Computational Physics

HW4

Cheng-Ching Lin

1.

The origin formula is

$$\frac{\partial \rho}{\partial t} = -\frac{\partial(\rho v)}{\partial x}$$

$$\frac{\partial v}{\partial t} = -v\frac{\partial v}{\partial x} - \frac{1}{\rho}\frac{\partial p}{\partial x} + \frac{f}{\rho}$$

(a).FTCS method

FTCS means that the formula transform follows the next two rules:

$$\frac{\partial a}{\partial t} \to \frac{a_i^{n+1} - a_i^n}{\tau}$$

$$\frac{\partial a}{\partial x} \to \frac{a_{i+1}^n - a_{i-1}^n}{2h}$$

For

$$t_n = t_0 + (n-1)\tau$$

$$x^i = x^0 + (i-1)h$$

So, the formula will become the followings:

$$\rho_i^{n+1} = \rho_i^n - \frac{\tau}{2h}\rho_i^n(v_{i+1}^n - v_{i-1}^n) - \frac{\tau}{2h}v_i^n(\rho_{i+1}^n - \rho_{i-1}^n)$$

$$v_i^{n+1} = v_i^n - \frac{\tau}{2h}v_i^n(v_{i+1}^n - v_{i-1}^n) - \frac{\tau}{2h}\frac{1}{\rho_i^n}(\rho_{i+1}^n - \rho_{i-1}^n) + \frac{f\tau}{\rho_i^n}$$

(b).Lax method

Lax method is just FTCS method but add one rule:

$$a_i^n = \frac{a_{i+1}^n + a_{i-1}^n}{2}$$

The formula will become the followings:

$$\rho_i^{n+1} = \frac{\rho_{i+1}^n + \rho_{i-1}^n}{2} - \frac{\tau}{2h}\frac{\rho_{i+1}^n + \rho_{i-1}^n}{2}(v_{i+1}^n - v_{i-1}^n) - \frac{\tau}{2h}\frac{v_{i+1}^n + v_{i-1}^n}{2}(\rho_{i+1}^n - \rho_{i-1}^n)$$

$$v_i^{n+1} = \frac{v_{i+1}^n + v_{i-1}^n}{2} - \frac{\tau}{2h}\frac{v_{i+1}^n + v_{i-1}^n}{2}(v_{i+1}^n - v_{i-1}^n) - \frac{\tau}{h}\frac{1}{\rho_{i+1}^n + \rho_{i-1}^n}(\rho_{i+1}^n - \rho_{i-1}^n) + \frac{2f\tau}{\rho_{i+1}^n + \rho_{i-1}^n}$$

(c).Leap-frog method

Leap-frog method follows these rules:

$$\frac{\partial a}{\partial t} \to \frac{a_i^{n+1} - a_i^{n-1}}{2\tau}$$

$$\frac{\partial a}{\partial x} \to \frac{a_{i+1}^n - a_{i-1}^n}{2h}$$

Therefore, the formula will become this:

$$\rho_i^{n+1} = \rho_i^{n-1} - \frac{\tau}{h}\rho_i^n(v_{i+1}^n - v_{i-1}^n) - \frac{\tau}{h}v_i^n(\rho_{i+1}^n - \rho_{i-1}^n)$$

$$v_i^{n+1} = v_i^{n-1} - \frac{\tau}{h}v_i^n(v_{i+1}^n - v_{i-1}^n) - \frac{\tau}{2h}\frac{1}{\rho_i^n}(\rho_{i+1}^n - \rho_{i-1}^n) + \frac{2f\tau}{\rho_i^n}$$

2.

In this case, the condition is given:

