ME 552: Lab 2 Hot Wire Anemometer Calibration and Jet Flow Experiment

Matthew Zaiger, Julia Thurber, Daniel Cowan, Andrew Alfermann

February 2017

1 Introduction

Hot-wire Anemometry (HWA) is a turbulent flow measurement technique used to determine mean and fluctuating velocity components, turbulence, and localized temperature. HWA sensors are thin metallic wires that are heated by electric current and cooled through forced convection in the incident flow. These sensors are typically made of platinum or tungsten and are drawn to a diameter of 0.5-5 μm and a length of 0.1-1 mm [1]. The main advantages of HWA over comparable methods, such as laser velocimetry, are a higher sampling frequency, smaller probe volume, and less expense.

Although a hot wire is more common for this technique, a hot film is sometimes used, typically in liquids, and both of these can be used with both constant-current and constant-temperature systems. Constant-temperature systems are composed of a Wheatstone bridge and amplifier circuit that pass changes in current, due to the temperature dependent nature of the electrical resistance of the wire, through a feedback loop to maintain constant temperature. The voltage at the top of the bridge can then be related back to the velocity of the flow [2].

For this lab exercise, the IFA 300 Constant Temperature Anemometer System was paired with a single sensor hot wire probe to first calibrate the probe and then to conduct a jet flow experiment. LabVIEW was used to collect the data, and a Python code was developed to process and analyze the data.

2 Assumptions

While using Bernoulli's equation to calculate flow velocity using differential pressure in the calibration procedure (see section 4.1), the change in height was considered negligible for low density air. Additionally, in the calibration procedure the velocity in the plenum is negligible. The temperature of the gas is uniform in all cases. While measuring flow velocity across the burner, the temperature was assumed to be constant. The wire temperature is held constant for each each overheat ratio. Radiation effects are negligible and temperature fluctuations across the wire are due to forced convection. The energy output of the wire is proportional to the voltage across the bridge, which is measured.

3 Design Stage Uncertainty

Design stage uncertainty trees were developed for flow velocity and turbulence intensity. The tree for flow velocity can be found in Appendix D with a brief explanation. The tree for turbulence intensity is not included due to its simplicity. Refer to the method used for the flow velocity tree and the turbulence intensity equations in Appendix B, if further investigation is desired.

Design stage uncertainty was calculated using the first order Kline-McClintock method, as automated in the Python code included in Appendix C. The calculated uncertainty values were integrated into error bars on each of the plots included in the analysis. These plots were also created using Python.

4 Calibration

4.1 Procedure

To calibrate the HWA, compressed air was fed through the hot wire calibrator and two nozzles before flowing vertically across the horizontally held probe. This probe was wired to the IFA 300 Anemometer System, which was connected to a USB DAQ and PC equipped with both ThermalPro and LabVIEW software packages. The flow velocity was calculated using Bernoulli's equation and the differential pressure between the calibrator plenum chamber and ambient.

After aligning the probe with the jet center at approximately 2mm height, the cold wire probe resistance, the shorted BNC cable resistance, and the probe holder resistance were obtained. The zero flow and ten other data points were then recorded for various flow rates at a predetermined overheat ratio (1.53). The pressure transducer and hot wire voltage readings over a 5 second period were also recorded for each flow rate. This was then repeated for two more overheat ratios (1.22 and 1.39).

4.2 Analysis

The data obtained during the experiment was analyzed using a Python code that found the average and standard deviation of the 50,000 samples taken for each data point. This Python code is found in Appendix C, and the film temperatures corresponding to the three overheat ratios are in the table below.

Overheat Ratio	T_w (K)
1.22	304.15
1.39	394.15
1.53	478.15

The calibration curves for each of the three overheat ratios were found and plotted in Python. The large degree of uncertainty attributed to the lower voltages in these figures is due to the relatively large tolerance of the pressure transducers used in the experiment when compared to the very small differential pressure values at low velocities. The uncertainty improved greatly at higher voltages as the differential pressure increased to values greater than the tolerance of the pressure transducers, (see figures 1, 2, and 3).

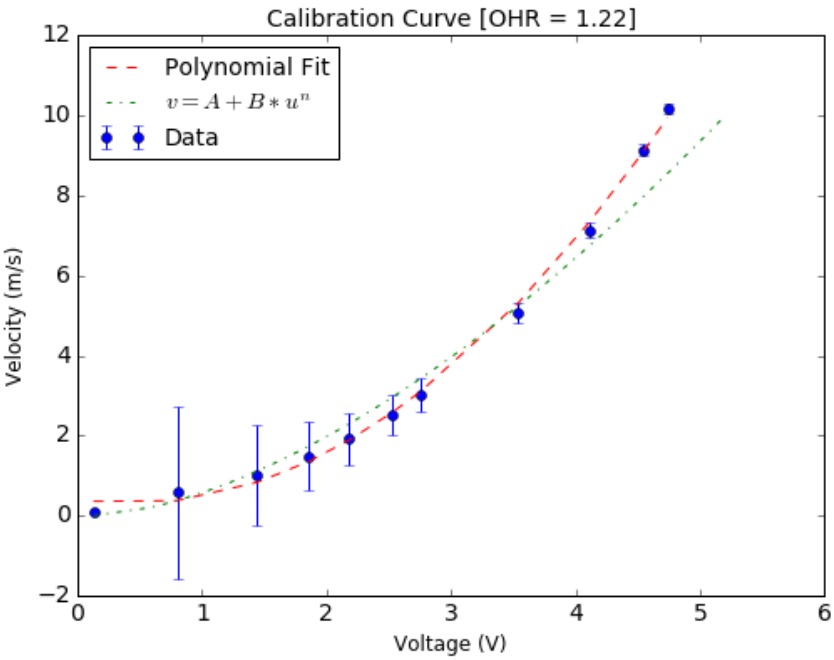


Figure 1: Calibration Curve for 1.22 Overheat Ratio, note that the error bars on the x axis are too small to be seen

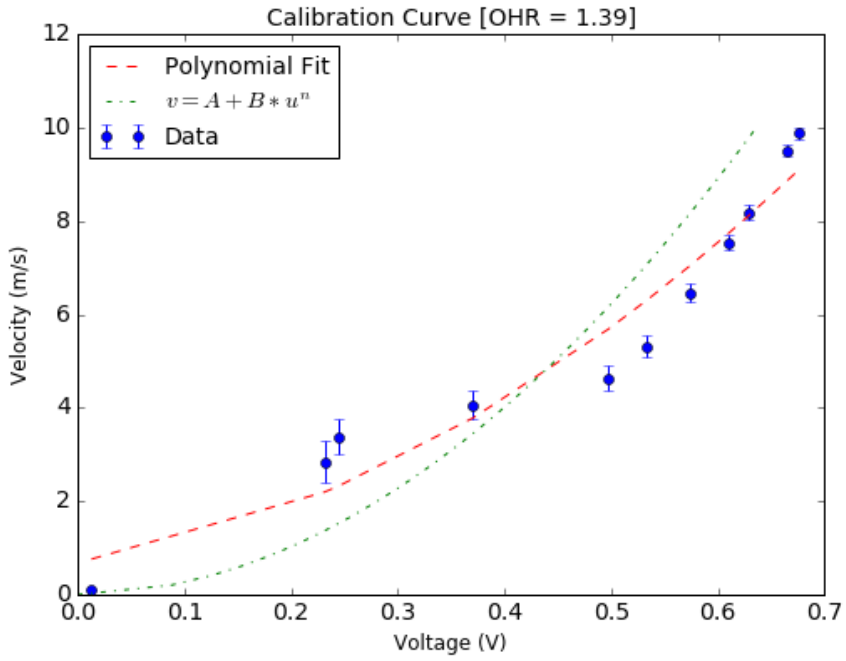


Figure 2: Calibration Curve for 1.39 Overheat Ratio, note that the error bars on the x axis are too small to be seen

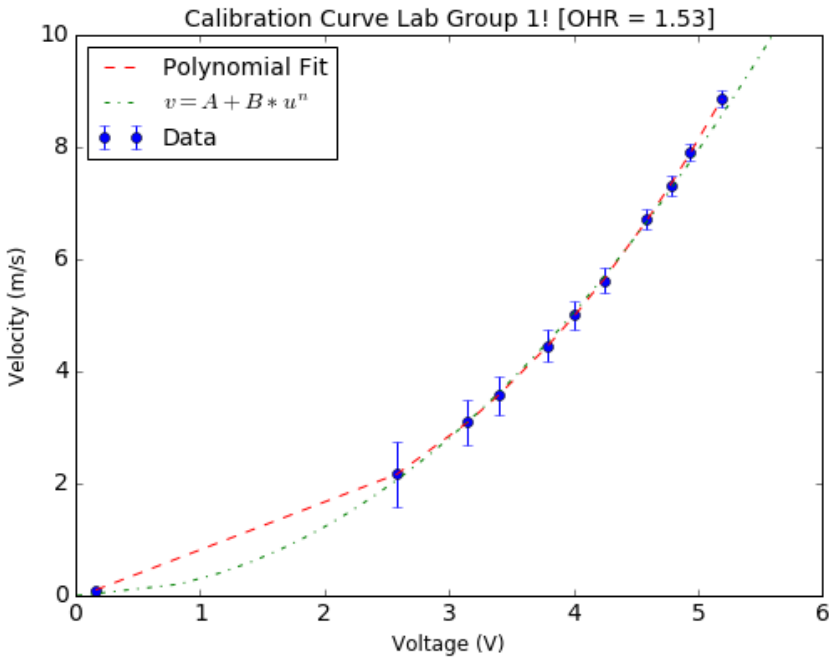


Figure 3: Calibration Curve for 1.53 Overheat Ratio, note that the error bars on the x axis are too small to be seen

Next, the velocity sensitivity (proportional to the voltage sensitivity) for the three overheat ratios was calculated from Equation 17 in Appendix B and plotted in Python, shown in figure 4. This is the sensitivity of the velocity to a change in voltage. Based on this plot, and the velocity for the 1.39 OHR was found to be a fraction of the velocity at the other OHRs, the data obtained for the 1.39 OHR is highly suspect. If time permitted, the calibration of the 1.39 OHR should have been repeated. Figure 5, shows a steady increase in sensitivity with increased voltage as expected when compared to Equations 13 and 14 in Appendix B. A higher OHR was expected to correspond to a higher sensitivity, however the trends shown in Figure 5 do not clearly support this hypothesis. Nevertheless, this trend implies that higher voltages have a greater impact on velocity sensitivity than changing the overheat ratio.

