Physics 566: Computational Physics Final Project

Modeling Channel Flow with the Navier-Stokes Equations

Varun Nair

April 16, 2019

```
In [1]: %precision %g
        %matplotlib inline
        %config InlineBackend.figure_format = 'retina'

        from math import sqrt, pi, sin, cos, floor, exp
        import numpy as np
        from numpy import linalg as LA
        from scipy import constants as con
        import matplotlib.pyplot as plt

        from channel_flow import solver, solved_u
```

1 Theory

1.1 Introduction

The Navier-Stokes equations (NSE) can only be solved analytically for a few systems. One of these is for steady-state, fully developed laminar flow in a single direction between two plates. Laminar flow occurs when fluids move in parallel layers, so that movement in one layer doesn't disturb movement in another layer. The equation of motion for this system can be written as

$$\frac{\partial \mathbf{u}}{\partial t}+(\mathbf{u}\cdot\nabla)\mathbf{u} = \nu\Delta\mathbf{u} - \frac{1}{\rho}\nabla P, \tag{1}$$

where $\mathbf{u}$ is the fluid velocity vector, $\nu = \frac{\text{viscosity}}{\text{density}}$ is the kinematic viscosity, and $P \equiv p + \Phi$. The fluid is propelled through the channel by a force $F = -\nabla\Phi$. This can be rewritten in terms of fluid density (ρ) and shear viscosity (η), giving the partial differential equation

$$\rho\frac{\partial \mathbf{u}}{\partial t} + \rho(\mathbf{u}\cdot\nabla)\mathbf{u} = -\nabla p + \eta\Delta\mathbf{u}, \tag{2}$$

where $\mathbf{u}$ is again the velocity vector and $P = P(x,t)$ is the pressure as a function of position and time. For incompressible fluid flow, I can get a Poisson equation for pressure by setting the divergence of the momentum equation zero (this ends up taking the divergence of velocity to be zero because mass is constant for incompressible fluids) and assuming the pressure must be continuous over the region.

$$\nabla \cdot \mathbf{u} = 0 \tag{3}$$

$$\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} = 0 \tag{4}$$

These conditions allow us to write three equations that govern the fluid flow through the channel. From now on, I use u and v to designate the velocities in the x and y directions, respectively. ρ and ν are still the fluid density and viscosity, respectively.

$$\frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} + v\frac{\partial u}{\partial y} = -\frac{1}{\rho}\frac{\partial p}{\partial x} + \nu\left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}\right)$$

$$\frac{\partial v}{\partial t} + u\frac{\partial v}{\partial x} + v\frac{\partial v}{\partial y} = -\frac{1}{\rho}\frac{\partial p}{\partial y} + \nu\left(\frac{\partial^2 v}{\partial x^2} + \frac{\partial^2 v}{\partial y^2}\right)$$

$$-\frac{1}{\rho}\left(\frac{\partial^2 p}{\partial x^2} + \frac{\partial^2 p}{\partial y^2}\right) = \left(\frac{\partial u}{\partial x}\right)^2 + 2\frac{\partial u}{\partial y}\frac{\partial v}{\partial x} + \left(\frac{\partial v}{\partial y}\right)^2$$

I now have a system of 2nd order, nonlinear partial differential equations that fully describe a fluid's behavior in a channel. However, these equations are only analytically solvable for a closed form solution under the following conditions: - laminar flow - steady flow - incompressible fluid - 2-D geometry - between two smooth surfaces of constant width.