$$\rho = kp, k = 1, f = 0$$

and I use Lax method to solve this question

the code:

```
import matplotlib.pyplot as plt
import numpy as np
tau = 0.001 #time span
h = 0.1#space span
leng = int(1/h+1)
def press(jdx,idx,pp,vv): #I create the function to produce p
    tt = jdx*tau
    xx = idx*h
    const = tau / (2 * h)
    if tt==0:#initial condition
        if xx>=0.4 and xx<0.7:
            ans = 1
        else:
            ans = 0
        return ans
    else:
        if xx == 0:#I suppose there is no anything pressure outside the boundary
            ans = 0.5* (pp[(jdx - 1) * leng + idx + 1]) - const * 0.5 * (
                    pp[(jdx - 1) * leng + idx + 1] ) * (
                          vv[(jdx - 1) * leng + idx + 1]) - const * 0.5 * (
                          vv[(jdx - 1) * leng + idx + 1] ) * (
                          pp[(jdx - 1) * leng + idx + 1] )
        elif xx == 1:
            ans = 0.5 * (pp[(jdx - 1) * leng + idx - 1]) - const * 0.5 * (pp[(jdx - 1) * leng + idx - 1]) * (- vv[(jdx
- 1) * leng + idx - 1]) - const * 0.5 * (vv[(jdx - 1) * leng + idx - 1]) * ( - pp[(jdx - 1) * leng + idx - 1])
        else:
            ans = 0.5 * (pp[(jdx - 1) * leng + idx + 1] + pp[(jdx - 1) * leng + idx - 1]) - const * 0.5 * (
                    pp[(jdx - 1) * leng + idx + 1] + pp[(jdx - 1) * leng + idx - 1]) * (
                          vv[(jdx - 1) * leng + idx + 1] - vv[(jdx - 1) * leng + idx - 1]) - const * 0.5 * (
                          vv[(jdx - 1) * leng + idx + 1] + vv[(jdx - 1) * leng + idx - 1]) * (
                          pp[(jdx - 1) * leng + idx + 1] - pp[(jdx - 1) * leng + idx - 1])
        return ans

def vol(jdx,idx,pp,vv):#I create the function to produce v
    tt = jdx * tau
    xx = idx * h
    const = tau / (2 * h)
```

```
    if tt==0:#initial condition
        ans =0
    elif xx==0 or xx==1:#boundary condition
        ans = 0
    else:
        if pp[(jdx-1)*leng+idx+1]+pp[(jdx-1)*leng+idx-1] == 0:#to avoid divdied by 0, so I put this before the function
            ans = 0
        else:
            ans = 0.5*(vv[(jdx-1)*leng+idx+1]+vv[(jdx-1)*leng+idx-1]) - const *
0.5*(vv[(jdx-1)*leng+idx+1]+vv[(jdx-1)*leng+idx-1]) * (vv[(jdx-1)*leng+idx+1]-vv[(jdx-1)*leng+idx-1]) - const *
(pp[(jdx-1)*leng+idx+1]-pp[(jdx-1)*leng+idx-1]) /(0.5*(pp[(jdx-1)*leng+idx+1]+pp[(jdx-1)*leng+idx-1]))
    return ans
```

(a). to find speed of sound and sound crossing time

```
def find_sound():#this function is to find sound speed and speed crossing time
    pp = []
    vv = []
    for jdx in range(101):
        dn = jdx * tau
        for idx in range(11):
            dh = idx * h
            resp = press(jdx, idx,pp,vv)
            resv = vol(jdx, idx,pp,vv)
            pp.append(resp)
            vv.append(resv)
        if resp != 0:
            break
    print(dn)
    sound_speed = 0.4/dn
    print(sound_speed)
    if resp == 0:
        print("it's not real answer")


find_sound()
```

and the result is:

sound crossing time = 0.004

speed of sound = 100.00

this means that in 1 time unit, the sound can reach 100 length units.