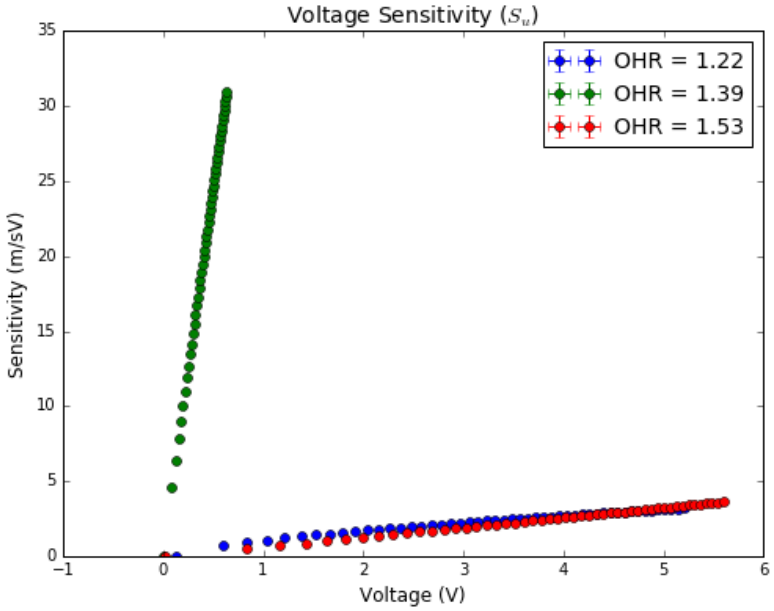


Figure 4: Voltage Sensitivity, note that the errorbars are not plotted correctly and as a result are not shown. However, the trends still are valuable.

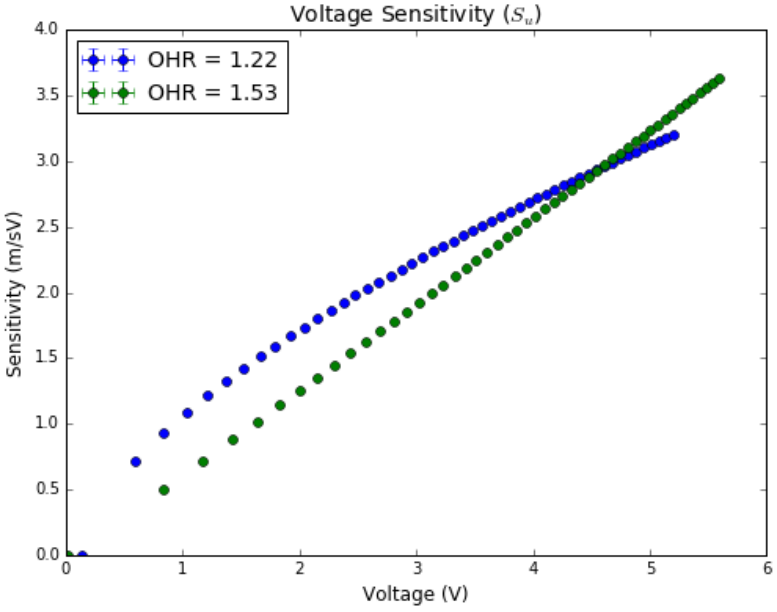


Figure 5: Voltage Sensitivity Zoom for OHR 1.22 and 1.53, note that the errorbars are not plotted correctly and as a result are not shown. However, the trends still are valuable.

5 Jet Flow Experiment

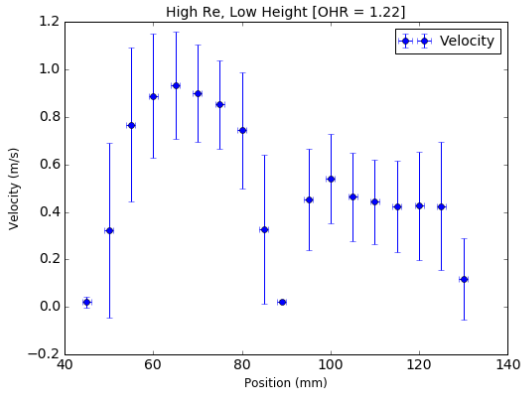
5.1 Procedure

The velocity and turbulence of coflow exiting a MILD flame burner were measured, and the volumetric flow rate was recorded through LabVIEW. The recently calibrated probe was again positioned perpendicular to the flow at a height of approximately 2 mm. After the outer diameter of the tube jet and inner diameter of the coflow tube were recorded, the pressure regulator at the build was set between 60 and 80 psi while the upstream pressure regulator was increased to achieve the desired flow rate. After verifying the LabVIEW recording capabilities with this test, flow was turned off and the signal voltage for the energized hot-wire at the predetermined overheat ratio was recorded. The flow was turned back on to achieve the first Reynolds number condition, and another recording was taken. The probe was then moved across the burner in 5mm increments, with a recording being taken at each increment. The height of the probe was increased to 30mm, and signal voltages were again recorded as the probe traversed the burner in 5mm increments.

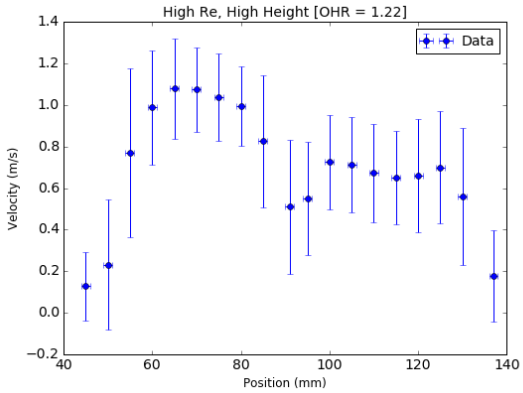
After collecting all data at the two heights across the entirety of the burner, the entire procedure was repeated for a second Reynolds number. Finally, the entire experiment was run again for both different overheat ratios. In summary, sensor voltages for all combinations of two Reynolds numbers, two heights, and three overheat ratios were collected. For data analysis purposes, only one overheat ratio will be used. Also we felt that it was necessary to include some precision error in the determining of turbulent intensities due to the impact of voltage sensitivies on voltages at different velocites, see Appendix D.

5.2 Analysis

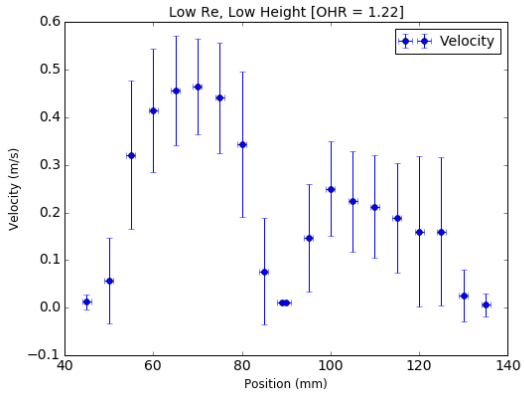
For the hot wire within 2 mm of the fuel jet, the velocity profile forms an 'm' (as seen in figure 6) shape where there is minimal velocity at the borders and at the center near the fuel jet. The velocities at the center are close to zero because there is no flow of gasses through the fuel jet. Without any flow from the jet, a stagnation point is formed. At the borders, the velocity is also close to zero because the air outside of the coflow cylinder is stagnant. Also, measurements of the velocity are shown to have a much greater error the radial location measurements, details of which can be found in Appendix D.



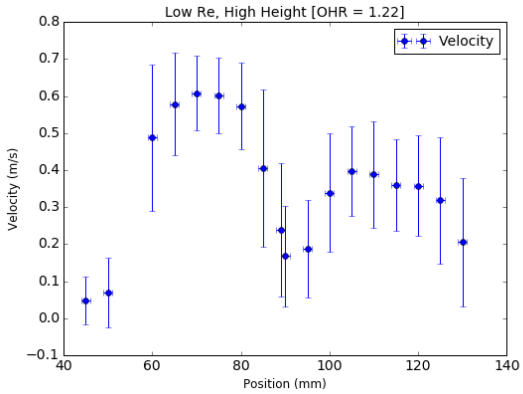
(a) high Re : low height



(b) high Re : high height



(c) low Re : low height

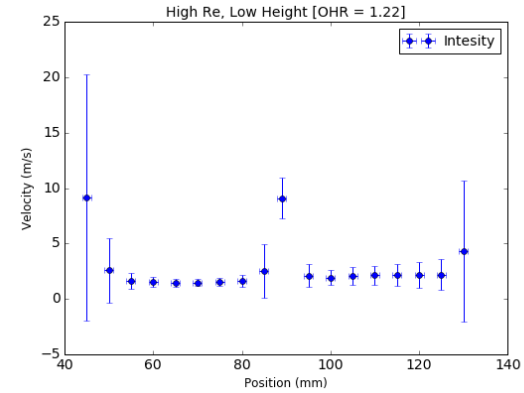


(d) low Re : high height

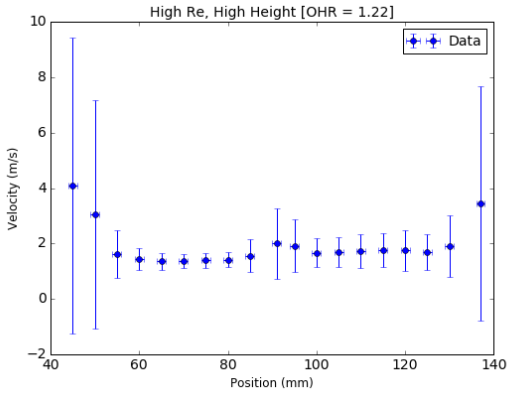
Figure 6: velocity profile across the burner for 2 different heights and 2 Reynolds numbers (x errorbars are small, for larger prints of each figure see Appendix E

As the flow raises in height, the 'm' shape is still present. A difference is that the flow at the edges and the center have increased in velocity. This occurs because the stagnant and dynamic gasses have been mixing. This mixing can be seen when examining the turbulence intensity plot.