1.2 More Complicated Flows

I'll use computational physics to numerically solve these equations when assumptions are relaxed. First I will relax the steady flow assumption for fluid velocity in the x direction by writing pressure in terms of steady and unsteady components. I write the total pressure

$$p = p_0 + p_1$$

as the sum of steady (p_0) and unsteady (p_1) pressures. Previously, $p_1 = 0$, but if it varied with time, it would alter the flow.

$$\frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} + v\frac{\partial u}{\partial y} = -\frac{1}{\rho}\left(\frac{\partial p_0}{\partial x} + \frac{\partial p_1}{\partial x}\right) + \nu\left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}\right) \tag{5}$$

$$\frac{\partial v}{\partial t} + u\frac{\partial v}{\partial x} + v\frac{\partial v}{\partial y} = -\frac{1}{\rho}\frac{\partial p}{\partial y} + \nu\left(\frac{\partial^2 v}{\partial x^2} + \frac{\partial^2 v}{\partial y^2}\right) \tag{6}$$

$$-\frac{1}{\rho}\left(\frac{\partial^2 p}{\partial x^2} + \frac{\partial^2 p}{\partial y^2}\right) = \left(\frac{\partial u}{\partial x}\right)^2 + 2\frac{\partial u}{\partial y}\frac{\partial v}{\partial x} + \left(\frac{\partial v}{\partial y}\right)^2 \tag{7}$$

1.3 Initial/Boundary Conditions

The initial conditions I use for the different cases remain the same. The initial velocity of the fluid at all points in the channel (for both x and y directions is zero). The fluid is also at equal pressure throughout the region.

$$u(x,y,0) = v(x,y,0) = 0$$
$$p(x,y,0) = 5$$

The boundary conditions for the most basic case are that fluid velocity along the upper and lower boundary is zero due to the no slip condition. Further the first derivative of pressure is zero.

$$u(x,0,t) = u(x,\tfrac{L}{2},t) = v(x,0,t) = v(x,\tfrac{L}{2},t) = 0 \tag{8a}$$
$$\left.\frac{\partial p}{\partial y}\right|_{y=0} = \left.\frac{\partial p}{\partial y}\right|_{y=\frac{L}{2}} = 0 \tag{8b}$$