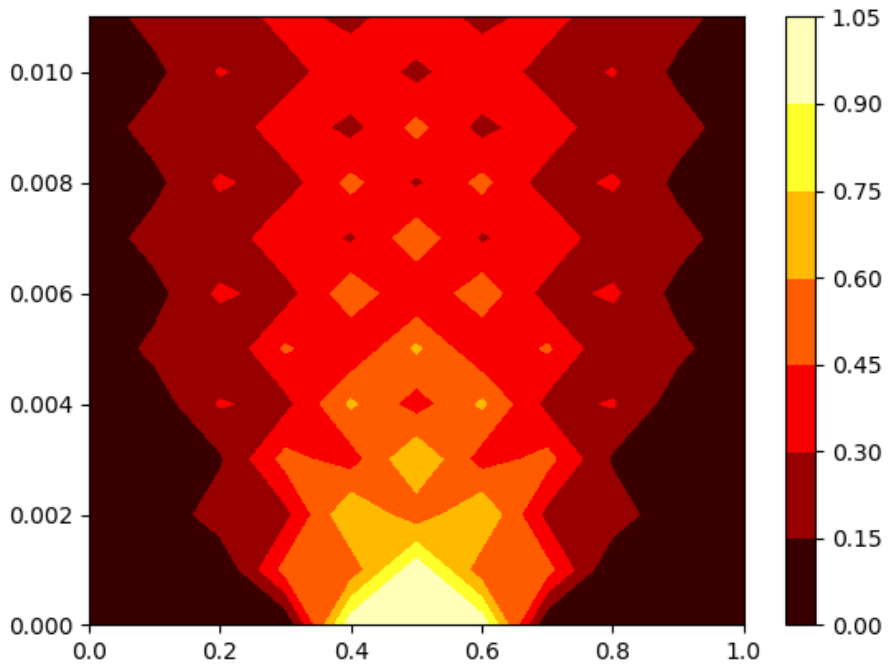
Analytic results is $\sqrt{\frac{\partial p}{\partial \rho}} = C_s$ = k = $1\left(\frac{\text{space span}}{\text{time span}}\right) = 1\left(\frac{0.1\ space\ unit}{0.001\ time\ unit}\right) = 100$

(b).solve the problem

Code:

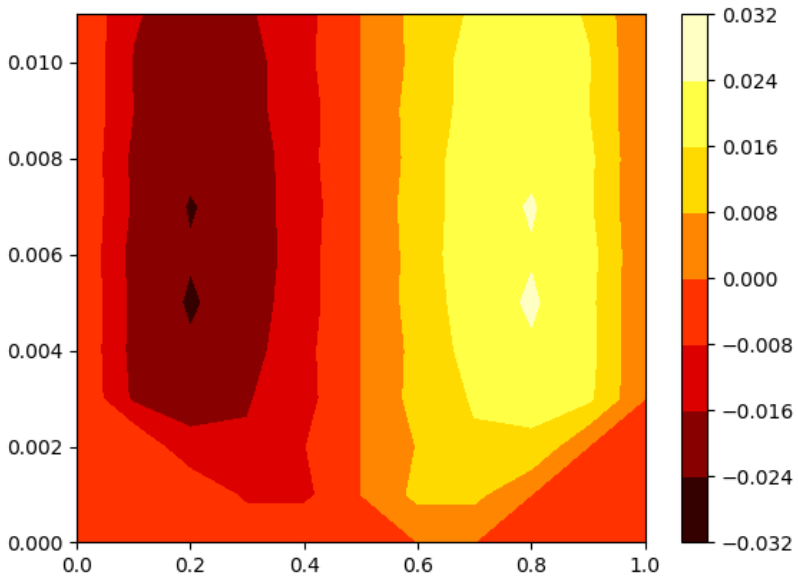
```
def test2():
    pp = []
    vv = []
    for jdx in range(12):#loop for time
        for idx in range(11):#loop for space
            resp = press(jdx, idx,pp,vv)
            resv = vol(jdx, idx,pp,vv)
            pp.append(resp)
            vv.append(resv)
    a = np.reshape(pp,(12,11))#reshape the list to 12*11
    b = np.reshape(vv,(12,11))
    dn = np.linspace(0,0.011,12)
    dh = np.linspace(0,1,11)
    X, T = np.meshgrid(dh, dn)#make a grid
    plt.contourf(X, T, a,cmap = plt.cm.hot)#plot contour
    plt.show()
    plt.contourf(X, T, b,cmap = plt.cm.hot)
    plt.show()

test2()
```

Result:



x-axis is x, y-axis is t, z-axis is p = ρ

we can see that the pressure is spread out from the center when time goes by. Because I use Lax method, we can see there some dots symmetric between both sides. Lax method try to solve the questions from previous time last step and next step, so there will be some unusual dots in the figure.



x-axis is x, y-axis is t, z-axis is v

This figure shows that the velocity is symmetry of center, left side velocity will go left, and right side velocity will go right. Also, when time goes by, the velocity become static compare to initial.

(3).