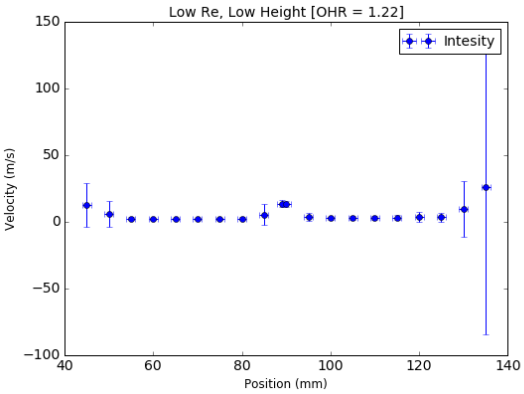
The turbulence intensity measured for the hot wire within 2 mm of the fuel jet forms backwards 'j', then a forward 'j', with a dip in the middle (see figure 7). The boundaries have a sharp increase in turbulence intensity because around the flow the gas is stagnant. Between the fuel jet and the coflow cylinder wall, the velocity is large and the turbulence intensity is almost 1 (negligible). This is because the flow is not interacting with the stagnant gasses outside the cylinder wall or the center, and therefore the flow is laminar. The center (fuel jet location) forms a stagnation point. This point has minimal turbulence intensity (close to 1) and is most likely cause by flow separation. Surrounding the stagnation point are points of higher turbulence intensity. This is most likely caused by mixing along the separation boundary.



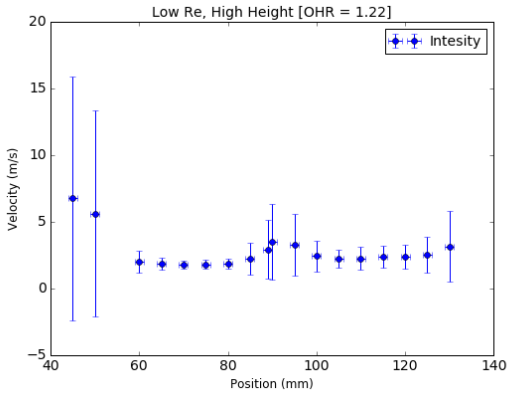
(a) high Re : low height



(b) high Re : high height



(c) low Re : low height



(d) low Re : high height

Figure 7: Turbulence intensity for 2 different heights and 2 Reynolds numbers (x errorbars are small, for larger prints of each figure see Appendix E

As the flow moves further away from the fuel jet the turbulence intensity changes to a 'w' shape. The center (at fuel jet) has a peak in turbulence intensity. This is caused by the wake from the flow separation. There is also increase in turbulence intensity at the boundaries but is is less drastic. There is more mixing moving inward from the cylinder radius.

For a higher Reynolds number the turbulence intensity permeates more slowly inwards when compared to the lower Reynolds number. This is because there is more inertia to overcome in the flow with a higher Reynolds number. Both flows are laminar, which is shown by both flows having low turbulence intensity where the flow is greatest.

One aspect on the flow that may need more analysis is that the flow increases at all locations when the flow is measured at a higher height (except for one point at the boundary). This implies that the volumetric flow rate increases after leaving the nozzle. This could be caused by the pressure being less than the ambient pressure thus shrinking the cross sectional flow diameter as it moves further away from the nozzle. This is unlikely because the shrinking of the cross sectional diameter would cause the boundary velocities to decrease as the flow leaves the burner. The velocity at the boundary increases for all cases but one point.

Another observation is that the flow on either side of the fuel jet is asymmetric. The right side has a lower velocity. The lower velocity also has a higher turbulence intensity. The coflow porous media may need to be relaid to have more symmetric flow.

6 Conclusion

HWA was used to find flow velocities and turbulence intensity across a MILD combuster. Subsequent analysis of the recorded data found that likely erroneous results were obtained with an OHR of 1.39, and therefore more data should be collected using this overheat ratio and the best voltage power-law fitting with an OHR of 1.53. The flow was found to stagnate near the outer walls of the burner, and again near the center of the burner where the fuel jet is located. Turbulence intensity was found to be greatest where the air flow interacted with the regions of stagnation. Additionally, in these areas where air flow interacted with the regions of stagnation, uncertainty values were found to be greatest due to the three-dimensional nature of turbulent flow and the directional nature of the HWA measurements.

In this laboratory, we learned the enormous magnitude of the data collection abilities of the HWA. Using the vast quantity of data obtained, relatively accurate determinations of turbulence intensity were able to be found, which would not be possible using other measurement techniques. Additionally, we gained knowledge regarding the sensitivity of the hot wire anemometer to a wide range of flow velocities, and potentially a lesson in its sensitivity to flow direction.

7 Acknowledgements

We would like to acknowledge the diligent efforts put forth by the teaching assistant, Kyle Zada. Between spending hours in the laboratory setting up and running the laboratory for each group, and spending many more hours compiling instructions and data, Kyle has shown an enormous amount of support towards our learning efforts, and has helped us to make the most of our experience.

References

[1] Genevieve Comte-Bellot. Hot-wire anemometry. Annu. Rev. Fluid Mech, pages 209–231, 1976.

[2] IFA 300 Constant Temperature Anemometer System. Trust. Science. Innovation Inc., TSI Incorporated, Shoreview, MN, 2010.

8 Appendices

This section contains all supplementary materials required to complete this lab exercise. The appendices are arranged in the following order:

(A) Nomenclature

(B) Equations

(C) Python Code

(D) Uncertainty Tree

(E) Figures

A Nomenclature

A.1 Variables:

- Density (ρ, kg/m^3)
- Electrical Current (I, A)
- Electrical Resistance (R, Ω)
- Energy (E, J)
- Height (z, m)
- Velocity (u, m/s)
- Ideal Gas Constant (R_{gas}, J/kg*K)
- Overheat Ratio (a)
- Pressure (P, Pa)
- Temperature (T, K)
- Turbulence Intensity (Y)
- Velocity (u, m/s)
- Voltage (V, V)
- Volume (v, m^3)

A.2 Subscripts:

- Across the bridge (bridge) (Example: V_{bridge})
- At/Across the Hot Wire (w) (Example: V_w)
- Ambient Conditions (amb) (Example: x_{amb})
- Average (avs) (Example: x_{avs})
- Internal to Probe (int) (Example: x_{int})
- Operating to Probe (op) (Example: x_{op})
- Plenum Conditions (pln) (Example: x_{pln})
- Reference Conditions (o) (Example: x_o)
- Relative to Gauge (g) (Example: $P_{pln,g}$)
- Root Mean Squares (rms) (Example: x_{rms})
- Measured (meas) (Example: x_{meas})
- When Flow is Zero (zero) (Example: V_{zero})

A.3 Known:

- $g = 9.81\ m/s^2$
- $P_{amb} = 100,200\ Pa$
- $R_{0^\circ C} = 5.85\ \Omega$
- $R_{100^\circ C} - R_{0^\circ C} = 1.33$
- $R_{gas} = 286.9\ J/kg * K$
- $R_{int} = 0.50\ \Omega$
- $T_{amb} = 293.5\ K$

B Equations

Ideal Gas Law

$$P * v = R_{gas} * T \tag{1}$$

$$\rho_{amb} = \frac{P_{amb}}{R_{gas} * T} \tag{2}$$

Bernoulli's Equation

$$\left(P + \frac{1}{2} * \rho * u^2 + \rho * g * h\right)_{pln} = \left(P + \frac{1}{2} * \rho * u^2 + \rho * g * h\right)_{amb} \tag{3}$$

$$u_{amb} = \sqrt{\frac{2 * P_{pln,g}}{\rho_{amb}}} \tag{4}$$

where:

$$P_{pln,g} = P_{pln} - P_{amb} = \Delta P \tag{5}$$

Ohm's Law

$$I = \frac{V}{R} \tag{6}$$

Overheat ratio from lab derivations:

$$a = \frac{R_{op} - R_{int}}{R_{meas} - R_{int}} \tag{7}$$

Resistance vs. temperature of wire (values given in Appendix A):

$$R_{op}(T_w) = R_{0^\circ C} + (R_{100^\circ C} - R_{0^\circ C})\frac{T_w - 0^\circ C}{100^\circ C} \tag{8}$$

Thus:

$$T_w = 100^\circ C * \frac{R_{op} - R_{0^\circ C}}{R_{100^\circ C} - R_{0^\circ C}} \tag{9}$$

Energy out at the wire:

$$\frac{E_w^2}{R_w} = (A + B * u^n) * (T_w - T_{amb}) \tag{10}$$

where:

$$R_w = R_{op} \tag{11}$$

For one overheat ratio and assuming the ambient temperature is constant:

$$E_w^2 = (A + B * u^n) \tag{12}$$

This can be reduced for the constant temperature circuit:

$$V = A + B * u^n \tag{13}$$

Which can be rewritten as:

$$u = \left(\frac{V - A}{B}\right)^{1/n} \tag{14}$$

Best Fit Powerlaw Trend Lines: Over heat ratio of 1.22 ($R^2 = 0.9594$):

$$A = 0.128165$$

$$B = 1.2324$$

$$n = 0.6148$$

Over heat ratio of 1.39 ($R^2 = 0.9695$):

$$A = 0.0$$

$$B = 0.1975$$

$$n = 0.5079$$

Over heat ratio of 1.53 ($R^2 = 0.9977$):

$$A = 0.013131$$

$$B = 1.7954$$

$$n = 0.4928$$

Turbulence Intensity:

$$Y = \frac{u_{rms}}{u_{avg}} \tag{15}$$

where:

$$u_{rms} = \sqrt{\frac{1}{N}\sum_i u_i^2} \tag{16}$$

Velocity Sensitivity:

$$S_u = \frac{\partial u}{\partial V} = \frac{1}{n}\left(\frac{V-A}{B}\right)^{\frac{1}{n}-1}\left(\frac{1}{B}\right) \tag{17}$$

C Python Code

C.1 Calibtration

Listing 1: Calibration reader.py

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-
"""
Created on Wed Feb 15 10:09:40 2017

@author: zaigerm and andrew alferman

This python code is meant to read the data collected for the HotWire...
...measurments lab.

