

# Simulating 2D-Convection and Diffusion equation.

Ranjithkumar B

SC22M008, M.Tech. Aerospace - Aerodynamics and Flight Mechanics

## I. Problem Definition

- 1) 2D linear convection problem using the FTBS method, forward time difference, backward space difference.
- 2) 2D linear diffusion with FD in time and CD in space

### A. Boundary condition

$x=0$  and  $x=2$ ,  $y=0$  and  $y=2$

t	n	$u(x,y,t)$ $v(x,y,t)$
$0 \leq t \leq 0.5$	$0 \leq n \leq 50$	1

### B. Initial condition

$t = 0$

x	i	y	j	$u(x,y,t)$ $v(x,y,t)$
0	0	0	0	1
$0 \leq x \leq 0.5$	$0 \leq i \leq 5$	$0 \leq y \leq 0.5$	$0 \leq j \leq 5$	1
$0.5 \leq x \leq 1$	$5 \leq i \leq 10$	$0.5 \leq y \leq 1$	$5 \leq j \leq 10$	2
$1 \leq x \leq 2$	$10 \leq i \leq 20$	$1 \leq y \leq 2$	$10 \leq j \leq 20$	1
2	20	2	20	1

## II. Governing Equations

Two dimensional linear convective equations Equations (1) and (2)

$$\frac{\partial u}{\partial t} + c \frac{\partial u}{\partial x} + c \frac{\partial u}{\partial y} = 0 \quad (1)$$

$$\frac{\partial v}{\partial t} + c \frac{\partial v}{\partial x} + c \frac{\partial v}{\partial y} = 0 \quad (2)$$

Two dimensional linear diffusion equations Equations (3) and (4)

$$\frac{\partial u}{\partial t} = \nu \left( \frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} \right) \quad (3)$$

$$\frac{\partial v}{\partial t} = \nu \left( \frac{\partial^2 v}{\partial x^2} + \frac{\partial^2 v}{\partial y^2} \right) \quad (4)$$

## III. Numerical calculation

The numerical scheme is used here is the Forward differencing in time and Backward differencing in space (FTBS) in linear convection and Forward time and central differencing in space in linear diffusion in u component is mentioned in

Equations (5) to (8)

$$\frac{u_i^{n+1} - u_i^n}{\Delta t} = -c \left( \frac{u_i^n - u_{i-1}^n}{\Delta x} + c \frac{u_i^n - u_{i-1}^n}{\Delta y} \right) \quad (5)$$

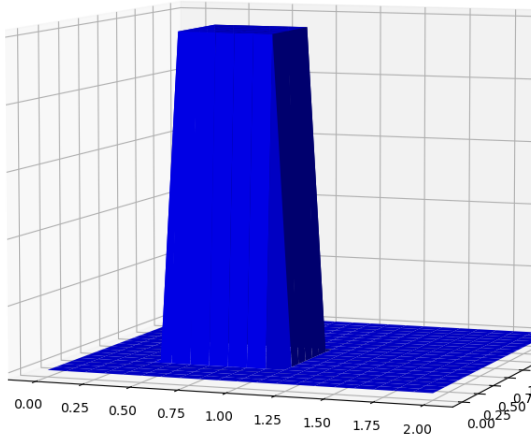
$$\frac{v_i^{n+1} - v_i^n}{\Delta t} = -c \left( \frac{v_i^n - v_{i-1}^n}{\Delta x} + c \frac{v_i^n - v_{i-1}^n}{\Delta y} \right) \quad (6)$$

$$\frac{u_i^{n+1} - u_i^n}{\Delta t} = v \left( \frac{u_{i+1}^n - 2u_i^n + u_{i-1}^n}{\Delta x^2} + \frac{u_{i+1}^n - 2u_i^n + u_{i-1}^n}{\Delta y^2} \right) \quad (7)$$

$$\frac{v_i^{n+1} - v_i^n}{\Delta t} = v \left( \frac{v_{i+1}^n - 2v_i^n + v_{i-1}^n}{\Delta x^2} + \frac{v_{i+1}^n - 2v_i^n + v_{i-1}^n}{\Delta y^2} \right) \quad (8)$$

