Lab 4: Particle Image Velocimetry in a Water Tunnel

Date of experiment: October 20, 2023

Date submitted: November 3, 2023

Gracynne Apke, Mahala Peters, Harrison Stoelting, Daniel Stone

# Abstract

**Using a water tunnel and a seeding solution, particle image velocimetry can be used to observe trends in velocity flow. Multiple test articles were used in order to observe behaviors for different shapes and objects. The goal was to analyze the drag for each article and compare their behaviors. In the end, each article was observed to obscure the flow directly behind it, and vector and velocity graphs were composed for each test article so that the physical affects could be visually examined. It is clearly shown in each graph where the velocity slowed and where the velocity increased due to the geometry of each test article.**

# Nomenclature

wake half-width

drag coefficient, *D/(d*)

total drag coefficient, (form drag + viscous drag)/(*d*)

corrected total drag coefficient, (form drag + viscous drag)/(*d*)

lift coefficient, *L/(d)*

pressure coefficient, *(p - ) /*

*d*  cylinder diameter

*ds* differential surface element

*D*  drag force per unit span

*g*  local gravitational acceleration

gravitational constant, 32.174 lbm-ft/lbf-

*h*  measured pressure in inches of water column

unit vector in streamwise direction

unit vector normal to streamwise direction

*l*  characteristic dimension

*L* lift force per unit span

unit vector normal to the cylinder surface

*p* pressure

*q* dynamic pressure, ()/2

Reynolds number based on some characteristic length, *Ul/ν*

*U*  fluid velocity

*x* non-dimensional streamwise direction

*y* non-dimensional normal direction to cylinder chord

density of fluid

kinematic viscosity of fluid

# Introduction

Accurately gathering complex aerodynamic data has historically been a challenge for experimentalists. Pitot probes, pressure taps, sensors, and thermocouples can collect accurate data at a single point, but expanding the data to reflect the entirety of the object geometry becomes exceedingly complicated. However, advancements in camera and laser technology have led to the development of image based computation. Particle image velocimetry (PIV) is used to dramatically reduce the time needed to compute large volumes of aerodynamic data; because of this, entire vector fields can be mapped in a matter of seconds, not hours. In this lab, PIV will be used to collect velocity and turbulence data of fluid flow as it passes around various geometries. The lab will be conducted in a water tunnel and will use seed particles and laser sheets to quickly and accurately map 2D vector fields.

# Theory and Background

Particle image velocimetry works by using a camera to map the position of seed particles within a fluid. A short time later (Dt, determined by the shutter speed), the positions are mapped again, and the particles’ velocities are calculated by dividing the change in positions over the change in time (V = Dx/Dt). This process is repeated many times until an accurate average velocity can be collected for each point. A laser sheet is used so that the camera only views the particles in the 2D plane. A computer system is used to control the camera and laser, ensuring that the correct number of images are collected and that a suitable field of view is achieved.

After the images are collected, a software program is used to mask shadows and irrelevant data and compute the velocities. Since the camera collects pixel data and not distance data, a scale factor is input into the software to convert pixels to millimeters. The size and accuracy of the vector fields are determined by the camera’s field of view (also known as window size). Large windows are useful for collecting data over a larger area, or to collect more precise data over a smaller area, however it is important to consider that the computation time is proportional to window size.

In order to minimize error, window sizes should be large enough that the particles do not travel more than ¼ of the window size between any given frame. A lower time between frames is also useful for increasing velocity precision.

# Experiment/Computational Setup

The Particle Image Velocimetry (PIV) experiment employs the Armfield H41 Laser PIV System, a highly precise tool for measuring flow velocities. This system incorporates a Class 3B laser with a 660 nm wavelength, a VGA CMOS sensor camera with a 640 x 480 pixel resolution, and integrated triggering and timing electronics. Laser safety is paramount, and laser safety glasses with an OD of 6+ for the 660 nm wavelength are provided for all participants. These safety glasses must be worn at all times during measurements to prevent direct exposure to the laser beam or reflected laser light.