HOW TO USE>>>>>

1) Type in the filepath to the directory of what data you want to read
2) delete the .csv from the filenames list "del filenames[index of .csv]
    You can look in filenames prior to the del comment and make sure.

and if linked to a plotter you can plot the voltages for all of the lvm files

"""
#import lvm
import numpy as np
import scipy as sc
import os
from os.path import isfile, join
import matplotlib.pyplot as plt
#from graph import graph
#from lvm_reader import Lvm_read

def double_sort(l1, l2):
    a = l1
    a2 = l2
    #b = a[:]

    for j in range(len(l1)-1, 0, -1):
    #while True:
        #if is_sorted(a) == True:
            #break
        for i in range(len(a)-1):
            if a[i] > a[i+1]:
                holder1 = a[i]
                holder2 = a2[i]
                a[i] = a[i+1]
                a2[i] = a2[i+1]
                a[i+1] = holder1
                a2[i+1] = holder2

    return a, a2


def reader():
    filepath = '/nfs/stak/students/z/zaigerm/Homework/me552/lab2
    ...../Data/OHR_122/CalibrationData'
    filenames = [f for f in os.listdir(filepath) if isfile(join(filepath, f))]

    filenames = sorted(filenames)

    #del filenames[3] # This removes the .csv File from the filenames list
```

```
    ...since its
    # set up differently then the others.

    aa = np.arange(11)
    R0 = 5.85 # resistance at zero
    RcR0 = 1.33 # Difference between zero and 100C  resistance
    Rop = 7.44 # Operation Resistance
    Rint = 0.5 #given value
    Rw = Rop + Rint # Resistance of the wire


    Tw = 100 * (Rop - R0 / RcR0)
    print Tw
    Tatm = 20.3 +273.15

    T_a = np.array([20.5, 20.8, 20.7, 20.6, 20.6, 20.5, 20.6, 20.6, 20.6, .
    ...20.6, 20.6]) +273.15
    DP122 = (np.array([0.0032, 0.0086 ,0.0648, 0.0038, 0.0528, 0.0054,...
    ...0.0045, 0.0184, 0.007, 0.0034, 0.0334]) - 0.0032)*1000

    DPatm = 0.0032*1000

    DP122 = np.array(sorted(DP122))
    #T = sorted(DP122)

    R = 286.9# sc.constants.gas_constant
    utm = (2*DPatm*R*Tatm/101325.0)**0.5
    ucal =(2*DP122*R*T_a/101325.0)**0.5

    avgV = []
    stdv = []
    counter = 0

    for name in filenames:
        voltage = []
        with open(filepath +'/' + name, 'r') as f:
            line_15 = f.readlines()[23:]
            for line in line_15:
                lines = line.strip()
                voltage.append(float(lines))
                if counter == 50000:
                    break
        vavg = np.average(voltage)
        avgV.append(vavg)
        stds = np.std(voltage)
        stdv.append(stds)
        f.close()
        counter = counter+1


    A = avgV[0] / (Rw *( Tw -Tatm))
    B = (avgV / (Rw * (Tw -T_a) - A))/(ucal**3)
    u = (((avgV / (Rw * (Tw -T_a) - A))/(B))**0.3333333)

    DP122 = sorted(DP122)

#
    u = sorted(u)
    ucal =sorted(ucal)

    volts = abs(np.array(avgV))

    ustds = np.array(stdv) * 2.
```

```
    return volts, ustds, T_a, DP122

if __name__ == '__main__':
    V, u_stds, Tambreadings, Pressreadings = reader()

#     plt.figure()
#     plt.errorbar(a[0], u, yerr = std2, fmt ='o')
#     z = np.polyfit(a[0], u, 3)
#     p = np.poly1d(z)
#     plt.plot(a[0], p(a[0]), 'r--')
#     #plt.plot(a[0], 'ro' )
#     plt.show()     #graph()
#     print "y=%.6fx^3+(%.6f)x^2+(%.6f)x+(%.6f)"%(z[0],z[1], z[2], z[3])
```

Listing 2: *Group*1$_L$*ab*2$_U$*ncertainty*$_1$*.py*

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
"""
Created on Wed Feb 15 14:47:49 2017

This program uses the uncertainties package in Python to automatically
calculate and propagate the uncertainties in the equations for the ME 552
Winter 2017 Lab 2.  The method used to propagate the uncertainties is
the Kline McClintock method.

@authors: Andrew Alferman and Matt Zaiger
"""

# Import the required packages
import numpy as np
import uncertainties as unc
import uncertainties.unumpy as unp
import matplotlib.pyplot as plt
import scipy.constants as const
import os
import powerlaw as pl

import reader as r

# Define the universal constants
R = 286.9 # J/kg*K

# For the purposes of the equation, we make an assumption about n
n = 1./3.

### Call reader function ###
V, u_stds, Tambreadings, Pressreadings = r.reader()

# List out all of the experimental data needed for the calculations
OHR = 1.22

# Conversion factors for handy dandy use later, if needed
intom = 0.0254 # Inches to meters
psitopa = 6894.76 # psi to Pascals
cftocm = 0.0283168 # cubic feet to cubic meters
inwtopa = 248.84 # Inches of water to Pascals
cfmtocms = 0.000471947 # Cubic feet per minute to cubic meters per second

# Put all the equipment tolerances in here because it's the cool thing to do
thermocouples = 1.0 # degrees C (Type T), but could be 0.75% reading
pressscale = 10.0 # Assume that the range will be 10 inches H2O
press = 0.00060 * pressscale
```

```
# hwrp = Hotwire resistance, percent
hwrp = 0.001 # Table states "0.1% +/- .01 Ohms" so 2 numbers are used
# hwra = Hotwire resistance, absolute
hwra = 0.01 # Ohms
hotwireoffset = 0.0015 # Percent accuracy
hotwiregain = 0.0015 # Percent accuracy

uv = ((hotwireoffset*V)**2 + (V*hotwiregain)**2 + u_stds**2)**0.5

Volts = [unc.ufloat(V[i], uv[i]) for i in range(len(V))]

# Add in the pressure measurements
# Input in inches of water, output in Pascals
P_atm = unc.ufloat(100200, 100)
P_diffs = [unc.ufloat(p, press*inwtopa) for p in Pressreadings]

# Add in the temperature measurements
# Input and output in Kelvin
T_atms = [unc.ufloat(t, thermocouples) for t in Tambreadings]

# Add in the resistances
# Input and output in ohms
R_0c = 5.85
R_0c = unc.ufloat(R_0c, R_0c * hwrp + hwra)
R_100c = 1.33 + R_0c
R_int = 0.50
R_int = unc.ufloat(R_int, R_int * hwrp + hwra)
R_op = 8.34
R_op = unc.ufloat(R_op, R_op * hwrp + hwra)

# Plug in each of the equations and solve.  The bottom of the tree is computed
# first in order to allow computation of the higher level tolerances.
rho_airs = [P_atm / (R * T_atm) for T_atm in T_atms]
upressures = [((2*P_diffs[i])/rho_airs[i])**0.5 for i in range(len(rho_airs))]

T_w = 100 * ((R_op - R_0c) / (R_100c - R_0c))


A = (Volts[0]**2) / (R_op * (T_w - T_atms[0]))

B = [None]*len(Volts)
u = [None]*len(Volts)


for i in range(len(Volts)):
    if i == 0:
        B[i]= unc.ufloat(-0.001, 0.012)
        u[i]= unc.ufloat(0.1, 0.01)
    else:
        B[i] = (((Volts[i]*Volts[i])/(R_op*(T_w-T_atms[i]))) - A) /\
                  (upressures[i]*upressures[i])
        u[i] = ((((Volts[i]*Volts[i])/(R_op*(T_w-T_atms[i])))-A)/B[i])**0.5

for i in range(len(B)-1):
    if i == 0:
        pass
    else:
        Bavg = (B[i]+B[i+1])/2

Bnorm = [val.n for val in B]
Bavgc = min(Bnorm)
Anorm = A.n
Aerr = A.s
```

```
umadeup = np.linspace(0, 10)
#voltage = [val.n for val in V]

############################## Figure Stuff ######################################
#print(u)
#print(V)

savepath = '/nfs/stak/students/z/zaigerm/Homework/me552/lab2/Data/figures/'


plt.figure(figsize = (8,6), dpi = 300)
unorm = np.array([val.n for val in u])
#check = pl.Fit(unorm,V)
#### OHR122
a122 = 0.128165
b122 = 1.2324
n122 = 0.6148
v122 = a122 + b122*umadeup**n122

#### OHR139
a139 = 0.0
b139 = 0.1975
n139 = 0.5079
v139 = a139 + b139*umadeup**n139
#### OHR153
a153 = 0.013131
b153 = 1.7954
n153 = 0.4928
v153 = a153 + b153*umadeup**n153
#print check.alpha
us = [val.s for val in u]
V = sorted(V)
plt.errorbar(V, unorm, yerr = us, fmt ='o', label = 'Data')
z = np.polyfit(V, unorm, 3)
p = np.poly1d(z)
plt.plot(V, p(V), 'r--', label = 'Polynomial Fit')
plt.plot(v153, umadeup,'g-.', label = '$v = A + B*u^n$')
plt.title('High Re, Low Height [OHR = 1.22]',fontsize=14)
plt.xlabel('Voltage (V)', fontsize=12)
plt.ylabel('Velocity (m/s)', fontsize=12)
plt.xticks(fontsize = 14)
plt.yticks(fontsize = 14)
plt.legend(loc = 2, fontsize = 14)

#plt.plot(a[0], 'ro' )
#plt.show()     #graph()
plt.savefig(savepath + 'OHR122_HRe_LH')
```