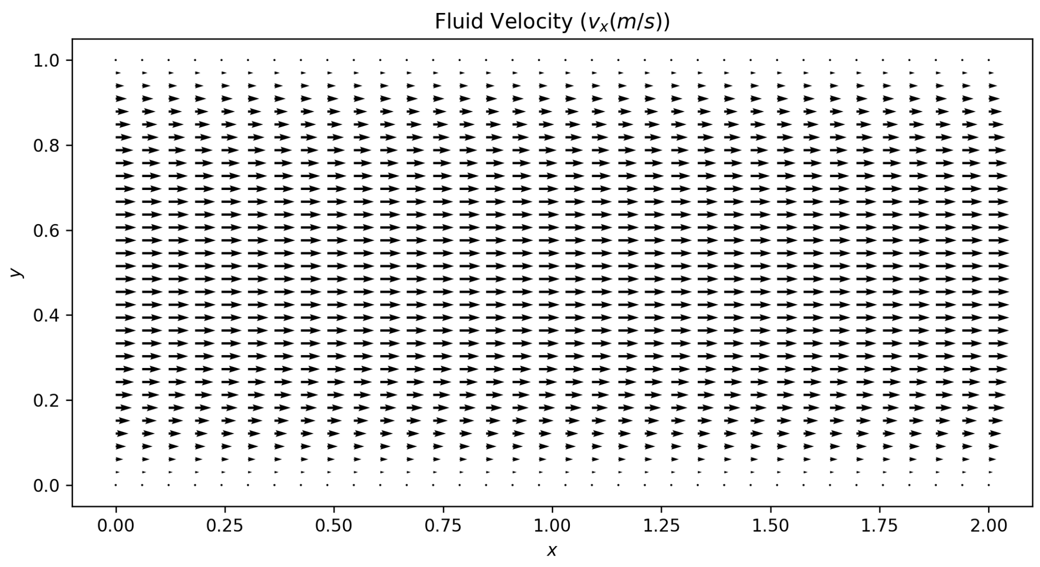
1.4 Modelling the Flow

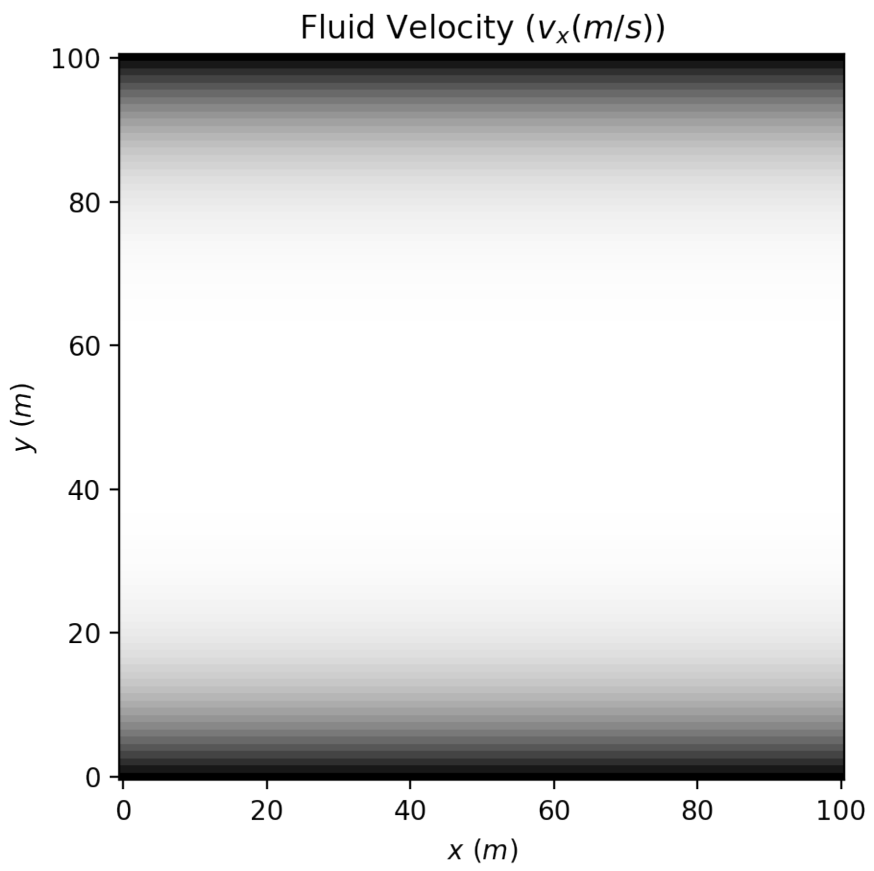
Given this, we can solve for the velocity of the fluid throughout the region. I'll write out the expression for v_x of the fluid on the interior of the region explicitly as

