AMath 574
Homework 2
Updated 1/21/23 to include final problem
Due by 11:00pm on January 26, 2023

For submission instructions, see:

`http://faculty.washington.edu/rjl/classes/am574w2023/homework2.html`

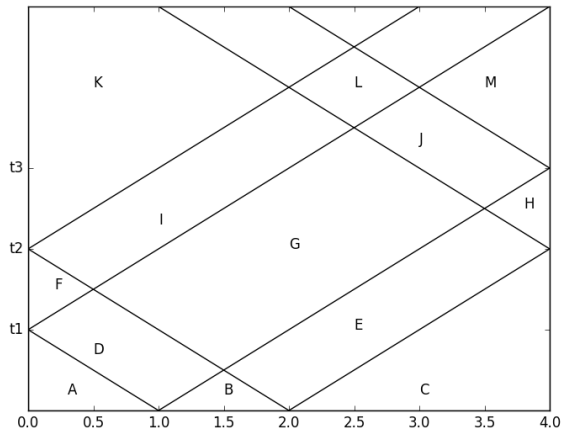
Problem #3.4 in the book

Also sketch the solution in the x–t plane, labeling the states that appear in the solution. Then show where these states are in the p–u phase plane and how they are connected by eigenvectors.

Note that the script `problem_3_5.py` was used to generate the figure for the next problem and could be simplified for this problem if desired.

Problem #3.5 in the book

The script `problem_3_5.py` was used to generate this figure:



To solve this problem, determine the states $A,\ B,\ \ldots,\ M$ and also the times $t_1,\ t_2,\ t_3$. The times can be written in terms of the parameters ρ_0 and K_0, which were not stated in the problem.

For example,

$$A = \begin{bmatrix} 0 \\ 0 \end{bmatrix}, \quad B = \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \quad C = \begin{bmatrix} 0 \\ 0 \end{bmatrix}, \quad \ldots$$

Hint: Another way to think about what happens when waves reflect off the boundary is suggested in Section 7.3.3.

Problem #3.8 in the book

Problem #4.2 in the book

Problem #4.3 in the book

Programming Problem.

The notebook in the class repository at `$AM574/notebooks/acoustics_godunov.ipynb` has been updated since class on 1/20 to make it a bit easier to modify for this problem. Please make sure you "git pull" to get the latest version. And please contact me if you are having problems with Jupyter of Python! We will do more with this same notebook in future homeworks.

(a) Implement a new function `LxF_step` similar to `Godunov_step` that takes a single step with the Lax-Friedrichs methods (for the same acoustics system):

$$Q_i^{n+1} = Q_i^n - \frac{\Delta t}{2\Delta x} A(Q_{i+1}^n - Q_{i-1}^n)$$

Since `Pn` and `Un` are separate arrays, you will have to implement the matrix-vector products implied in Lax-Friedrichs componentwise.

You might want to test with Courant number 1 first when debugging.

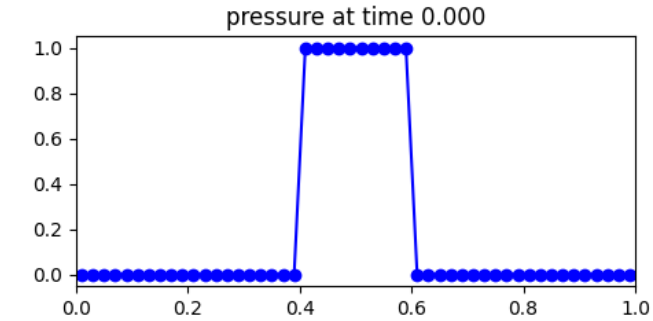
Submit your notebook that contains the implementation and also does at least the experiment described in part (b) below. You can include other tests too if you want.

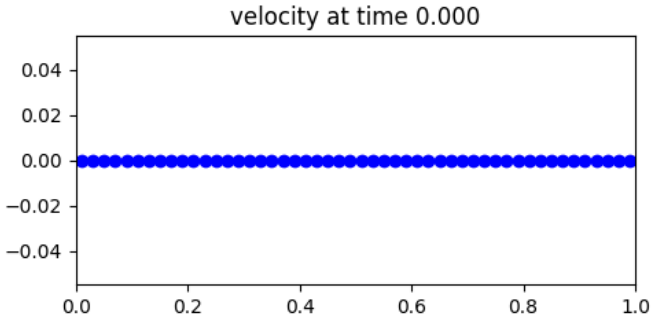
(b) When you do the following experiment:

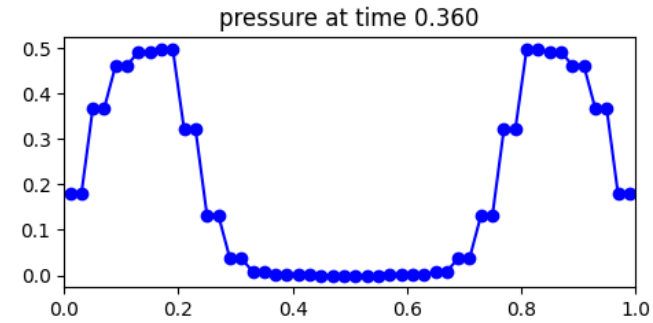
In [55]:
```
num_cells = 50
dx = (xupper - xlower)/num_cells
x = arange(xlower-dx/2, xupper+dx, dx)
print('including 2 ghost cells, the grid has %i cells' % len(x))

P0 = where(logical_and(x>0.4,x<0.6), 1., 0.)
U0 = zeros(x.shape)

t0 = 0.
fig = figure(figsize=(5,5))
plotQ(P0,U0,t0)
savefig('plot1.png', bbox_inches='tight')

dt = 0.018
cfl = c0*dt/dx
nsteps = 20
tn = t0 + nsteps*dt
print('Using dt = %.4f, cfl = %.2f, taking %i steps to time %.3f' \
      % (dt,cfl,nsteps,tn))

Pn,Un = time_stepper(t0, P0, U0, dt, nsteps, LxF_step)
fig = figure(figsize=(5,5))
plotQ(Pn,Un,tn)
savefig('plot2.png', bbox_inches='tight')
```

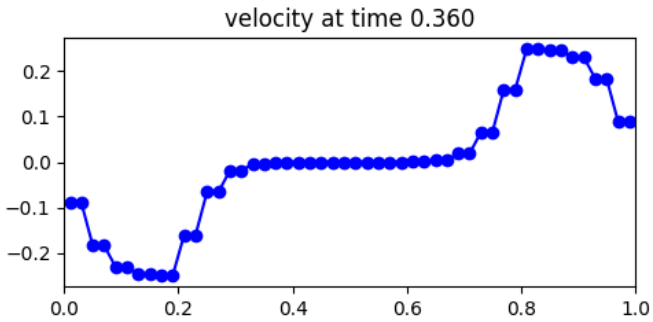
```
including 2 ghost cells, the grid has 52 cells
Using dt = 0.0180, cfl = 0.90, taking 20 steps to time 0.360
```

you should get plots like theese:









Explain why there are always two adjacent points with exactly the same value, for this particular choice of initial data.

Note that that I inserted `savefig` commands in the input to create the png files used here.

See `https://matplotlib.org/stable/api/_as_gen/matplotlib.pyplot.savefig.html` for more about this command.