C.2 Flow Profile Code

Listing 3: Calibration readerflow.py

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-
"""
Created on Wed Feb 15 10:09:40 2017

@author: Andrew Alferman and Matt Zaiger

This python code is meant to read the data collected for the HotWire...
...measurments lab.
```

```
HOW TO USE>>>>>

1) Type in the filepath to the directory of what data you want to read
2) delete the .csv from the filenames list "del filenames[index of .csv]
    You can look in filenames prior to the del comment and make sure.

and if linked to a plotter you can plot the voltages for all of the lvm files

"""
#import lvm
import numpy as np
import scipy as sc
import os
from os.path import isfile, join
import matplotlib.pyplot as plt
#from graph import graph
#from lvm_reader import Lvm_read

def remove_stuff(s):
    return s[5:-4]

def double_sort(l1, l2):
    a = l1
    a2 = l2
    #b = a[:]

    for j in range(len(l1)-1, 0, -1):

        for i in range(len(a)-1):
            if a[i] > a[i+1]:
                holder1 = a[i]
                holder2 = a2[i]
                a[i] = a[i+1]
                a2[i] = a2[i+1]
                a[i+1] = holder1
                a2[i+1] = holder2

    return a, a2


def reader():
    filepath = '/nfs/stak/students/z/zaigerm/Homework/me552/lab2/Data
    ...../OHR_122/Jet_Flow_Experiment/Low_RE_High_H'
    filenames = [f for f in os.listdir(filepath) if
    ...isfile(join(filepath, f))]

    xvalues = [float(s) for s in filenames]


    a = double_sort(xvalues, filenames)
    xvalues = a[0]
    filenames = a[1]
    print filenames
    xvalues = np.array(xvalues)


    avgV = []
    voltraw = np.zeros([len(xvalues) , 50000])
    stdv = []

    j = 0

    for name in filenames:
```

```
            counter = 0
            voltage = []
            with open(filepath +'/' + name, 'r') as f:
                line_15 = f.readlines()[23:]
                for line in line_15:
                    lines = line.strip()
                    voltage.append(float(lines))
                    if counter == 49999:
                        break
                    counter = counter+1
            voltraw[j,:] = voltage
            vavg = np.average(voltage)
            avgV.append(vavg)
            stds = np.std(voltage)
            stdv.append(stds)
            f.close()

            j = j+1



        volts = abs(np.array(avgV))

        ustds = np.array(stdv) * 2.

        return np.array(volts), ustds, xvalues, voltraw

if __name__ == '__main__':
    V, u_stds, xvalues , voltraw= reader()
```

Listing 4: Calibration reader.py

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
"""
Created on Wed Feb 15 14:47:49 2017

This program uses the uncertainties package in Python to automatically
calculate and propagate the uncertainties in the equations for the ME 552
Winter 2017 Lab 2.  The method used to propagate the uncertainties is
the Kline McClintock method.

@authors: Andrew Alferman and Matt Zaiger
"""

# Import the required packages
import numpy as np
import uncertainties as unc
import uncertainties.unumpy as unp
import matplotlib.pyplot as plt
import scipy.constants as const
import os
import powerlaw as pl

import readerflow as r

# Define the universal constants
R = 286.9 # J/kg*K

# For the purposes of the equation, we make an assumption about n
n = 1./3.

### Call reader function ###
```

```
V, u_stds, x , voltraw= r.reader()

# List out all of the experimental data needed for the calculations
OHR = 1.22

voltraw = np.abs(voltraw)

hotwireoffset = 0.0015 # Percent accuracy
hotwiregain = 0.0015 # Percent accuracy

uv = ((hotwireoffset*V)**2 + (V*hotwiregain)**2 + u_stds**2)**0.5
ur = (hotwireoffset*(voltraw)**2 + (voltraw*hotwiregain)**2)**0.5
Volts = [unc.ufloat(V[i], uv[i]) for i in range(len(V))]


Bavg = unc.ufloat(-0.0002611245724436587, 7.017367301517613e-06)
A = unc.ufloat(-1.85055977503383e-05, 2.2172178966699554e-05)



############################## Figure Stuff ####################################
#print(u)
#print(V)

savepath = '/nfs/stak/students/z/zaigerm/Homework/me552/lab2/Data/figures/'

#### OHR122
a122 = unc.ufloat(0.128165, 2.2172178966699554e-05)
b122 = unc.ufloat(1.2324, 7.017367301517613e-06)
n122 = 0.6148
#v122 = a122 + b122*umadeup**n122
u122=[None]*len(Volts)#

uraw = []
urs = [None]*50000
usum = 0.0
urms = []
Y = []

for i in range(len(Volts)):
    u122[i] = (np.abs((Volts[i] - a122)/b122))**(1/n122)

us = [val.s for val in u122]
unorm =[val.n for val in u122]
xer = 1

#v1 = [unc.ufloat(voltraw1[m], ur[0, m]) for m in range(len(voltraw1)-1)]
for i in range(len(V)):
    voltn = voltraw[i,:]
    voltcheck =  [unc.ufloat(voltn[c], ur[i,c]) for c in range(49999)]
    for j in range(49999):
        urs[j] = np.array((np.abs((voltcheck[j]-a122)/b122))**(1/n122))
        if j > 0:
            usum = urs[j-1] + urs[j] + usum
    usum = usum/50000
    usum =(usum)**0.5
    usum = usum/u122[i]
    urms.append(usum)

    #uraw.append([])
    #uraw[i].append(urs)
ynorm = [val.n for val in urms]
ys = [val.s for val in urms]
```

```
plt.figure(figsize = (8,6), dpi = 300)
plt.errorbar(x, ynorm, xerr = xer, yerr = ys, fmt ='o', label = 'Intesity')

plt.title('Low Re, High Height [OHR = 1.22]',fontsize=14)
plt.xlabel('Position (mm)', fontsize=12)
plt.ylabel('Velocity (m/s)', fontsize=12)
plt.xticks(fontsize = 14)
plt.yticks(fontsize = 14)
plt.legend(loc = 1, fontsize = 14)

plt.savefig(savepath + 'OHR122_LRe_HH_I.png')


plt.figure(figsize = (8,6), dpi = 300)
plt.errorbar(x, unorm, xerr = xer, yerr = us, fmt ='o', label = 'Velocity')

plt.title('Low Re, High Height [OHR = 1.22]',fontsize=14)
plt.xlabel('Position (mm)', fontsize=12)
plt.ylabel('Velocity (m/s)', fontsize=12)
plt.xticks(fontsize = 14)
plt.yticks(fontsize = 14)
plt.legend(loc = 1, fontsize = 14)

plt.savefig(savepath + 'OHR122_LRe_HH_u.png')
```

D Uncertainty Tree

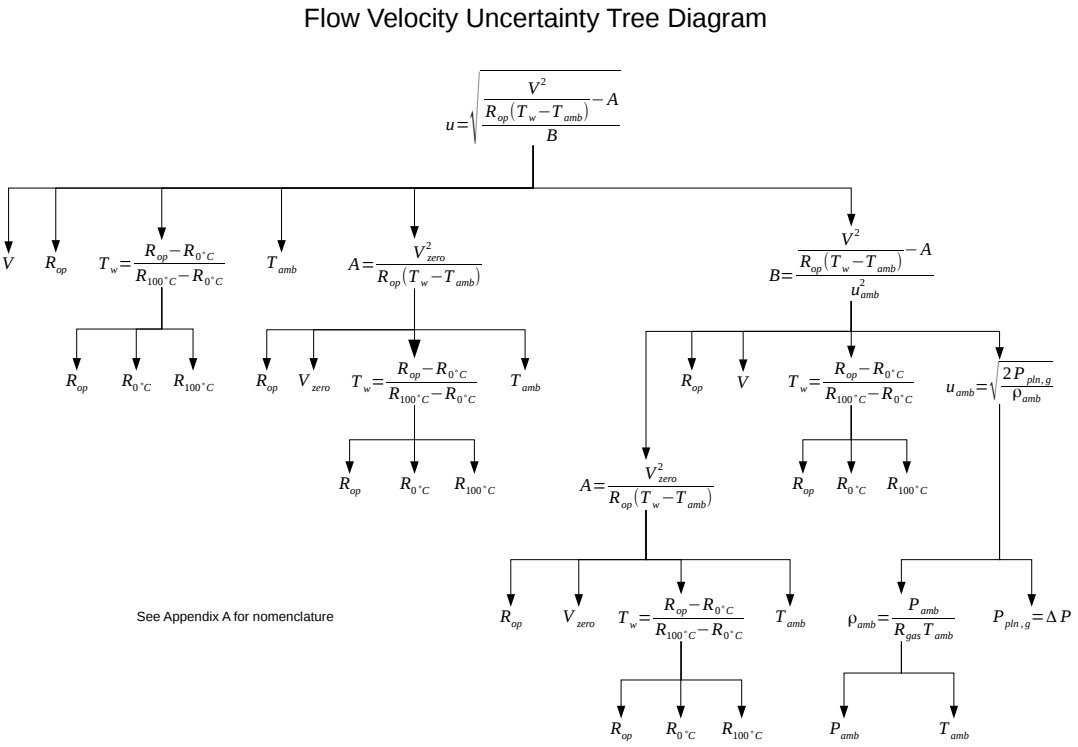


Figure 8: Flow Velocity Uncertainty Tree Diagram

Omega PCL-MA-10WC uni-directional plug-in pressure module used has a range of 0 to 10 in H20. The uncertainty was 0.06% of the instrument range, which was equal to approximately 1.5 Pascals.