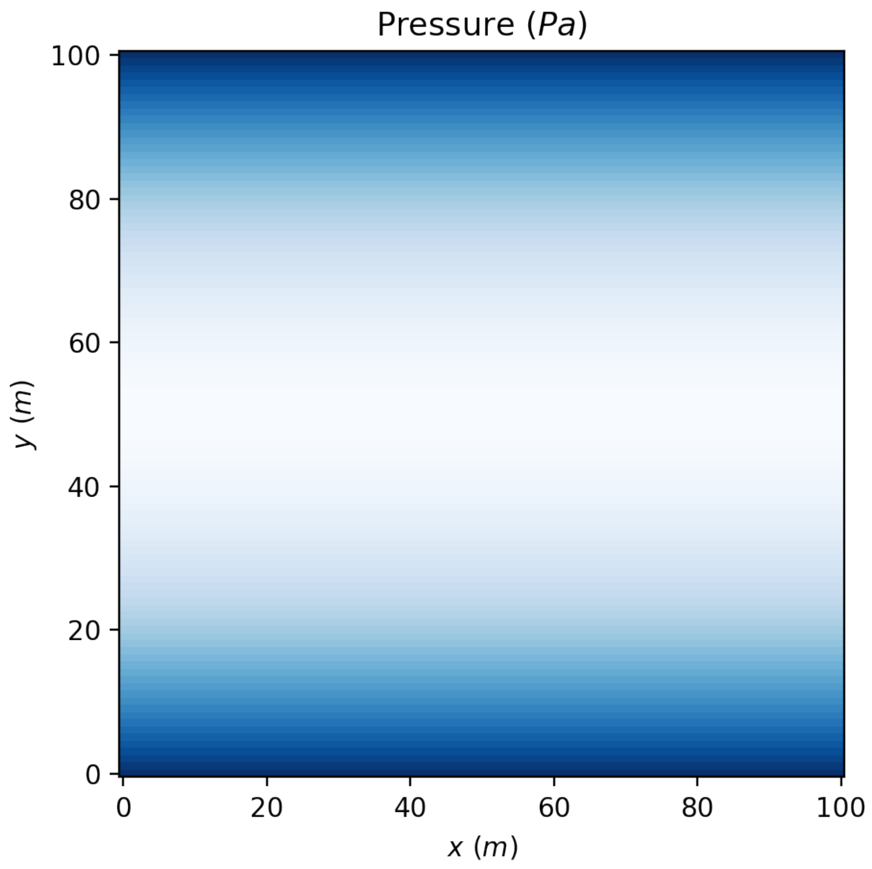
$$\begin{aligned} u(x,y) \leftarrow u(x,y) &- \frac{u(x,y)\mathrm{d}t}{a}\Big[u(x,y) - u(x-a,y)\Big] \\ &- \frac{v(x,y)\mathrm{d}t}{a}\Big[u(x,y) - u(x,y-a)\Big] - \frac{\mathrm{d}t}{2\rho a}\Big[p(x+a,y) - p(x-a,y)\Big] \\ &+ \nu\left[\frac{\mathrm{d}t}{a^2}\Big(\big(u(x+a,y) - 2u(x,y) + u(x-a,y)\big) + \big(u(x,y+a) - 2u(x,y) + u(x,y-a)\big)\Big)\right] \\ &+ P\mathrm{d}t \end{aligned} \tag{9}$$

```
In [2]: tend = 0.5
        N = 100
        L = 1
        res = 100 #image resolution

        #initiates quiver plot
        x = np.linspace(0, L, N)
        y = np.linspace(0, L/2, N)
        X, Y = np.meshgrid(x, y)

        u1,v1,p1 = solver(tend,N,L)#,obstacle='ball')

In [3]: fig1, ax1 = plt.subplots(1, 1, figsize = (10, 5),
                                 dpi=res)

        a = 2 #frequency of arrows
        ax1.quiver(X[::a, ::a], Y[::a, ::a],
                   u1[::a, ::a], v1[::a, ::a])
        ax1.set_title("Fluid Velocity ($v_x (m/s)$)")
        ax1.set_xlabel("$x\ (m)$")
        ax1.set_ylabel("$y\ (m)$")
        #plt.savefig("{:3.0f}_basic_quiver.png".format(N))


        fig2, ax2 = plt.subplots(1, 1, figsize = (10, 5),
                                 dpi=res)

        ax2.imshow(u1,origin='lower',cmap='gray')
        ax2.set_title("Fluid Velocity ($v_x (m/s)$)")
        ax2.set_xlabel("$x\ (m)$")
        ax2.set_ylabel("$y\ (m)$")
        #plt.savefig("{:3.0f}_basic_velocity.png".format(N))


        fig3, ax3 = plt.subplots(1, 1, figsize = (10, 5),
                                 dpi=res)

        ax3.imshow(p1,origin='lower',cmap='Blues')
        ax3.set_title("Pressure ($Pa$)")
        ax3.set_xlabel("$x\ (m)$")
        ax3.set_ylabel("$y\ (m)$")
        #plt.savefig("{:3.0f}_basic_pressure.png".format(N))
```

