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≤t≤0.5	0≤n≤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≤x≤0.5	0≤i≤5	0≤y≤0.5	0≤j≤5	1
0.5≤x≤1	5≤i≤10	0.5≤y≤1	5≤j≤10	2
1≤x≤2	10≤i≤20	1≤y≤2	10≤j≤20	1
2	20	2	20	1

II. Governing Equations

Two dimentional linear convective equationsEquations (1) and (2)

$$\frac{\partial u}{\partial t} + c\frac{\partial u}{\partial x} + c\frac{\partial u}{\partial y} = 0 \tag{1}$$

$$\frac{\partial v}{\partial t} + c\frac{\partial v}{\partial x} + c\frac{\partial v}{\partial y} = 0 \tag{2}$$

Two dimentional linear diffusion equationsEquations (3) and (4)

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

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

III. Numerical calculation

The numerical scheme is used here is the Forward diffrencing in time and Backward diifrencing in space (FTBS) in linear convection and Forward time and cntrakl diffrencing in space in linear diffusion in u component is metioned in

Equations (5) to (8)

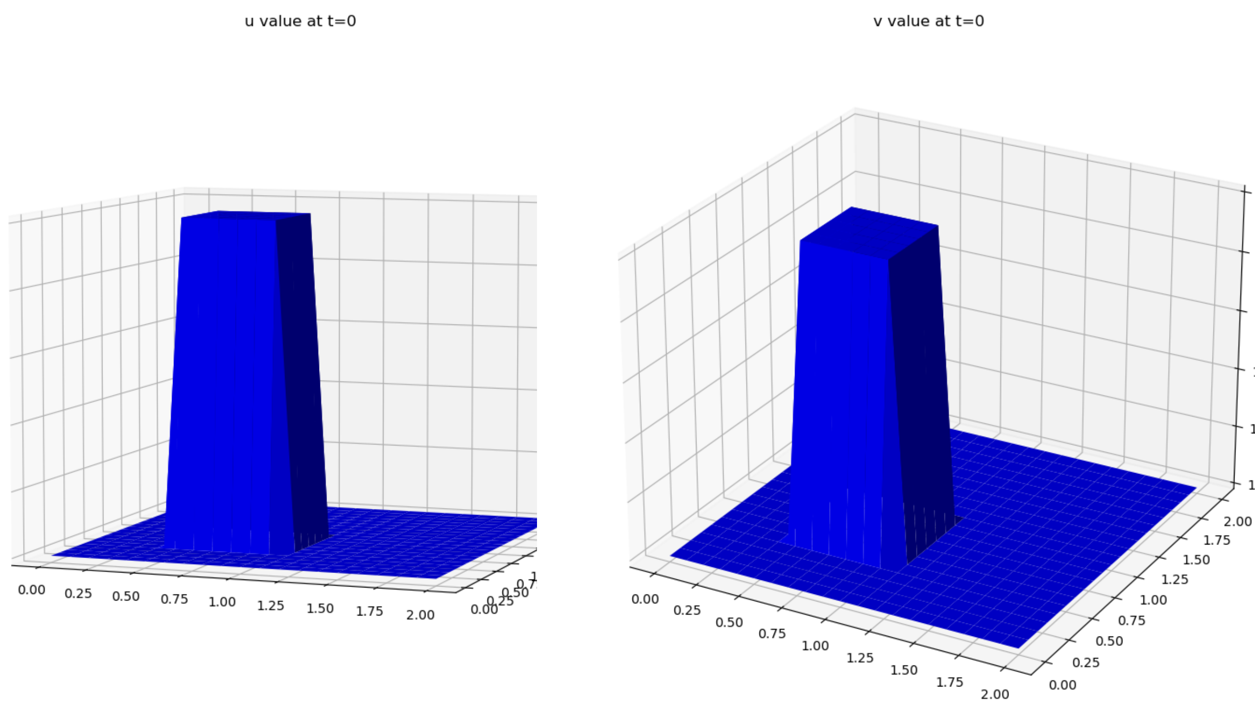
$$\frac{u_i^{n+1} - u_i^n}{\Delta t} = -c \left(\frac{u_i^n - u_{i-1}^n}{\Delta x} + c \frac{u_i^n - u_{i-1}^n}{\Delta y} \right) \tag{5}$$

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

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

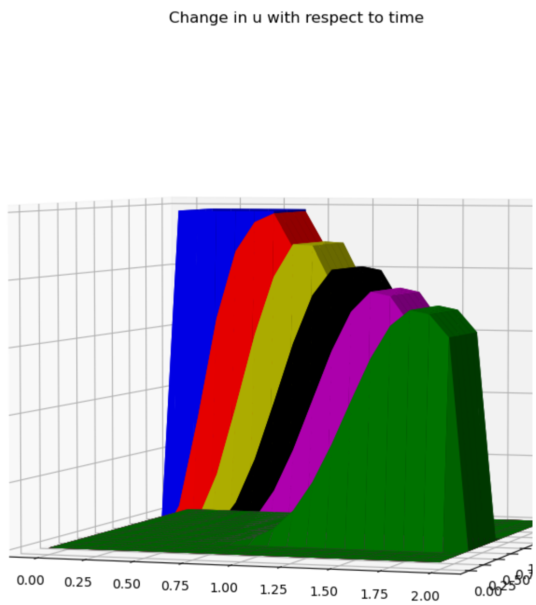
$$\frac{v_i^{n+1} - v_i^n}{\Delta t} = \nu \left(\frac{v_{i+1}^n - 2v_i^n + v_{i-1}^n}{\Delta x^2} + \frac{v_{i+1}^n - 2v_i^n + v_{i-1}^n}{\Delta y^2} \right) \tag{8}$$