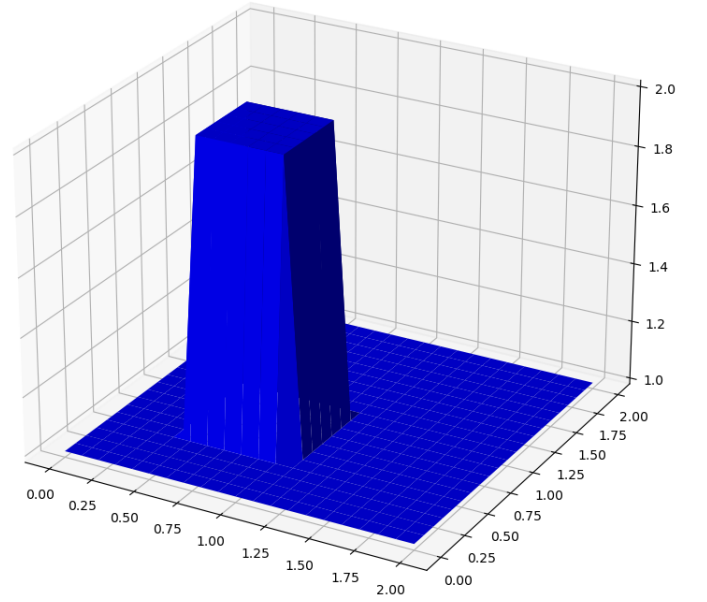
## IV. Results

u value at t=0



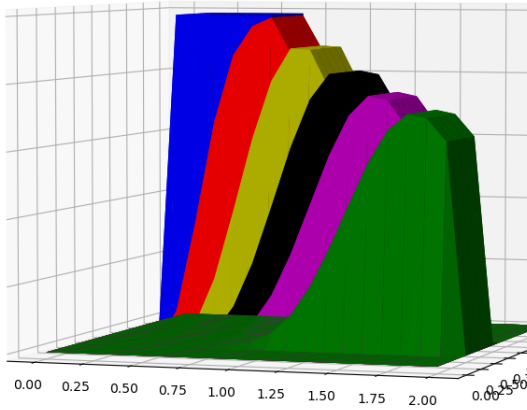
(a) Initial value of U

v value at t=0



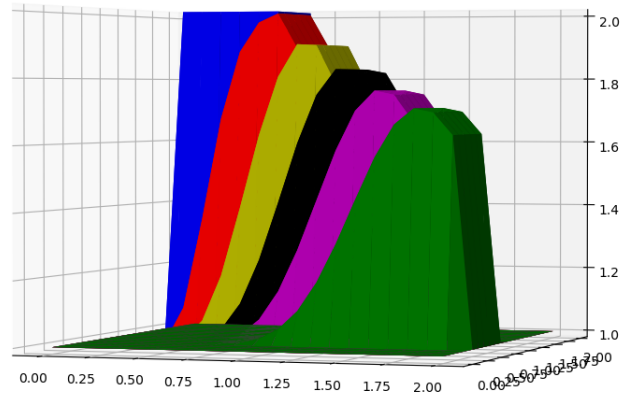
(b) Initial value of V

Change in u with respect to time



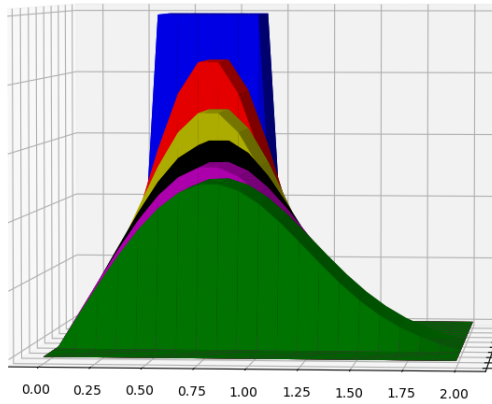
(a) Linear Convection of U at each 0.1 time steps