Out[3]:
```
<matplotlib.text.Text at 0x11be6d668>
```




Fluid Velocity ($v_x(m/s)$)
y (m)
x (m)
0
20
40
60
80
100



1.5 Testing Results

For the most basic case of channel flow, I can compare my results to the results predicted by the analytically solved equation for the fluid's velocity in the x direction. The equation for velcoity before applying the relevant boundary conditions is

$$u(y) = -\frac{10}{\nu}\left(\frac{y^2}{2} + C_1 y + C_2\right).$$

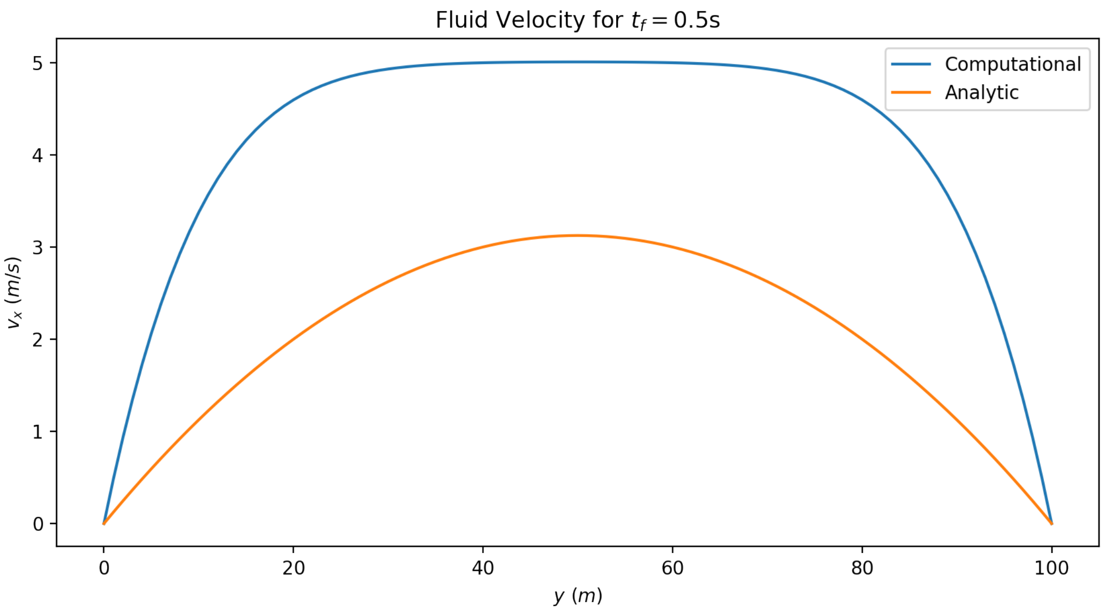
When we apply the no slip conditions at $y = 0, \frac{L}{2}$, which force fluid velocity to be zero, we constrain the equation such that $C_2 = 0$ and $C_1 = -\frac{L}{4}$. This gives us the equation for velocity as a function of height in a channel as

$$u(y) = -\frac{10}{\nu}\left(\frac{y^2}{2} - \frac{L}{4}y\right). \tag{10}$$

```
In [4]: #compare to analytical solution

        #array with analytically determined velocity
        yrange = np.linspace(0,L/2,N+1)
```

```
perfect_u = solved_u(yrange)

fig4, ax4 = plt.subplots(1, 1, figsize = (10, 5),
                         dpi=res)

ax4.plot(u1[:,int(L/2)],label='Computational')
ax4.plot(perfect_u,label='Analytic')
ax4.set_title("Fluid Velocity for $t_f={:1.1f}$s"\
              .format(tend))
ax4.set_xlabel("$y\ (m)$")
ax4.set_ylabel("$v_x\ (m/s)$")
ax4.legend()
#plt.savefig("100_basic_compare.png")
```

```
Out[4]: <matplotlib.legend.Legend at 0x11d3d0a20>
```



