02_Step_2

January 5, 2016

[@LorenaABarba](https://twitter.com/LorenaABarba)

1 12 steps to Navier-Stokes

This IPython notebook continues the presentation of the **12 steps to Navier-Stokes**, the practical module taught in the interactive CFD class of Prof. Lorena Barba. You should have completed Step 1 before continuing, having written your own Python script or notebook and having experimented with varying the parameters of the discretization and observing what happens.

1.1 Step 2: Non-linear Convection

Now we're going to implement non-linear convection using the same methods as in step 1. The 1D convection equation is:

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

Instead of a constant factor c multiplying the second term, now we have the solution u multiplying it. Thus, the second term of the equation is now <u>non-linear</u> We're going to use the same discretization as in Step 1 — forward difference in time and backward difference in space. Here is the discretized equation.

$$\frac{u_i^{n+1}-u_i^n}{\Delta t} + u_i^n \frac{u_i^n-u_{i-1}^n}{\Delta x} = 0$$

Solving for the only unknown term, u_i^{n+1}, yields:

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

As before, the Python code starts by loading the necessary libraries. Then, we declare some variables that determine the discretization in space and time (you should experiment by changing these parameters to see what happens). Then, we create the initial condition u_0 by initializing the array for the solution using $u = 2$ @ $0.5 \leq x \leq 1$ and $u = 1$ everywhere else in $(0, 2)$ (i.e., a hat function).

```
In [1]: import numpy as np                  #we're importing numpy and calling it np locally
        import matplotlib.pyplot as plt     #and our 2D plotting library, calling it plt
        %matplotlib inline


        nx = 41
        dx = 2./(nx-1)
        nt = 20    #nt is the number of timesteps we want to calculate
```

```
dt = .025  #dt is the amount of time each timestep covers (delta t)

u = np.ones(nx)      #as before, we initialize u with every value equal to 1.
u[.5/dx : 1/dx+1]=2  #then set u = 2 between 0.5 and 1 as per our I.C.s

un = np.ones(nx) #initialize our placeholder array un, to hold the time-stepped solution
```

The code snippet below is unfinished. We have copied over the line from Step 1 that executes the time-stepping update. Can you edit this code to execute the non-linear convection instead?

```
In [ ]: for n in range(nt):  #iterate through time
            un = u.copy() ##copy the existing values of u into un
            for i in range(1,nx):  ##now we'll iterate through the u array

             ###This is the line from Step 1, copied exactly.  Edit it for our new equation.
             ###then uncomment it and run the cell to evaluate Step 2

                   ###u[i] = un[i]-c*dt/dx*(un[i]-un[i-1])


        plt.plot(np.linspace(0,2,nx),u) ##Plot the results
```

What do you observe about the evolution of the hat function under the non-linear convection equation? What happens when you change the numerical parameters and run again?

1.2 Learn More

For a careful walk-through of the discretization of the convection equation with finite differences (and all steps from 1 to 4), watch **Video Lesson 4** by Prof. Barba on YouTube.

```
In [2]: from IPython.display import YouTubeVideo
        YouTubeVideo('y2WaK7_iMRI')

Out[2]: <IPython.lib.display.YouTubeVideo at 0x7fa86dd4ce90>

In [3]: from IPython.core.display import HTML
        def css_styling():
            styles = open("../styles/custom.css", "r").read()
            return HTML(styles)
        css_styling()

Out[3]: <IPython.core.display.HTML at 0x7fa88537b390>
```

(The cell above executes the style for this notebook.)