Change in v with respect to time



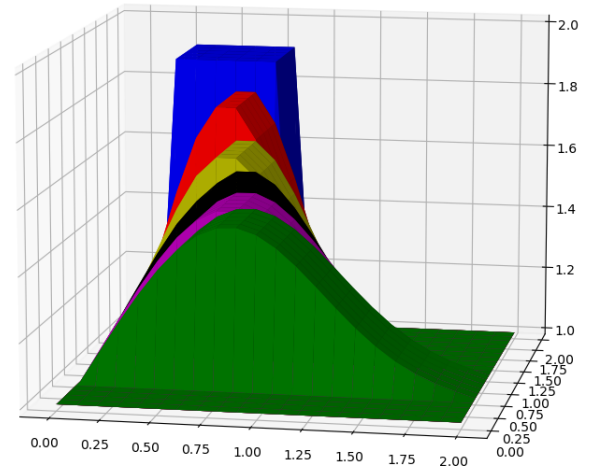
(b) Linear Convection of V at each 0.1 time steps

Change in u with respect to time



(a) Linear Diffusion of U at each 0.1 time steps

Change in v with respect to time



(b) Linear Diffusion of V at each 0.1 time steps

## A. Appendix - Python code for Linear Convection

```
# /bin/python3
2
import numpy as np
4 import matplotlib.pyplot as plt
import matplotlib as mpl
6 import pandas as pd

8 Lx = 2.0
Ly = 2.0
10 max_t = 0.5
c = 1

12 Nx = 21
14 Ny = 21
Nt = 51

16 [x,y] = [np.linspace(0,2,Nx),np.linspace(0,2,Ny)]
18 X,Y = np.meshgrid(x,y)

20 dt = max_t/(Nt-1)
dx = Lx/(Nx-1)
22 dy = Ly/(Ny-1)

24 #
# Boundary condition
26 u = np.ones([Nt,Ny,Nx])
v = np.ones([Nt,Ny,Nx])

28 for i in range(Nx):
30     u[0,0,i] = 1.0
    v[0,0,i] = 1.0
32
for i in range(Nx):
34     u[0,Ny-1,i] = 1.0
    v[0,Ny-1,i] = 1.0
36
for j in range(Ny):
38     u[0,j,0] = 1.0
    v[0,j,0] = 1.0
40
for j in range(Nx):
42     u[0,j,Nx-1] = 1.0
    v[0,j,Nx-1] = 1.0
44
#
46 # initial condition
for i in range(Nx):
48     if x[i]==0.5:
        x1 = i
        print(x1)
        if x[i]==1:
            x2 = i
            print(x2)
54
for j in range(Ny):
56     if y[j]==0.5:
        y1 = j
        if y[j]==1:
            y2 = j
60
for i in range(x1,x2+1):
62     for j in range(y1,y2+1):
        u[0,j,i] = 2.0
        v[0,j,i] = 2.0
64
#
```

```

# computing section
68 alpha = c*dt*(1/dx+1/dy)
   print(alpha)
70 for k in range(1,Nt):
       for j in range(1,Ny-1):
92         for i in range(1,Nx-1):
               u[k,j,i] = u[k-1,j,i]-alpha*(u[k-1,j,i]-u[k-1,j,i-1])
94               v[k,j,i] = v[k-1,j,i]-alpha*(v[k-1,j,i]-v[k-1,j,i-1])

76 #
# plotting section
78 fig = plt.figure()

80 ax = fig.gca(projection='3d')
   l1=ax.plot_surface(X,Y,v[00],color='b')
82 plt.title("v value at t=0")
   plt.show()

84 fig = plt.figure()

86 ax = fig.gca(projection='3d')
   l1=ax.plot_surface(X,Y,v[00],color='b')
88   l2=ax.plot_surface(X,Y,v[10],color='r')
   l3=ax.plot_surface(X,Y,v[20],color='y')
90   l4=ax.plot_surface(X,Y,v[30],color='k')
   l5=ax.plot_surface(X,Y,v[40],color='m')
92   l6=ax.plot_surface(X,Y,v[50],color='g')
   plt.title("Change in v with respect to time")
94   plt.show()

96 fig = plt.figure()

98 ax = fig.gca(projection='3d')
100 l1=ax.plot_surface(X,Y,u[00],color='b')
   plt.title("u value at t=0")
102 plt.show()

104 fig = plt.figure()

106 ax = fig.gca(projection='3d')
   l1=ax.plot_surface(X,Y,u[00],color='b')
108   l2=ax.plot_surface(X,Y,u[10],color='r')
   l3=ax.plot_surface(X,Y,u[20],color='y')
110   l4=ax.plot_surface(X,Y,u[30],color='k')
   l5=ax.plot_surface(X,Y,u[40],color='m')
112   l6=ax.plot_surface(X,Y,u[50],color='g')
   plt.title("Change in u with respect to time")
114 plt.show()