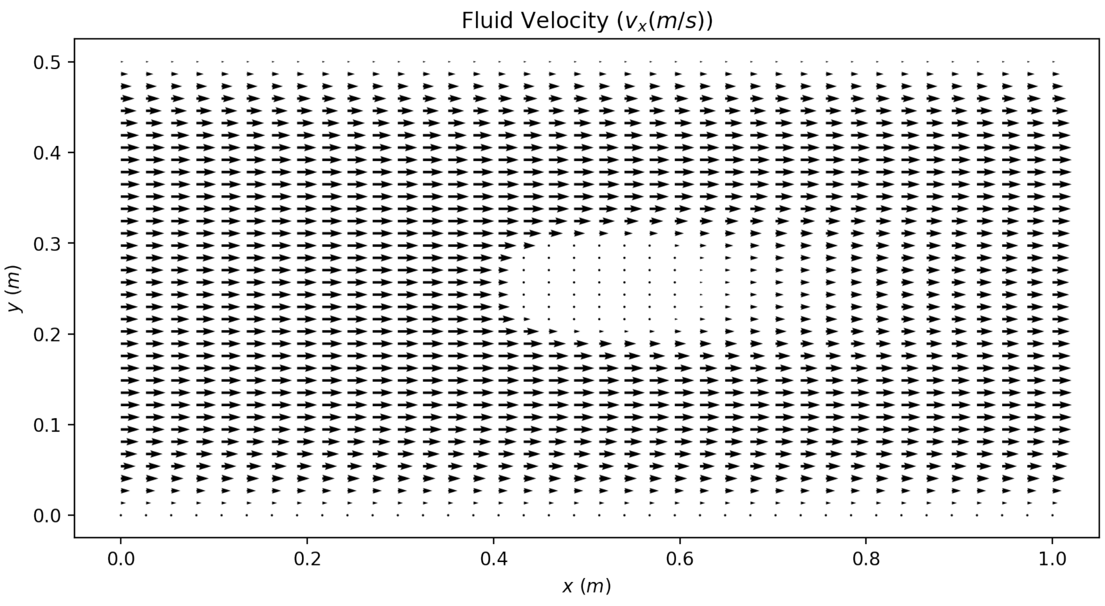
1.6 Ball in Channel

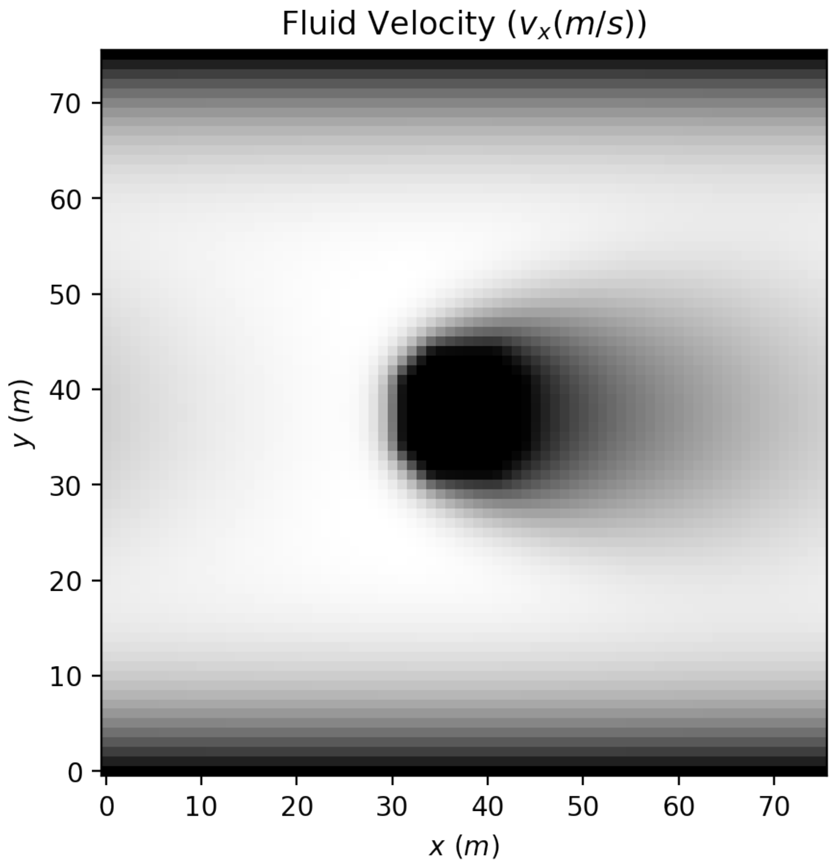
I'll now analyze the impact of placing a ball into a channel with this flow. To ease the computational expense, I will model it as a circle in a 2-d channel, which will give us a cross-section of the channel's flow. The equations modelling the flow remain the same. The difference is that we now add additional boundary conditions around the circle. These boundary conditions are the same as for the upper/lower channel boudnaries, i.e., the no slip condition and the unchanging pressure condition. I developed the code so that the optional argument "obstacle='ball'" triggers the additional boundary conditions to be taken into account when solving the system.

```
In [5]: tend = 0.5
        N = 75
        L = 1

        #initiates quiver plot
        x = np.linspace(0, L, N)
        y = np.linspace(0, L/2, N)
        X, Y = np.meshgrid(x, y)

        u2,v2,p2 = solver(tend,N,L,obstacle='ball')
```

```
In [6]: fig5, ax5 = plt.subplots(1, 1, figsize = (10, 5),
                                 dpi=res)

        a = 2 #frequency of arrows
        ax5.quiver(X[::a, ::a], Y[::a, ::a],
                   u2[::a, ::a], v2[::a, ::a])
        ax5.set_title("Fluid Velocity ($v_x (m/s)$)")
        ax5.set_xlabel("$x\ (m)$")
        ax5.set_ylabel("$y\ (m)$")
        #plt.savefig("{:2.0f}_ball_quiver.png".format(N))


        fig6, ax6 = plt.subplots(1, 1, figsize = (10, 5),
                                 dpi=res)

        ax6.imshow(u2,origin='lower',cmap='gray')
        ax6.set_title("Fluid Velocity ($v_x (m/s)$)")
        ax6.set_xlabel("$x\ (m)$")
        ax6.set_ylabel("$y\ (m)$")
        #plt.savefig("{:2.0f}_ball_velocity.png".format(N))
```

```
Out[6]: <matplotlib.text.Text at 0x11c6a9d30>
```