A TSI 1201-6 hot wire probe was used. The probe had an resistance uncertainty value of 0.1% of the reading +/- 0.01 Ohms, and gain and offset accuracy of 0.15%.

For part one of this lab the goal is to find the velocities up on top of the tree and determine a power-law fit using 'A' and 'B' as constants. For part 2, the coefficients constants and associated error found in part one, 'A' and 'B', are used to calculate new velocities from our measured data at OHR 1.22. We introduce measurement error for each data point measured for the u_{flux} in determination of u_{rms}. This adds to very large error bars in our newly calulated velocities and turbulent intensities.

E Figures

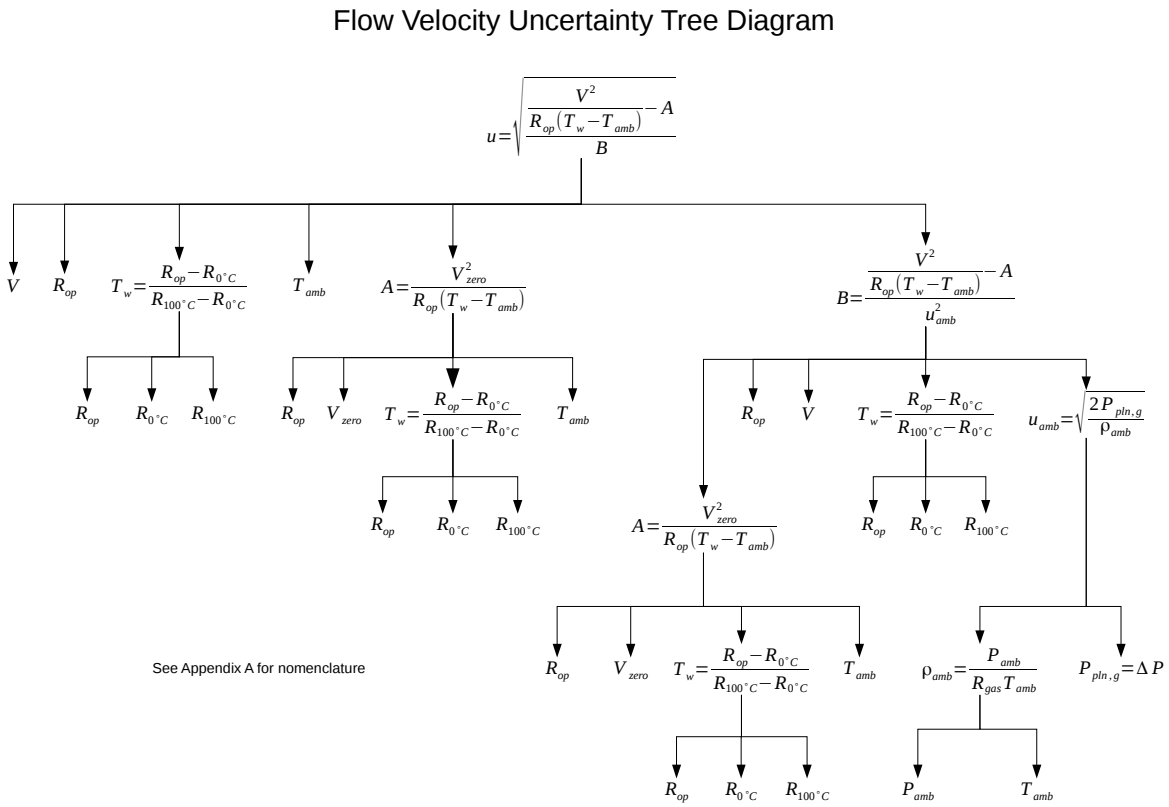


Figure 9: Flow Velocity Uncertainty Tree Diagram

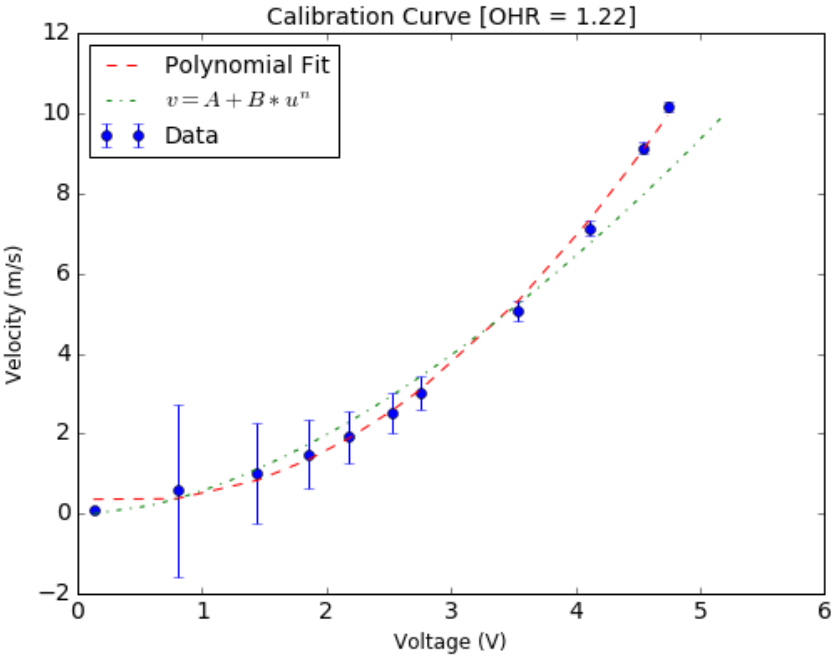


Figure 10: Calibration Curve for OHR=1.22

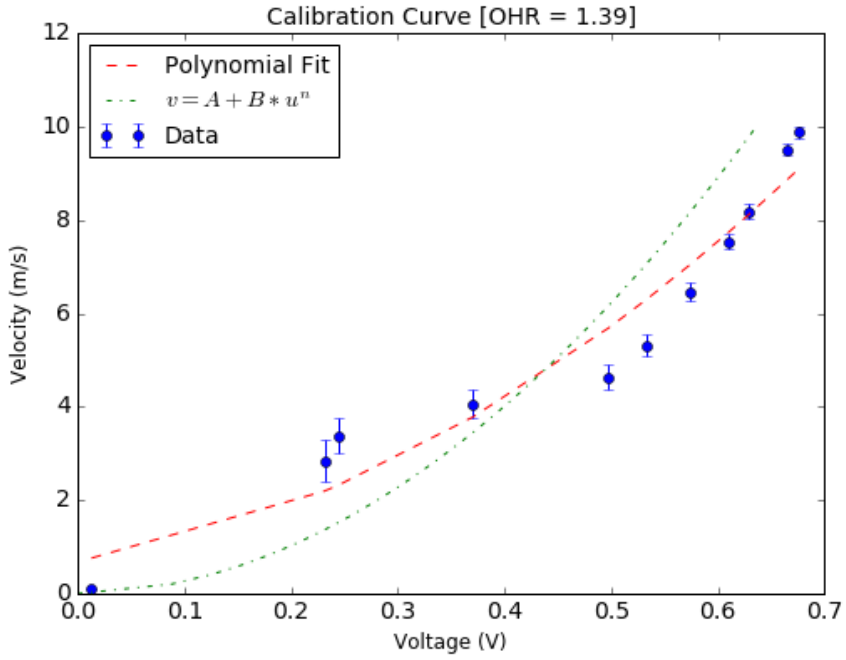


Figure 11: Calibration Curve for OHR=1.39

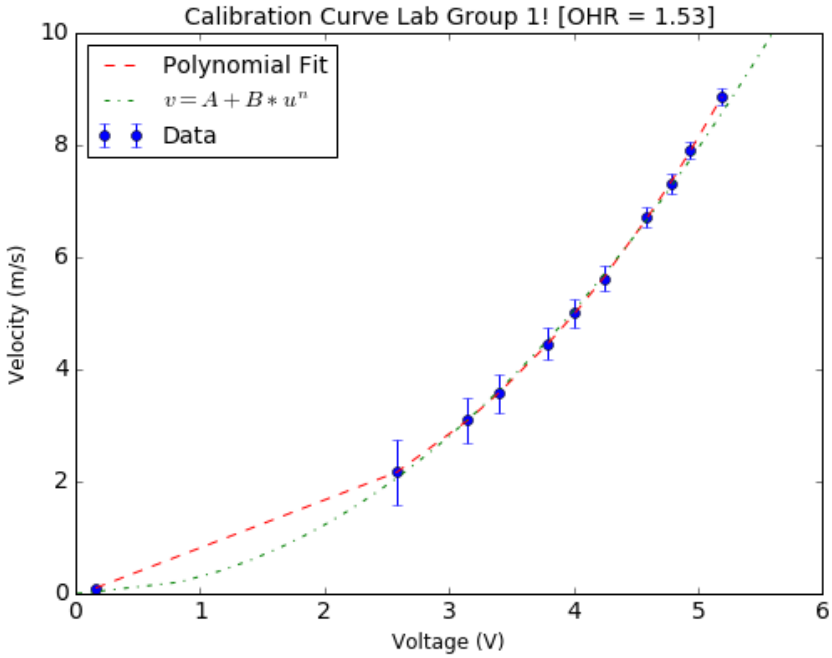


Figure 12: Calibration Curve for OHR=1.53

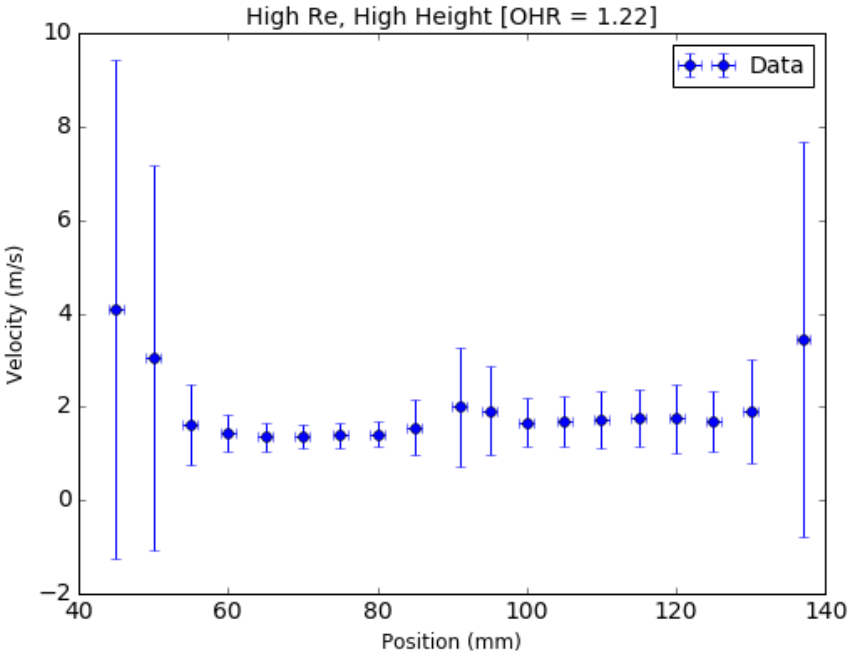


Figure 13: Turbulence Intensity, High Re, High Height

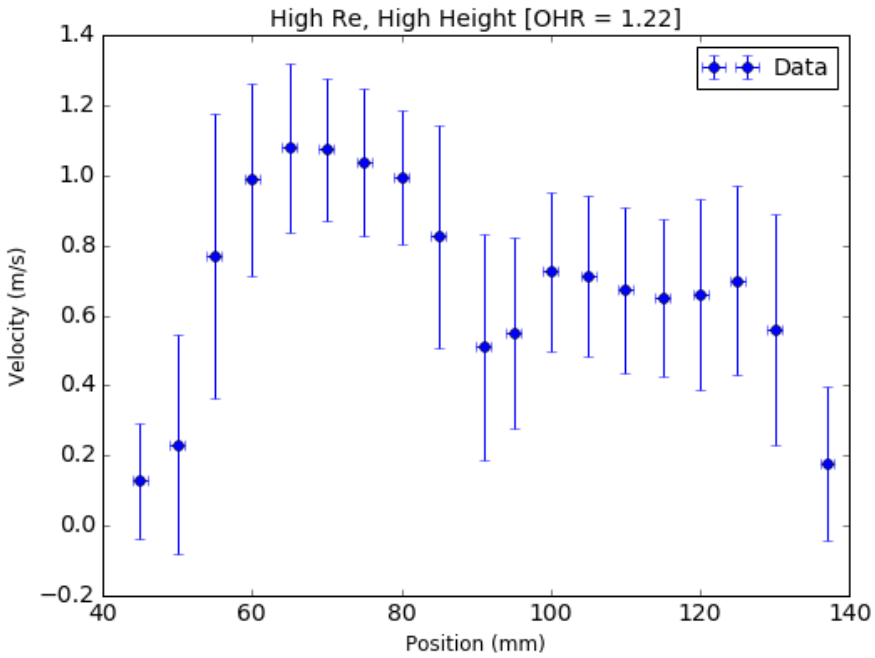


Figure 14: Velocity Profile, High Re, High Height

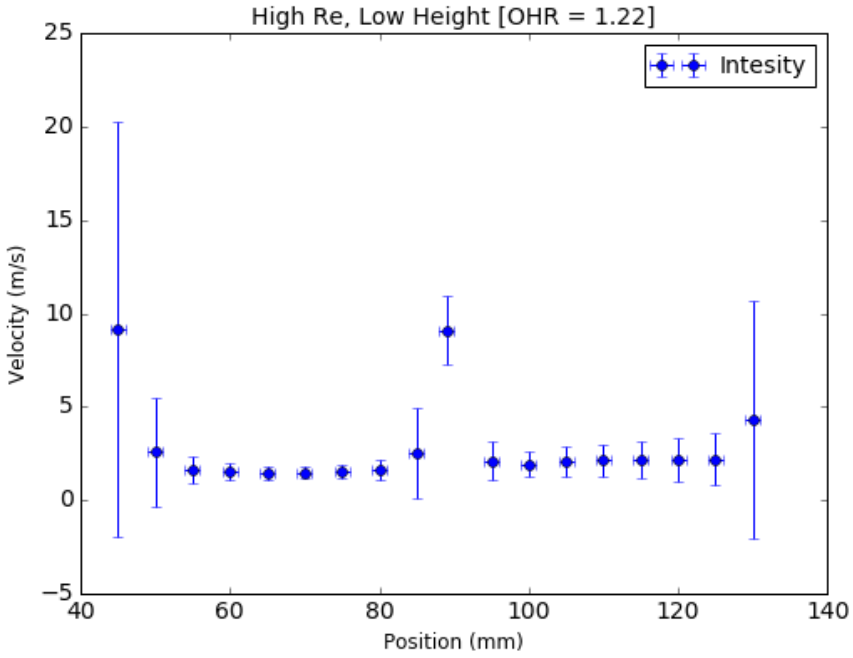


Figure 15: Turbulence Intensity, High Re, Low Height