```

## B. Appendix - Python code or Linear Diffusion

```
# /bin/python3
2
import numpy as np
4 import matplotlib.pyplot as plt
import matplotlib as mpl
6 import pandas as pd

8 Lx = 2.0
Ly = 2.0
10 max_t = 0.5
nu = 0.1
12 a,b = 0.5,1.0

14 Nx = 21
Ny = 21
16 Nt = 51

18 [x,y] = [np.linspace(0,2,Nx),np.linspace(0,2,Ny)]
X,Y = np.meshgrid(x,y)
20
dt = max_t/(Nt-1)
22 dx = Lx/(Nx-1)
dy = Ly/(Ny-1)
24
# -----
26 # Boundary condition
u = np.ones([Nt,Ny,Nx])
28 v = np.ones([Nt,Ny,Nx])

30 for i in range(Nx):
    u[0,0,i] = 1.0
32     v[0,0,i] = 1.0

34 for i in range(Nx):
    u[0,Ny-1,i] = 1.0
36     v[0,Ny-1,i] = 1.0

38 for j in range(Ny):
    u[0,j,0] = 1.0
40     v[0,j,0] = 1.0

42 for j in range(Nx):
    u[0,j,Nx-1] = 1.0
44     v[0,j,Nx-1] = 1.0

46 # -----
# initial condition
48 for i in range(Nx):
    if x[i]==a:
50         x1 = i
        print(x1)
52     if x[i]==b:
        x2 = i
54         print(x2)

56 for j in range(Ny):
    if y[j]==a:
58         y1 = j
        if y[j]==b:
60             y2 = j

62 for i in range(x1,x2+1):
    for j in range(y1,y2+1):
64         u[0,j,i] = 2.0
        v[0,j,i] = 2.0
66
```

```

#
68 # computing section
alpha = nu*dt*(1/dx**2+1/dy**2)
70 print(alpha)
for k in range(1,Nt):
72     for j in range(1,Ny-1):
        for i in range(1,Nx-1):
74             u[k,j,i] = u[k-1,j,i]+alpha*(u[k-1,j,i+1]-2*u[k-1,j,i]+u[k-1,j,i-1])
                v[k,j,i] = v[k-1,j,i]+alpha*(v[k-1,j,i+1]-2*v[k-1,j,i]+v[k-1,j,i-1])
76
#
78 # plotting section
fig = plt.figure()
80
ax = fig.gca(projection='3d')
82 l1=ax.plot_surface(X,Y,v[00],color='b')
ax.set_title("v vale at t=0")
84 plt.show()

86 fig = plt.figure()

88 ax = fig.gca(projection='3d')
l1=ax.plot_surface(X,Y,v[00],color='b')
90 l2=ax.plot_surface(X,Y,v[10],color='r')
l3=ax.plot_surface(X,Y,v[20],color='y')
92 l4=ax.plot_surface(X,Y,v[30],color='k')
l5=ax.plot_surface(X,Y,v[40],color='m')
94 l6=ax.plot_surface(X,Y,v[50],color='g')
ax.set_title("Change in v with respect to time")
96 plt.show()

98 fig = plt.figure()

100 ax = fig.gca(projection='3d')
l1=ax.plot_surface(X,Y,u[00],color='b')
102 ax.set_title("u vale at t=0")
plt.show()

104 fig = plt.figure()

106 ax = fig.gca(projection='3d')
108 l1=ax.plot_surface(X,Y,u[00],color='b')
l2=ax.plot_surface(X,Y,u[10],color='r')
110 l3=ax.plot_surface(X,Y,u[20],color='y')
l4=ax.plot_surface(X,Y,u[30],color='k')
112 l5=ax.plot_surface(X,Y,u[40],color='m')
l6=ax.plot_surface(X,Y,u[50],color='g')
114 plt.title("Change in u with respect to time")
plt.show()

```