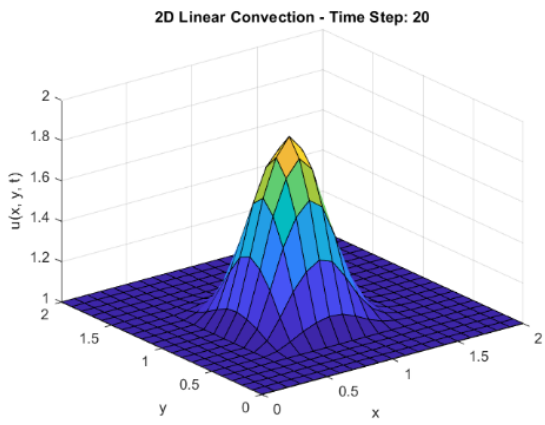
```
% Parameters
c = 1; % Convection velocity in x direction
Lx = 2; % Domain size in x direction
Ly = 2; % Domain size in y direction
Nx = 21; % Number of grid points in x direction
Ny = 21; % Number of grid points in y direction
T = 0.5; % Total simulation time
dx = Lx / (Nx - 1); % Grid spacing in x direction
dy = Ly / (Ny - 1); % Grid spacing in y direction
dt = 0.01; % Time step
Nt = 51; % Number of time steps

x = linspace(0, Lx, Nx);
y = linspace(0, Ly, Ny);

% Initial condition
u = ones(Nx, Ny);
v = ones(Nx, Ny);
for i = 5:10
    for j = 5:10
        x_val = (i - 1) * dx;
        y_val = (j - 1) * dy;
        if 0.5 <= x_val && x_val <= 1 && 0.5 <= y_val && y_val <= 1
            u(i, j) = 2;
            v(i, j) = 2;
        else
            u(i, j) = 1;
            v(i, j) = 1;
        end
    end
end

[X, Y] = meshgrid(x, y);

% FTBS method
for n = 1:Nt
    un=u;
    vn=v;
    for i = 2:Nx
        for j = 2:Ny
            u(i, j) = un(i, j) - c * dt / dx * (un(i, j) - un(i-1, j)) - c * dt /
                dy * (un(i, j) - un(i, j-1));
            v(i, j) = vn(i, j) - c * dt / dx * (vn(i, j) - vn(i-1, j)) - c * dt /
```

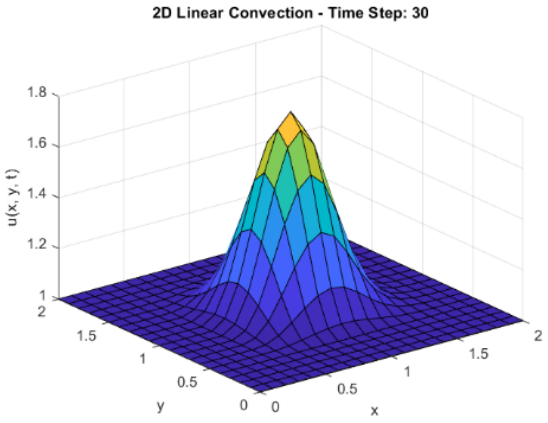
```
                dy * (vn(i, j) - vn(i, j-1));
            end
        end

        % Plot the solution at specific time steps
        if mod(n, 1) == 0
            surf(X, Y, u);
            title(['2D Linear Convection - Time Step: ' num2str(n)]);
            xlabel('x');
            ylabel('y');
            zlabel('u(x, y, t)');
            drawnow;
        end
    end

    % Plot the final solution
    figure;
    surf(X, Y, u);
    title('Final Solution of 2D Linear Convection u');
    xlabel('x');
    ylabel('y');
    zlabel('u(x, y, t)');

    % Plot the final solution
    figure;
    surf(X, Y, v);
    title('Final Solution of 2D Linear Convection v');
    xlabel('x');
    ylabel('y');
    zlabel('v(x, y, t)');
```

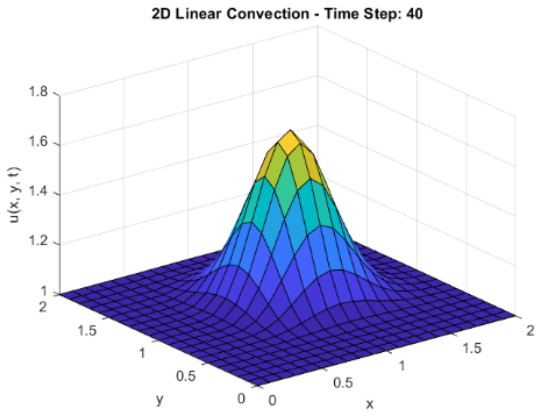
3.5 Output of Matlab code

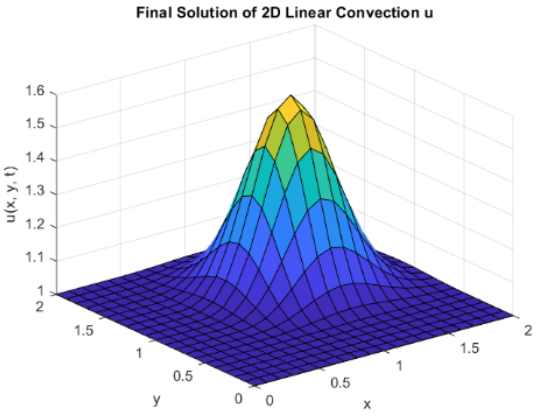


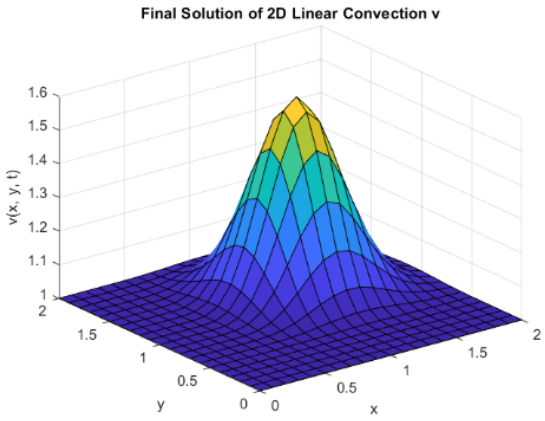



2D Linear Convection - Time Step: 20
u(x, y, t)
x
y


2D Linear Convection - Time Step: 30
u(x, y, t)
x
y


2D Linear Convection - Time Step: 40
u(x, y, t)
x
y


Final Solution of 2D Linear Convection u
u(x, y, t)
x
y


Final Solution of 2D Linear Convection v
v(x, y, t)
x
y