The experimental setup occurs within a custom-built, open-top water tunnel made of clear acrylic, allowing optical access through its walls and the upper surface. Water is pumped from a reservoir beneath the tunnel using a variable-speed pump. A system of baffles, flow guide vanes, and honeycomb flow straighteners ensures smooth and controlled flow. The tunnel features an 8-inch-wide test section open at the top, with water levels ranging from 8 to 10 inches. In this test section, two models are employed for measurements: a finite cylinder measuring 6 inches in length with a 1.5-inch diameter and a subscale Onera M6 symmetric, swept, and tapered wing. Detailed information regarding the Onera M6 wing's profile coordinates and geometry is available in the supplementary information.

The two models utilized in this experiment, the finite cylinder and the subscale Onera M6 wing, play pivotal roles in our velocity measurements. The finite cylinder measures 6 inches in length with a diameter of 1.5 inches. In contrast, the subscale Onera M6 wing is derived from the design report, maintaining the same sweep and taper angles. However, it is scaled down with a 4.0-inch root chord and a 2.27-inch tip chord. The geometrical data and coordinates of the Onera M6 wing can be found in the supplementary information.

The flexibility of these models enables us to conduct a comprehensive study of flow patterns, velocities, and interactions within the water tunnel. The PIV system will capture particle motion and velocity vectors, providing valuable insights into fluid dynamics and aerodynamic phenomena. A well-executed experiment, coupled with precise data analysis, will deepen our understanding of fluid behavior in various scenarios and geometries.

Water Tunnel Setup: Assemble the custom-built open-top water tunnel securely and verify proper alignment. Place the tunnel on a stable surface. Connect the variable-speed pump to the water reservoir beneath the tunnel. Position the over-under baffles in the plenum to suppress unsteady flow and ensure water flows through the contraction section with flow guide vanes and honeycomb flow straighteners before entering the 8-inch-wide test section. The water level should be maintained at approximately 8 to 10 inches in height.

Model Installation: Carefully position the finite cylinder and the subscale Onera M6 wing in the constant-area test section of the water tunnel. Ensure that the models are securely mounted for measurements.

Data Acquisition: Set up the Armfield H41 Laser PIV System for data acquisition. Adjust the laser pulsing, image exposure timing, and other system parameters according to the specific experiment requirements.

Measurement Procedure: Conduct PIV measurements by releasing polyamide particles into the water flow. Capture particle images using the laser and camera system. Perform measurements under different flow conditions and model configurations.

Data Analysis: After acquiring the PIV images, perform data analysis to determine velocity vectors and flow patterns using specialized software and algorithms.

Safety Protocol: Throughout the experiment, ensure that laser safety measures are strictly followed, and laser goggles are worn at all times during measurements. Any need to briefly observe the illuminated particles should be executed cautiously. Avoid any physical exposure to the laser beam and prevent direct viewing of the laser beam or its reflections.