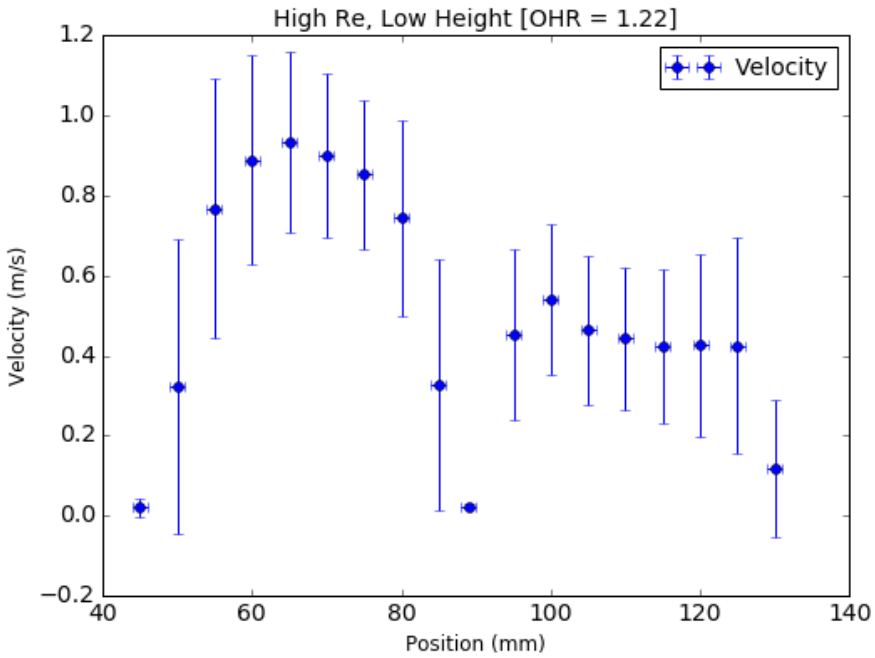


Figure 16: Velocity Profile, High Re, Low Height

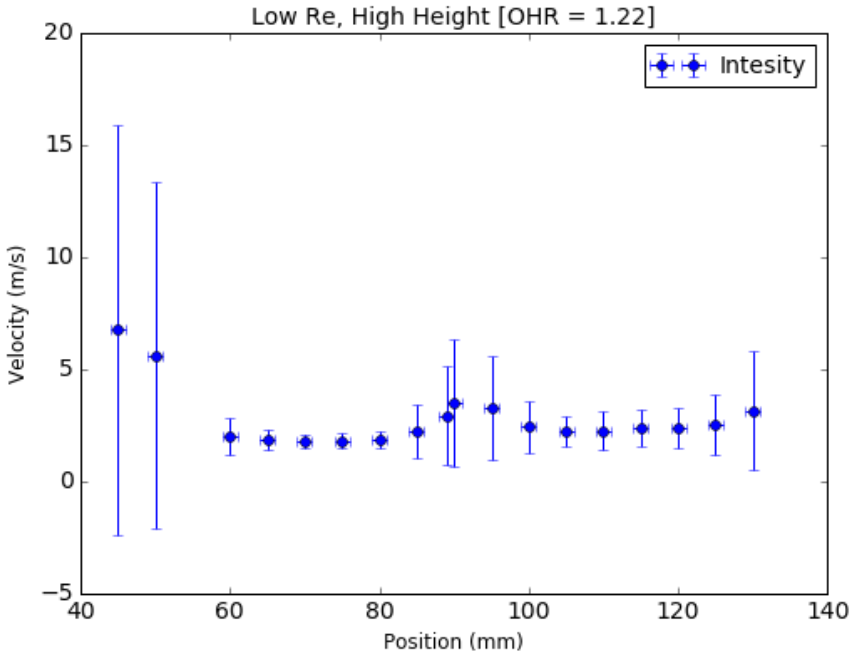


Figure 17: Turbulence Intensity, Low Re, High Height

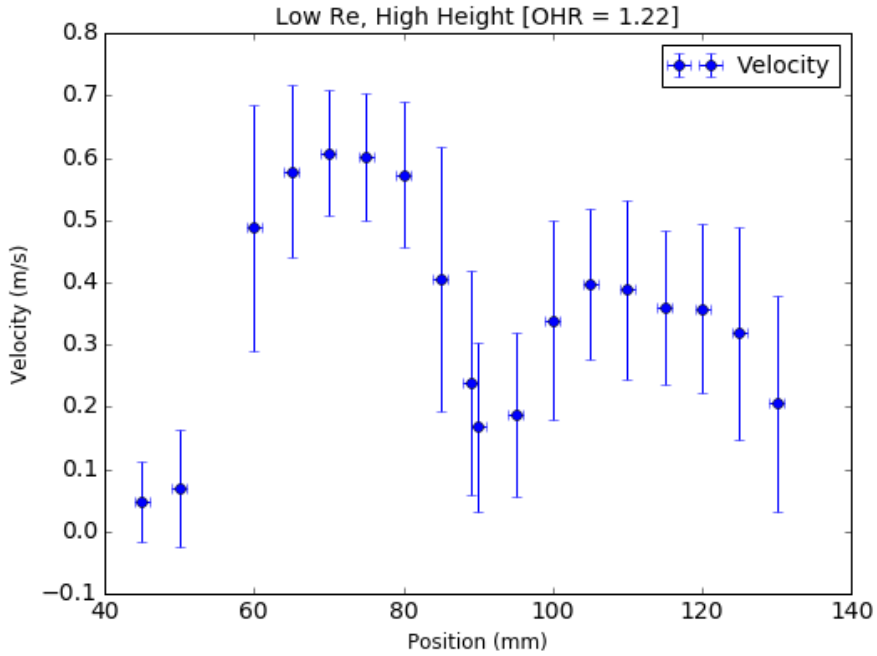


Figure 18: Velocity Profile, Low Re, High Height

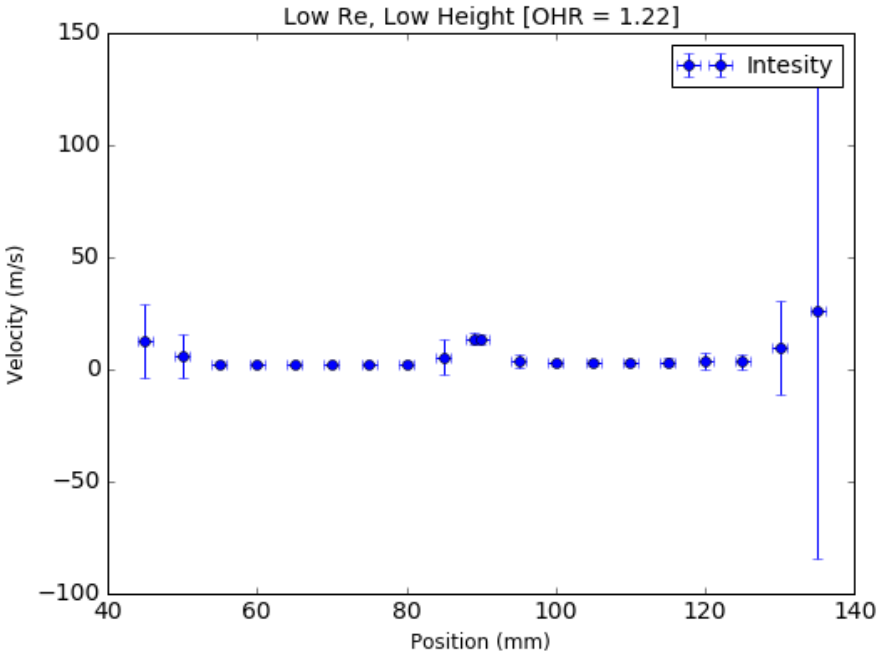


Figure 19: Turbulence Intensity, Low Re, Low Height

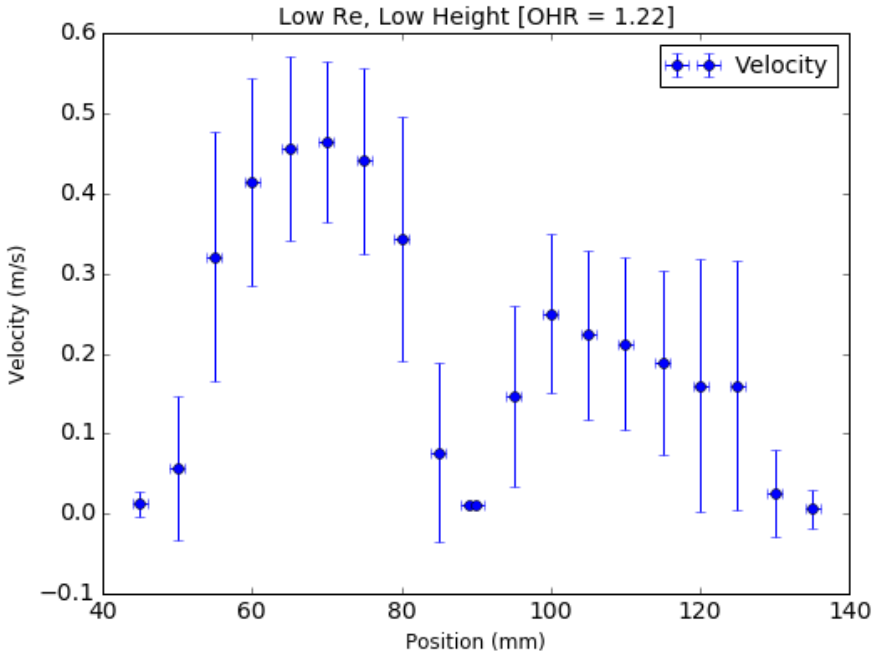


Figure 20: Velocity Profile, Low Re, Low Height

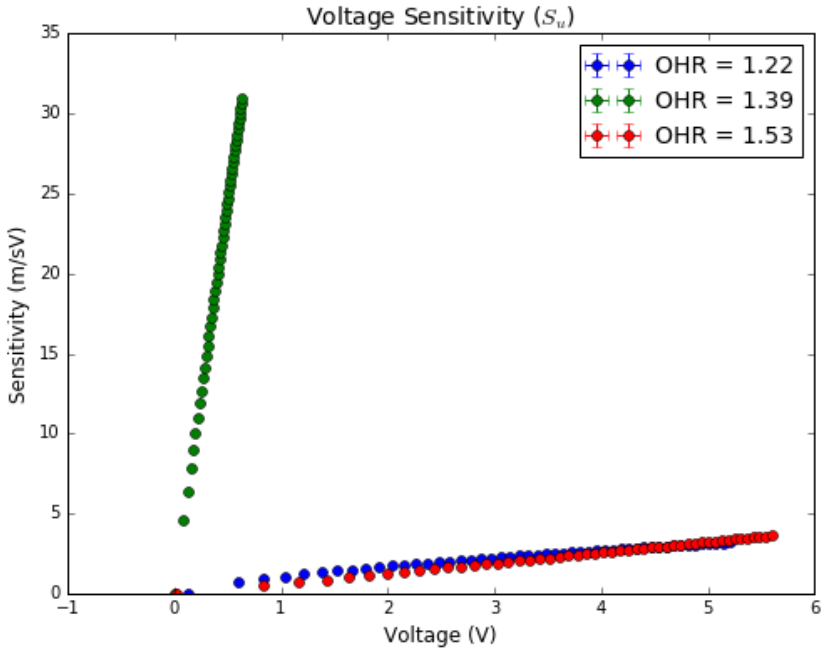


Figure 21: Voltage Sensitivity

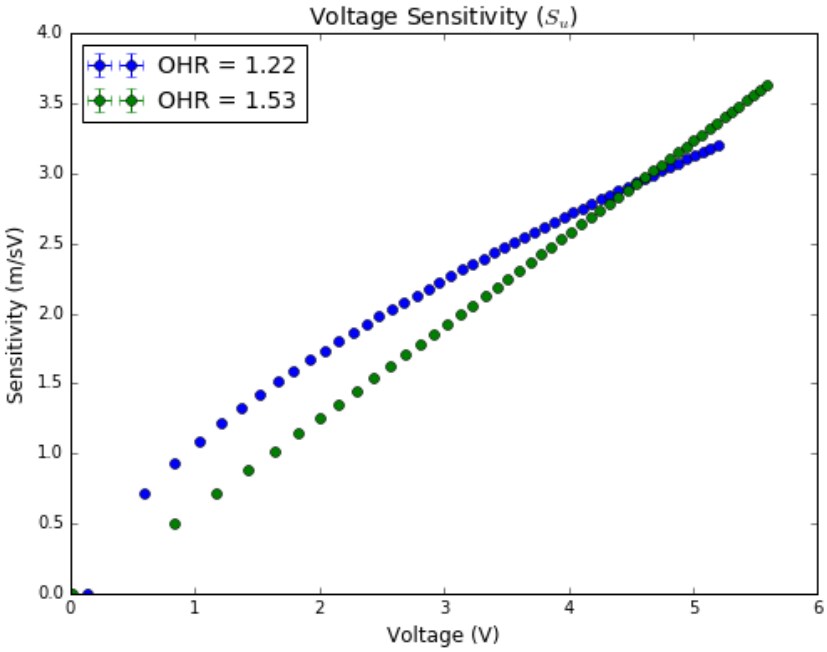


Figure 22: Voltage Sensitivity, OHR = 1.39 Removed