IV. Results

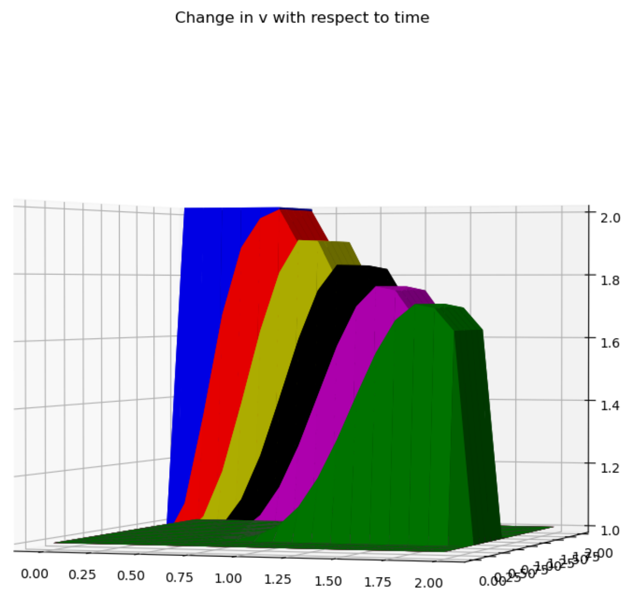


(a) Initial value of U

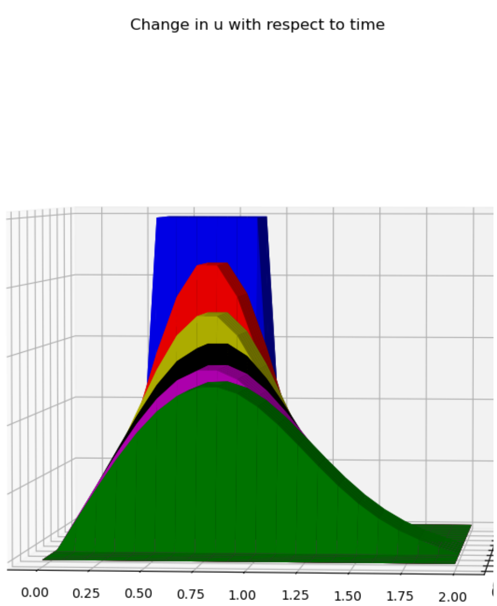
(b) Initial value of V



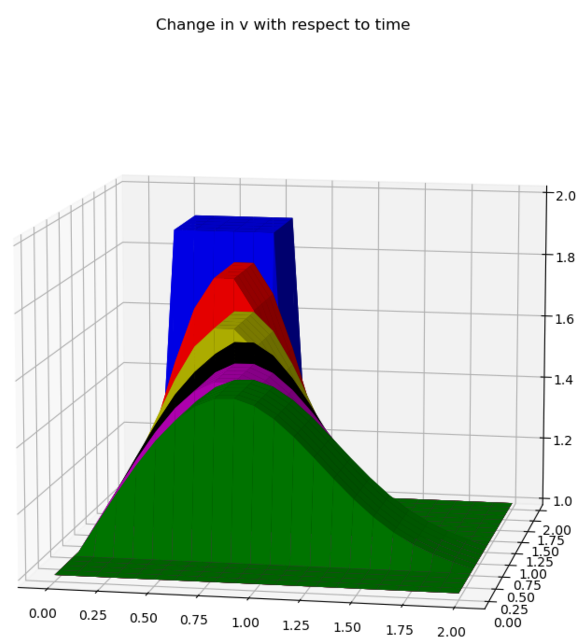
(a) **Linear Convection of U at each 0.1 time steps**



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



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



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

A. Appendix - Python code for Linear Convection

```
# /bin/python3

import numpy as np
import matplotlib.pyplot as plt
import matplotlib as mpl
import pandas as pd

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

Nx = 21
Ny = 21
Nt = 51

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

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

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

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

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

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

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

#------------------------------------------------------------------------
# initial condition
for i in range(Nx):
    if x[i]==0.5:
        x1 = i
        print(x1)
    if x[i]==1:
        x2 = i
        print(x2)

for j in range(Ny):
    if y[j]==0.5:
        y1 = j
    if y[j]==1:
        y2 = j

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

#------------------------------------------------------------------------
```

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

#--------------------------------------------------------------------------------
# ploting section
fig = plt.figure()

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

fig = plt.figure()

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

fig = plt.figure()

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

fig = plt.figure()

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

B. Appendix - Python code or Linear Diffusion

```
# /bin/python3

import numpy as np
import matplotlib.pyplot as plt
import matplotlib as mpl
import pandas as pd

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

Nx = 21
Ny = 21
Nt = 51

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

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

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

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

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

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

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

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

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

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

```

```
#-------------------------------------------------------------------------------
# computing section
alpha = nu*dt*(1/dx**2+1/dy**2)
print(alpha)
for k in range(1,Nt):
    for j in range(1,Ny-1):
        for i in range(1,Nx-1):
            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])

#-------------------------------------------------------------------------------
# ploting section
fig = plt.figure()

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

fig = plt.figure()

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

fig = plt.figure()

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

fig = plt.figure()

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