# Results

A black and white pattern

Description automatically generated

Figure 1

A screenshot of a computer generated image

Description automatically generated

Figure 2

A screenshot of a computer generated image

Description automatically generated

Figure 3

A screenshot of a computer generated image

Description automatically generated

Figure 4

A screen shot of a screen

Description automatically generated

Figure 5

A black and white pattern

Description automatically generated

Figure 6

A screenshot of a computer generated image

Description automatically generated

Figure 7

A screenshot of a computer generated image

Description automatically generated

Figure 8

A screenshot of a computer generated image

Description automatically generated

Figure 9

A screen shot of a screen

Description automatically generated

Figure 10

A black and white pattern

Description automatically generated

Figure 11

A screenshot of a computer generated image

Description automatically generated

Figure 12

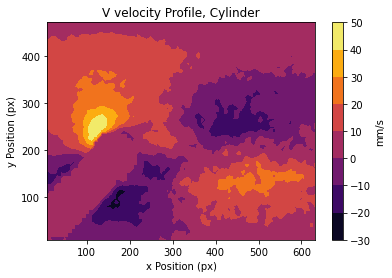


Figure 13

A screenshot of a computer generated image

Description automatically generated

Figure 14

A screen shot of a computer screen

Description automatically generated

Figure 15

A close-up of a grid

Description automatically generated

Figure 16

A screenshot of a computer screen

Description automatically generated

Figure 17

A colorful background with black border

Description automatically generated

Figure 18

A screenshot of a computer screen

Description automatically generated

Figure 19

1. **Discussion**

Using the Python code shown below in the appendix C labelled part 1 it is found that R = 4520.76 and R=9591.35. In Lab 1 it was found that Reynold number for the NACA 4412 is 320,660 and for the NACA 0012 is 314,058. The values form Lab 4 are much smaller form those calculated in Lab 1. Though all four Reynolds numbers are larger than 1000 so the assumptions made in Lab 1 are still valid. Meaning that inertial forces must be significantly larger than inertial forces. The code is shown below in appendix C labeled part 2 shows an average mean vector field and averaged turbulence field. These graphs are shown above in figures 1, 6, 11, and 16 show the mean vector field using a quiver plot. The graphs are shown above in figures 2-4, 7-9, 12-14, and 17-19 show the velocity components and velocity magnitude as separate contour plots. The graphs are shown above in figures 5, 10, and 15 show a plot of velocity fields upstream and two locations downstream while also referencing free stream conditions.

By conducting a control volume analysis, the drag coefficients for the 0 AoA airfoil, 16 AoA airfoil, and cylinder were found to be 2.004, 3.369, and 1.880 respectively. These numbers are much larger than the numbers available in literature, which typically sit at around .02 – 1. This discrepancy is most likely due to the high root mean squared values of the downstream velocity, which resulted in the mean velocities not being accurate representations of any given instance.

The benefits of PIV measurements are that we can clearly see how the velocity in the stream is affected by each test article. It is very easy to see how the velocity slows down downstream from each test article, and how the velocity is higher farther away from the article. This behavior makes it very easy to observe differences between different shapes, and it is easy to compare and contrast the behavior of velocities around different articles. One drawback/challenge is that there is a very small test area that can be observed in the water tunnel. It can be seen in the velocity graphs that we cannot see where the obstruction of the test article stops affecting the flow. There’s also a shadow cast by each article in the bottom left corner of each graph, and due to this shadow, we cannot observe the behavior of the flow in that area. Another challenge could be that the grain of the seed could deteriorate the test articles over time, causing unpredictability in the surfaces of each article. Even a small flaw in the surface of the test article could greatly affect the velocity vectors and profiles. When it comes to measuring, reducing, and analyzing data, there are a lot of imprecisions. The camera used to collect data did so at a rate much slower than what we were observing, especially when considering the lag from the camera to the software. The vector fields are also not incredibly precise, because since the seeding solution consists of such small grains, it is difficult to accurately display how all of the surrounding flow is acting. Overall, although the data collected and analyzed does give a good impression of how the flow is acting, there is a lot of specificity that we are unable to collect and analyze simply due to the accuracy of equipment and analytical tools and techniques.

When looking at our results, it can be observed that there are areas of the graphs that appear to have no flow. In all of the vector graphs, there are sections with arrows much smaller than the rest of the vectors on the graphs. For all three test cases (cylinder, 0º AoA airfoil, 16º AoA airfoil), these empty areas represent the area of the picture that was occupied by the test article. Since there is no flow in this area, naturally the vectors will be virtually nonexistent. Similarly, in the U velocity graphs, there are extremely dark purple sections representing essentially no velocity; these areas represent the flow obscured by the test articles as well. Building off of this, it can be observed in all of the U velocity graphs that the area downstream from each test article is dark and tapers off into a lighter color. This is due to the test article obscuring the flow, so the seeding in the water does not appear in the water directly behind the test article.

It can also be observed from the U velocity that the velocity is the highest at the furthest distance from the test article. This is because there is no object obscuring the flow, so the seed can move freely without interruption, following the direction of the water. An interesting trend can also be observed when looking at the V velocity graphs. At the front of each test article, there is a large spot where the velocity increases. This can be attributed to the seed deflecting off of the curved surface of the front of each test article, causing the seed to move quickly in the V direction. It can also be observed that there is little to no velocity on the upper back half of each test article; this is due to each item’s curvature. Since they are curved on that side in the direction of the flow, the seed does not deflect off of that surface in the V direction.