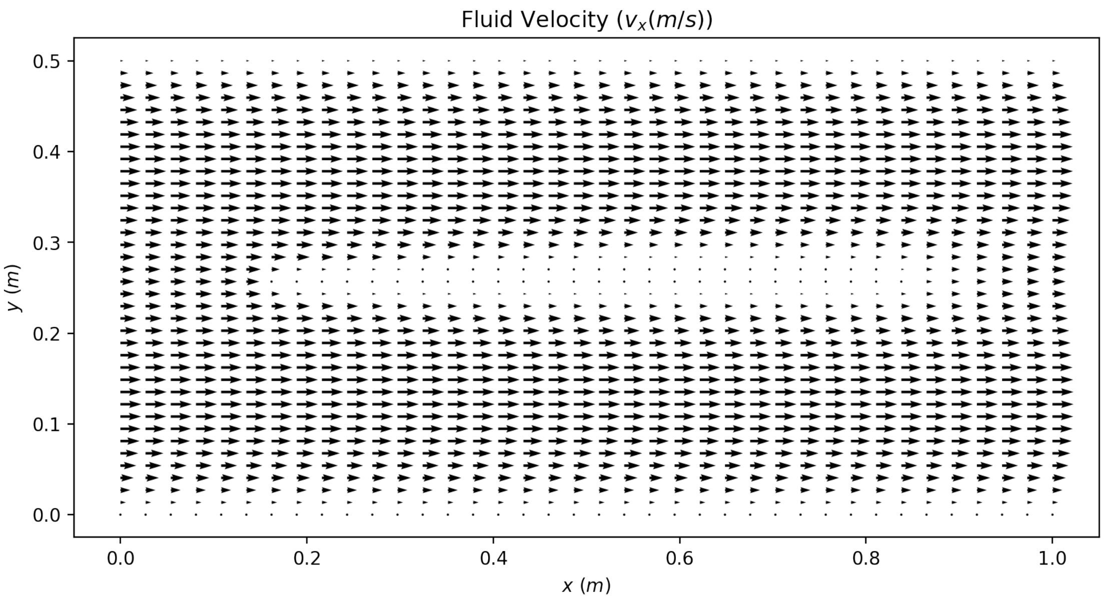
1.7 Flow Around a Wing

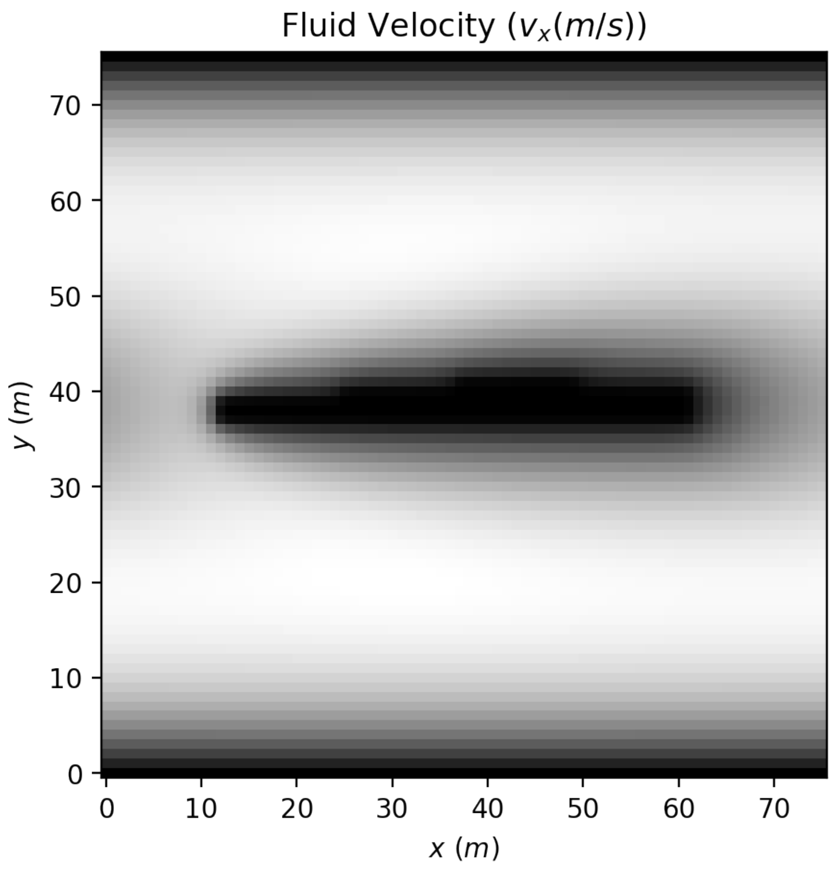
We can also create a wing-shaped object in the channel, modelling its effect on the flow of fuid. I've done this below, with the boundary conditions adjusted to the new object's shape.

```
In [7]: tend = 0.5
        N = 75
        L = 1

        #initiates quiver plot
        x = np.linspace(0, L, N)
        y = np.linspace(0, L/2, N)
        X, Y = np.meshgrid(x, y)

        u3,v3,p3 = solver(tend,N,L,obstacle='wing')

In [8]: fig7, ax7 = plt.subplots(1, 1, figsize = (10, 5),
                                 dpi=res)
```

```
a = 2 #frequency of arrows
ax7.quiver(X[::a, ::a], Y[::a, ::a],
           u3[::a, ::a], v3[::a, ::a])
ax7.set_title("Fluid Velocity ($v_x (m/s)$)")
ax7.set_xlabel("$x\ (m)$")
ax7.set_ylabel("$y\ (m)$")
#plt.savefig("{:2.0f}_wing_quiver.png".format(N))


fig8, ax8 = plt.subplots(1, 1, figsize = (10, 5),
                         dpi=res)

ax8.imshow(u3,origin='lower',cmap='gray')
ax8.set_title("Fluid Velocity ($v_x (m/s)$)")
ax8.set_xlabel("$x\ (m)$")
ax8.set_ylabel("$y\ (m)$")
#plt.savefig("{:2.0f}_wing_velocity.png".format(N))
```

```
Out[8]: <matplotlib.text.Text at 0x11c131198>
```





```
In [ ]:
```