1 Results: The Code

At this point, the Navier-Stokes equations have been presented, as well as tools for discretizing those equations. Next we will go over the components of the code that make solving the Navier-Stokes approximate equations. An aspect of the code that needs to be established in addition to the approximate equations is the setup of the system; what are the initial conditions, boundary conditions, and what does the grid look like?

1.1 Setting up the System

The simulation is of fluid flow in a pipe, therefore the shape of the domain will be rectangular. Let the domain span from $x = 0$ to $x = 2$ in the x direction, and from $y = 0$ to $y = 2$ in the y direction. Now that the shape and domain of our pipe has been established, now we must discretize the domain into a set of finite points. To simplify the code overall, the numerical grid will be regular: divide the rectangular domain by a set of equally spaced vertical lines and a set of equally spaced horizontal lines. Let the number of vertical lines dividing the space be nx, and the number of horizontal lines be ny. All of the variable names in this section correspond to the variables used in the python code below. The spacing in between points in the x direction on the grid will equal the length of the domain in the x direction, divided by the number of lines dividing the space: $dx = \frac{2}{nx}$. Similarily, the spacing in between points in the y direction on the grid will be $dy = \frac{2}{ny}$.

```
nx = 101
ny = 101

dx = 2.0/(nx-1)
dy = 2.0/(ny-1)

x = numpy.linspace(0,2,nx)
y = numpy.linspace(0,2,ny)
```

In the final two lines of code above, the variables x and y are each set equal to an array spanning from 0 to 2, with nx and ny entries respectively. Next, we should set the values for the density **rho**, dynamic viscosity **mu**, kinematic viscosity **nu**, and the length of timesteps in the simulation, **dt**.

```
rho = 0.125
mu = 0.001
nu = mu/rho
dt = 0.001
```

The values of **rho**, **mu**, and **nu** are determined by the desired Reynolds number for the simulation, rather than by experimental values. The Reynolds number is a dimensionless parameter that characterizes the type of flow, laminar, turbulent or something in between. The Reynolds number takes into account the flow velocity, density, viscosity, and characteristic length and gives a general idea of what the flow should be like.

$$Re = \frac{\rho v L}{\mu} \tag{1}$$

The flow through the pipe is driven by a pressure differential between the two ends of the pipe. The pressure differential will be the variable **F**, and is constant throughout the simulation, so we will set that value now.

```
F = 2
```

The stability of the program depends on the magnitude of the pressure differential. The value of **F** can be thought of as a factor that scales up the velocity of the flow. The resolution of the numerical grid must be increased as the pressure differential is increased. Since the flow is driven by this pressure differential, the initial velocity throughout the pipe will be zero.

```
u = numpy.zeros((ny,nx))
v = numpy.zeros((ny,nx))
p = numpy.zeros((ny,nx))
```

In the code above, **u** is a 2-dimensional array representing the x-component of the velocity vector field throughout the pipe, and **v** is a 2-dimensional array representing the y-component of the velocity. Above, **p** is the 2-dimensional array representing the scalar field for pressure throughout the pipe, which is also initially zero. One last variable we need to declare before running the code is **nt**, the number of time steps we would like the system to run for. This is all of the code needed to setup the initial condition of the system.

1.2 Implementing Finite Differences

1.3 Stepping Through the Program