On the free stream vs test article graphs, we can see that the areas where the free stream velocity is bigger than the test article velocity are the areas that are behind the test article. This is because in the free stream, there is no object obscuring the path of the seed, but the test article does cause the areas behind them to have a slower velocity. At the areas not affected by the test article, the velocity for both the test article case and the free stream case appear to be about the same. All in all, we can observe that the velocities in general are higher farther away from the test article, and lower downstream from the test article.

# Conclusion

In this lab velocity field was able to be more accurately measured using Particle Image Velocimetry. Using the water tunnel to measure the flow is a less intrusive way to capture this data as nothing is inserted in the flow to obstruct it and take measurements. With the data captured Reynolds number and multiple graphs were able to be generated. Reynolds number was compared to the Reynolds number calculated from lab 1. Vector field graphs were generated as well as multiple velocity graphs with the data found.

The data and the findings of this lab could be continued by adding more objects in the water tunnel. More angles of attack could be observed and more data could be collected from this as well. Using a larger observation window and a higher speed camera (due to the lag) could also improve the data collected. Also knowing exactly how much seed/particles are in the tunnel by properly cleaning it out could improve data collection. A more controlled environment as far as light surfaces like the particles being tracked that were just smudges on the observation window or a shadow reflection from an object not in the lab could improve collected data.

# Appendix A: Raw Data

*A screenshot of a computer

Description automatically generated*

*Figure 20: Calibration Scaling Factor*

Due to the extensiveness of the data, all original vector files can be found in the following O neDrive folder: [Lab Data](https://mailuc-my.sharepoint.com/:f:/g/personal/stoelthd_mail_uc_edu/EjsPSWUO9tNJsUi_nP4UfS4BXnwVFgGrcMtOz0HbQjrCZw?e=tH82Ea)

# Appendix B: Sample Calculations

Sample Calculation Reynolds Number

For this sample calculation, the kinematic viscosity of water, , is 1.002E-3 Ns/m^2, the density is 997 kg/m^3, and the d is the cylinder diameter.

**Appendix C: Python Code**

*# %% [markdown]*

*# ## Part 1*

*# %%*

import glob

import os

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

*#getting the directory for the stored CSV's for the flow only case*

csvdir = "C:\\Users\\Harrison\\OneDrive - University of Cincinnati\\Year 5\\Aero Lab\\Lab Work\\Lab 4\\Lab Data\\Flow only, vectors"

*#importing the csv files*

csvfiles = glob.glob(os.path.join(csvdir, "\*.csv"))

*#creating an array to store the imported csv dataframes*

dataframes = []

for csvfile in csvfiles:

    df = pd.read\_csv(csvfile) *#reading each csv file*

    dataframes.append(df) *#adding the dataframe from the csv into the dataframes array*

*# %% [markdown]*

*# The cylinder and wing are placed in the freestream where the effects from the walls (bottom of tunnel, top of water) are the lowest, therefore we want to use that velocity to determine Reynold's number. For each frame, every row (y position) will be averaged and then all the frame rows will be averaged (gives v\_avg row 1, v\_avg row 2, etc.). Plotting the data will show which rows are the highest, and therefore the freestream. The freestream rows will then be averaged and the resulting speed will be used in the Re number calculation.*

*# %%*

avg\_speed = []

row = 0

print(len(dataframes))

for k in range(len(dataframes)):

    df = dataframes[k] *#getting individual frame data*

    speeds = []

    speed\_arr = []

    for i in range(len(df)):

        u = df.loc[i][2] *#getting u component*

        v = df.loc[i][3] *#getting v component*

        speed = np.sqrt(u\*\*2 + v\*\*2) *#calculating speed for the position*

        speed\_arr.append(speed) *#adding the speed value to the array*

    avg\_speed.append(np.average(speed\_arr)) *#average speed for the frame*

*#counter to help track progress*

    if k/100 == int(k/100):

        print("Frame {}".format(k))