First to derive the analytic static solution

$$\frac{\partial \rho}{\partial t} = -\frac{\partial(\rho v)}{\partial x}$$

$$\frac{\partial v}{\partial t} = -v\frac{\partial v}{\partial x} - \frac{1}{\rho}\frac{\partial p}{\partial x} + \frac{f}{\rho}$$

In static condition, v = 0. Therefore, first equation will become this:

$$\frac{\partial \rho}{\partial t} = 0, \rho \; is \; a \; constant$$

And the second equation will become this:

$$0 = 0 - \frac{1}{\rho}\frac{\partial p}{\partial x} + \frac{f}{\rho}$$

$$f = \frac{\partial p}{\partial x}$$

$$p = fx + c \text{ , for } c = \text{constant}$$

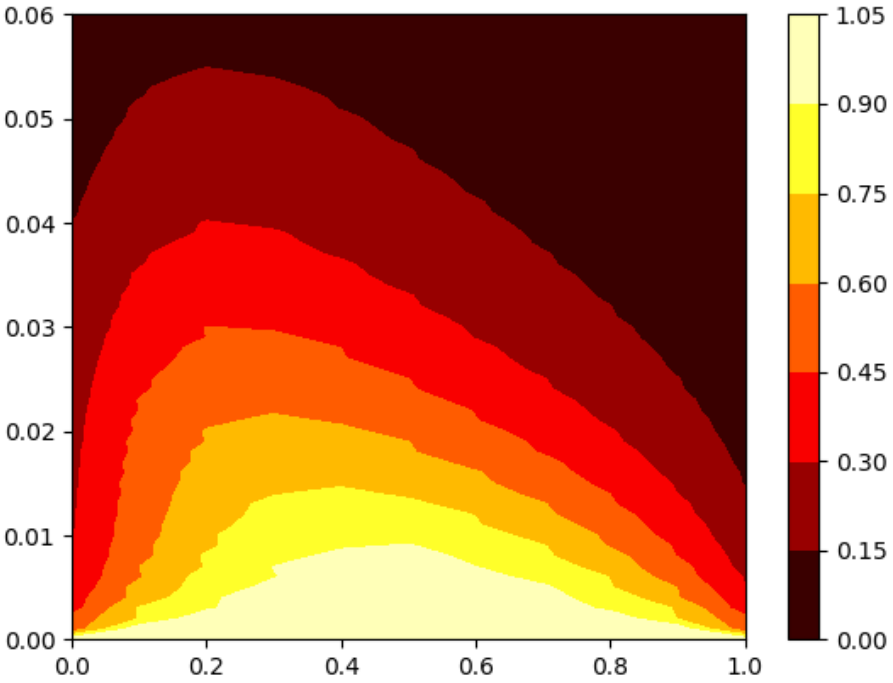
we suppose f = -10ρ

code:

```
import matplotlib.pyplot as plt
import numpy as np
tau = 0.01
h = 0.1
leng = int(1/h+1)

def press(jdx,idx,pp,vv):
    tt = jdx*tau
    xx = idx*h
    const = tau / (2 * h)
    if tt==0:#initial condition
        ans = 1
        return ans
    else:
        if xx == 0:#boundary condition
            ans = 0.5* (pp[(jdx - 1) * leng + idx + 1]) - const * 0.5 * (
                    pp[(jdx - 1) * leng + idx + 1] ) * (
                          vv[(jdx - 1) * leng + idx + 1]) - const * 0.5 * (
                          vv[(jdx - 1) * leng + idx + 1] ) * (
                          pp[(jdx - 1) * leng + idx + 1] )
        elif xx == 1:#boundary condition
            ans = 0.5 * (pp[(jdx - 1) * leng + idx - 1]) - const * 0.5 * (pp[(jdx - 1) * leng + idx - 1]) * (- vv[(jdx
- 1) * leng + idx - 1]) - const * 0.5 * (vv[(jdx - 1) * leng + idx - 1]) * ( - pp[(jdx - 1) * leng + idx - 1])
        else:
            ans = 0.5 * (pp[(jdx - 1) * leng + idx + 1] + pp[(jdx - 1) * leng + idx - 1]) - const * 0.5 * (
                    pp[(jdx - 1) * leng + idx + 1] + pp[(jdx - 1) * leng + idx - 1]) * (
                          vv[(jdx - 1) * leng + idx + 1] - vv[(jdx - 1) * leng + idx - 1]) - const * 0.5 * (
                          vv[(jdx - 1) * leng + idx + 1] + vv[(jdx - 1) * leng + idx - 1]) * (
                          pp[(jdx - 1) * leng + idx + 1] - pp[(jdx - 1) * leng + idx - 1])
        return ans

def vol(jdx,idx,pp,vv):
    tt = jdx * tau
    xx = idx * h
    const = tau / (2 * h)
    if tt==0:#initial condition
        ans =0
    elif xx==0 or xx==1:#boundary condition
        ans = 0
    else:
```