print(avg\_speed) *#printing the avg speed array*

flow\_speed = np.average(avg\_speed) *#calculating the average flow speed*

print(flow\_speed) *#displaying average flow speed*

*# %%*

*#constants*

rho = 997 *#kg/m^3*

mu = 1.002E-3 *#Ns/m^2*

flow\_speed = np.average(avg\_speed)/1000 *#m/s*

r\_c = 101.6 *#mm*

r\_t = 57.66 *#mm*

Cyl\_dia = 1.5/39.37 *#m*

t = (r\_t/r\_c) *#non-dimensional, taper ratio*

MAC = r\_c \* .66 \* ((1+t+t\*\*2)/(1+t))/1000 *#m, mean aerodynamic chord*

Re\_cyl = (rho\*flow\_speed\*Cyl\_dia)/mu

Re\_MAC = (rho\*flow\_speed\*MAC)/mu

print("Re\_cyl: {:.3f} \nRe\_MAC: {:.3f}".format(Re\_cyl, Re\_MAC))

*# %% [markdown]*

*# # Part 2*

*# A python function will take in the instantaneous vector fields for each image and then average the entire window to get the average vector field. The averaged mean turbulence RMS will also be computed*

*# %%*

*#thought out method:*

*## Part A: Compiling all the data*

*# 1. Open the folder that contains the csv files*

*# 2. Open each csv file and then place the data into a dataframe and then into an array*

*# 3. Generate a panda's data frame (rows = frame #, columns = vector at position) [refer as total frame]*

*## Part B: Determining the mean velocity field*

*# 4. Calculate the vector for each position (per frame) and then populate the corresponding cell of the data frame*

*# 5. After all frames have been analyzed, find the average vector value for each column (if a cell is NaN, 0, -1.#ND, etc. it is not included)(every column is the same position between frames)*

*# 6. Populate the Average\_data\_frame with the average vector value (row = position, column = vector)*

*## Part C: Determining the RMS mean velocity field*

*# 7. Using the total data frame, subtract the mean velocity from each position*

*# 8. Populate new total\_RMS\_frame*

*# 9. Average the columns, and populate the RMS\_data\_frame*

*# %%*

*## Part A.1 and A.2 ##*

*#getting the directory for the stored CSV's for the flow only case (ensure all \ are \\)*

csvdir = "C:\\Users\\Harrison\\OneDrive - University of Cincinnati\\Year 5\\Aero Lab\\Lab Work\\Lab 4\\Lab Data\\23-10-20 10.30.02\\Masked Data"

*#importing the csv files*

csvfiles = glob.glob(os.path.join(csvdir, "\*.csv"))

*#creating an array to store the imported csv dataframes*

dataframes = []

for csvfile in csvfiles:

    df = pd.read\_csv(csvfile) *#reading each csv file*

    dataframes.append(df) *#adding the dataframe from the csv into the dataframes array*

*# %%*

*## Part A.3 & B.4 ##*

*#importing time to track how long it takes to compute*

import time

*#creating a dataframe with number of columns equal to the number of data points*

*#the row is equal to the frame # while the column # is the position of the vector*

dt = dataframes[0]

vector\_total\_frame = pd.DataFrame(columns=np.arange(len(dt)))

u\_total\_frame = pd.DataFrame(columns=np.arange(len(dt)))

v\_total\_frame = pd.DataFrame(columns=np.arange(len(dt)))

*#initial time*

if k/100 == int(k/100):

    time0 = time.time()

for k in range(len(dataframes)):

    dt = dataframes[k]

    vector\_values = []

    u\_values = []

    v\_values = []

    time0 = time.time() *#starting time*

*#computing the vector for each position*

    for i in range(len(dt)):

        vector = np.sqrt((dt.loc[i][2]\*(1/1.3499/0.019))\*\*2 + (dt.loc[i][3]\*(1/1.3499/0.019))\*\*2) *#calculating the vector*

        vector\_values.append(vector) *#adding vector value to array*

        u\_values.append(dt.loc[i][2]\*(1/1.3499/0.019)) *#u values*

        v\_values.append(dt.loc[i][3]\*(1/1.3499/0.019)) *#v values*

*#adding the computed vectors to the total\_data frame*

    vector\_total\_frame.loc[k] = vector\_values

    u\_total\_frame.loc[k] = u\_values

    v\_total\_frame.loc[k] = v\_values

    if k/100 == int(k/100):

        timef = time.time() *#time after computation*

        timet = timef - time0

        print("Frame {:.1f} took {:.3f} s to compute".format(k, timet))

*# %%*

*#Debugging cell*

print(vector\_total\_frame)

*# %%*

*## Part B.5 and B.6 ##*

*# Determining the average vector for the cell*

*#creating the Average\_data\_frame*

Average\_data\_frame = pd.DataFrame(columns=["x (mm)", "y (mm)", "u mm/s", "v mm/s", "v\_avg (mm/s)"])

dt = dataframes[0]

*#populating Average\_data\_frame with the x and y coords*

time0 = time.time()

for i in range(len(dt)):

    Average\_data\_frame.loc[i] = dt.loc[i][0]

    Average\_data\_frame.loc[i][1] = dt.loc[i][1]

timef = time.time()

print("Positions placed in: {:.2f} s".format(timef-time0))

*#calculating the average value*

for k in range(len(dt)): *#iterating through the columns (each column is a position)*

    vector\_avg = 0

    u\_avg = 0

    v\_avg = 0

    v\_divider = 0

    u\_divider = 0

    vec\_divider = 0

*# if k/100 == int(k/100):*

*#     time0 = time.time()*

*#checking the vectors for if the value is zero or null*

    for i in range(len(dataframes)): *#iterating the row*

        if vector\_total\_frame.loc[i][k] != 0: *#pd.isnull(total\_frame.loc[i][k]) == False:*

            vector\_avg = vector\_avg + vector\_total\_frame.loc[i][k] *#adding the values together*

            vec\_divider = vec\_divider + 1 *#number of usable values*

        if u\_total\_frame.loc[i][k] != 0:

            u\_avg = u\_avg + u\_total\_frame.loc[i][k]

            u\_divider = u\_divider + 1

        if v\_total\_frame.loc[i][k] != 0:

            v\_avg = v\_avg + v\_total\_frame.loc[i][k]

            v\_divider = v\_divider + 1

        else:

            divider = divider *#keeping the divisible number constant*

*# if avg == 0:*

*#     print("Value is zero")*

*# else:*

*#     cell\_avg\_vector = avg/divider #calculating the average vector*

*#Calculating the average vector and adding it to the frame*

    if u\_divider != 0:

        Average\_data\_frame.loc[k][2] = u\_avg/u\_divider

    else:

        Average\_data\_frame.loc[k][2] = 0

    if v\_divider != 0:

        Average\_data\_frame.loc[k][3] = v\_avg/v\_divider

    else:

        Average\_data\_frame.loc[k][3] = 0

    if vec\_divider != 0:

        Average\_data\_frame.loc[k][4] = vector\_avg/vec\_divider

    else:

        Average\_data\_frame.loc[k][4] = 0

*# print(Average\_data\_frame.loc[k])*

*# print(Average\_data\_frame)*

*# %%*

*## Debugging cell*

print(Average\_data\_frame)

*# %%*

*## Part C ##*

*# Creating the RMS dataframe*

RMS = pd.DataFrame(index=np.arange(len(dataframes)), columns=np.arange(len(dt)))

*#computing the RMS for each cell*

for i in range(len(dataframes)): *#row of the Mean\_RMS*

    for j in range(len(vector\_total\_frame.loc[0][:])): *#columns of Mean\_RMS*

        RMS.loc[i][j] = (vector\_total\_frame.loc[i][j] - Average\_data\_frame.loc[j][4])

print(RMS)

*# %%*

*#Computing the mean RMS*

mean\_RMS = pd.DataFrame(columns=["Mean RMS"])

for i in range(len(dataframes)):

    avg = 0

    divider = 0

    for j in range(len(RMS)):

        if pd.isnull(RMS.loc[j][i]) == False:

            avg = avg + RMS.loc[i][j]

            divider = divider + 1

        else:

            divider = divider

    RMS\_avg = avg/divider

    mean\_RMS.loc[i] = RMS\_avg

print(mean\_RMS)