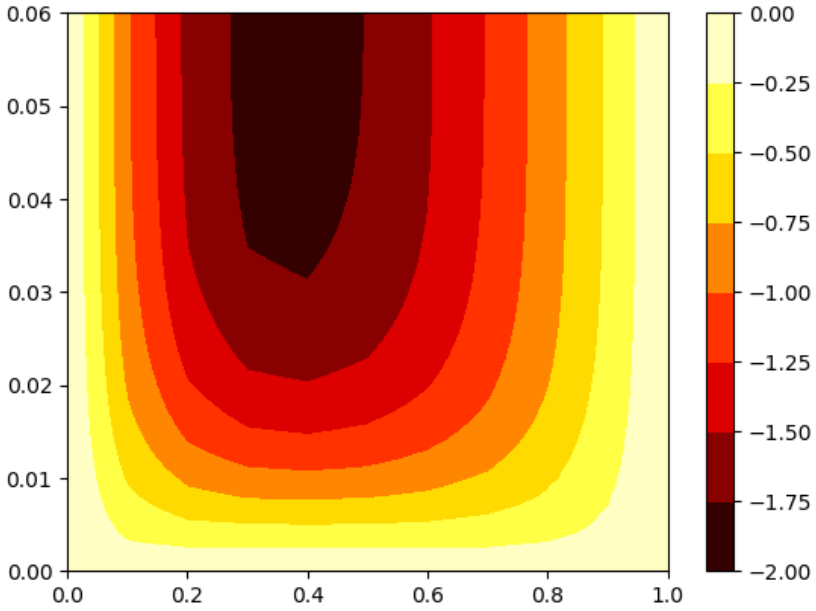
```
        if pp[(jdx-1)*leng+idx+1]+pp[(jdx-1)*leng+idx-1] == 0:
            ans = 0
        else:
            ans = 0.5*(vv[(jdx-1)*leng+idx+1]+vv[(jdx-1)*leng+idx-1]) - const *
0.5*(vv[(jdx-1)*leng+idx+1]+vv[(jdx-1)*leng+idx-1]) * (vv[(jdx-1)*leng+idx+1]-vv[(jdx-1)*leng+idx-1]) - const *
(pp[(jdx-1)*leng+idx+1]-pp[(jdx-1)*leng+idx-1]) /(0.5*(pp[(jdx-1)*leng+idx+1]+pp[(jdx-1)*leng+idx-1]))-10*tau
    return ans

def test():
    pp = []
    vv = []
    for jdx in range(61):
        for idx in range(11):
            resp = press(jdx, idx,pp,vv)
            resv = vol(jdx, idx,pp,vv)
            pp.append(resp)
            vv.append(resv)
    a = np.reshape(pp, (61, 11))
    b = np.reshape(vv, (61, 11))
    dn = np.linspace(0, 0.06, 61)
    dh = np.linspace(0, 1, 11)
    X, T = np.meshgrid(dh, dn)
    aa = plt.contourf(X, T, a, cmap=plt.cm.hot)
    plt.colorbar(aa, orientation="vertical")
    plt.show()
    bb = plt.contourf(X, T, b, cmap=plt.cm.hot)
    plt.colorbar(bb, orientation="vertical")
    plt.show()
test()
```

Results:



x-axis is x, y-axis is t, z-axis is ρ

We can see that it still spread out, but not from center. The most high value keep going left when time goes by, and it shows that f = -10ρ this statement makes gas keep going left(down).



x-axis is x, y-axis is t, z-axis is v

At first, there no velocity, but when time goes by, the velocity become stronger, and more left(down) too. That's because we add accelerate in the system, so the velocity becomes stronger, and also because accelerate is on left, so the velocity is on left too.