*# %% [markdown]*

*# # Part 3*

*# %%*

*## Quiver Plot ##*

*# Getting the Data*

x = Average\_data\_frame["x (mm)"].to\_numpy()

y = Average\_data\_frame["y (mm)"].to\_numpy()

u = Average\_data\_frame["u mm/s"].to\_numpy()

v = Average\_data\_frame["v mm/s"].to\_numpy()

*#Plotting every other 4 data point*

plt.quiver(x[::4], y[::4], u[::4], v[::4])

plt.xlabel("x Position (px)")

plt.ylabel("y Position (px)")

plt.show()

*# %% [markdown]*

*# # Part 4*

*# %%*

*## Contour plots*

vector = Average\_data\_frame["v\_avg (mm/s)"].to\_numpy()

*#plotting the contours*

u\_plot = plt.tricontourf(x,y,u, cmap="inferno")

plt.xlabel("x Position (px)")

plt.ylabel("y Position (px)")

plt.title("U velocity Profile")

cbar = plt.colorbar(u\_plot)

cbar.set\_label("mm/s")

plt.show()

v\_plot = plt.tricontourf(x,y,v, cmap="inferno")

plt.xlabel("x Position (px)")

plt.ylabel("y Position (px)")

plt.title("V velocity Profile")

cbar = plt.colorbar(v\_plot)

cbar.set\_label("mm/s")

plt.show()

vector\_plot = plt.tricontourf(x,y,vector, cmap="inferno")

plt.xlabel("x Position (px)")

plt.ylabel("y Position (px)")

plt.title("Vector velocity Profile")

cbar = plt.colorbar(vector\_plot)

cbar.set\_label("mm/s")

plt.show()

*# %% [markdown]*

*# # Part 6*

*# %% [markdown]*

*# $$ C\_{d\_o} = \frac{1}{\frac{1}{2} \rho {U\_{atm}^2}} \Sigma \left[ \frac{\rho (U\_{atm}^2 - U\_{downstream}^2)}{g} \right]$$*

*# %%*

*# Computing drag for each case*

import pandas as pd

import numpy as np

*# Constants*

rho = 997 *#kg/m^3*

mu = 1.002E-3 *#Ns/m^2*

*# %%*

*# Importing average data*

airfoil\_0\_raw = pd.read\_csv('Summary Data\Average\_Data Airfoil 0 AoA.csv')

airfoil\_16\_raw = pd.read\_csv('Summary Data\Average\_Data Airfoil AoA.csv')

cylinder\_raw = pd.read\_csv('Summary Data\Average\_Data Cylinder.csv')

freestream\_raw = pd.read\_csv('Summary Data\Average\_Data Free Stream.csv')

*# Filtering data for only max x value*

airfoil\_0 = airfoil\_0\_raw[airfoil\_0\_raw['x'] == 632]

airfoil\_16 = airfoil\_16\_raw[airfoil\_16\_raw['x'] == 632]

cylinder = cylinder\_raw[cylinder\_raw['x'] == 632]

freestream = freestream\_raw[freestream\_raw['x'] == 632]

*# %%*

*# Joining each test case to the freestream data*

Cds = []

for case in [airfoil\_0, airfoil\_16, cylinder]:

    joined = pd.merge(freestream, case, left\_index = True, right\_index = True, how='outer') *# Join test case to freestream data*

    joined['Cd'] = (1 / (.5 \* rho \* (joined['u\_x']\*.001)\*\*2)) \* ((rho / 9.81) \* ((joined['u\_x']\*.001)\*\*2 - (joined['u\_y']\*.001)\*\*2)) *# Compute drag coefficient at each Y position at max X*

    Cd = joined['Cd'].sum() *# Total drag coefficient*

    Cds.append(Cd)

*# %%*

print(Cds)

# References

1. Cuppoletti, D. (2022) AEEM5081L- Lab 4 – Particle Image Velocimetry in a Water Tunnel. rep., pp. 1–8.

2. Cuppoletti, D. (2022) AEEM5081L Lab Report Grading Rubric. rep., pp. 1–3.

3. Cuppoletti, D. (2022) AEEM5081L 01 – Review of Fluid Mechanics and Aerodynamics. pp. 6.