ME 5448 Experimental Fluid Dynamics

Two-Point Axial Velocity Correlation Study on a Free Round Turbulent Jet

Final Project Report

Authors: Lianxia Li, Brendan Taedter, David Tobin and Luke McLaughlin

Submitted on: 12/13/2019

Table of Contents

List of Figures

1. Introduction

Turbulent jet flows are one of the most commonly studied turbulent free shear flows, due to their vast applications in engineering as well as their commonality in nature. Therefore, turbulent jets are very important to our fundamental understanding of turbulence. Through two-point measurements, the integral length scale in the radial direction at the centerline of a turbulent jet will be calculated at multiple streamwise locations. This will be done by measuring the streamwise velocity at the centerline and a corresponding radial location simultaneously. The radial distance of the second point will then be increased gradually such that it can be observed where the cross-correlation of the two velocity measurements converges to zero. Repeating this process at multiple streamwise locations will allow conclusions to be made about the integral length scale at the centerline of a jet with respect to distance from the jet nozzle.

1.1 Background

Turbulent flows have many applications in the physical world and have been studied extensively in recent history. For example, information gathered from the study of a turbulent jet can prove critical in the design of a rocket engine nozzle (Georgiadis, 2006), the Diesel spray in an engine (Payri, 2015), the dispersion of ash from the plume of a volcanic eruption (Kieffer, 1984; Ogden, 2008), or the water-jet propulsion methods (Durán-Grados, 2018) of various aquatic creatures. This problem has been extensively studied by means of experiments (Kamotani, 1972) and numerical simulations (Edwards, 1979; Prabhakaran, 2019).

Two-point correlation study in the round jet has also attracted researchers' interests. Chang et.al (1985) measured two-point space time pressure velocity correlations in the near field of a round jet flow. Ewing et. Al (2007) examined the governing equations for the two-point velocity correlations in the far field of the axisymmetric jet and showed they have equilibrium similarity solutions for jets with finite Reynolds number.

A basic diagram of a turbulent jet is shown below in Fig. 1. The turbulent jet which will be the focus of this experiment is said to be free, meaning it exits into an unbounded environment, and round, meaning simply that it exits from a circular opening. The fluid issuing from the jet will be air.

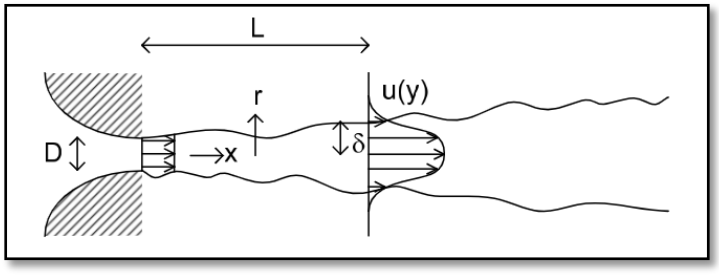


Figure 1. Cross-section of free round jet.

The simplified governing equations (Appendix 6.1) for the mean flow of a round jet are given by

$$\frac{1}{r}\frac{\partial(r\bar{u}_r)}{\partial r}+\frac{\partial\bar{u}_x}{\partial \mathrm{x}}=0 \tag{1-1}$$

$$\frac{1}{\rho}\frac{\partial\bar{P}}{\partial r}+\frac{\partial}{\partial r}\left(\overline{u_r'^2}\right)=0 \tag{1-2}$$

$$\bar{u}_r\frac{\partial\bar{u}_x}{\partial r}+\bar{u}_x\frac{\partial\bar{u}_x}{\partial x}=-\frac{1}{r}\frac{\partial\left(\overline{ru_r'u_z'}\right)}{\partial r} \tag{1-3}$$

1.2 Quantities of Interest

In order to determine the integral length scale in the radial direction at the centerline of the flow, the cross-correlation between the streamwise velocity at the centerline and the streamwise velocity at various radial locations will be calculated. These pairs of velocities must be measured simultaneously. This will also be repeated at multiple streamwise locations as indicated by Fig. 2. Therefore, the structure of the measured data will be in the form:

$$U_i(x_j,r_c,\theta,t)\,,U_i(x_j,r_c+k\Delta r,\theta,t) \tag{1-4}$$

where i ranges from 1 to N, j ranges from 1 to $j_{\max}$, and k ranges from 1 to $k_{\max}$. A visual representation of these measurement locations is portrayed in Fig. 2 on the following page.

The cross-correlation of these quantities is then defined by

$$R_{uu}(k\Delta r)=\overline{u_i(x_j,r_c,\theta,t)\,u_i(x_j,r_c+k\Delta r,\theta,t)} \tag{1-5}$$

where $u_i(x_j,r_c,\theta,t)$ and $u_i(x_j,r_c+k\Delta r,\theta,t)$ are the fluctuating components of the two velocity measurements described above. This quantity will also be normalized by $\sigma_1\sigma_2$ such that $|R_{uu}|\leq 1$. A quantitative criterion must also be established to determine when the cross-correlation has indeed converged to zero, so that the same criteria can be applied at each streamwise location. In this study, the selected criteria were simply to apply a linear interpolation between the first instance where the cross-correlation becomes negative, and the preceding (positive) cross-correlation.

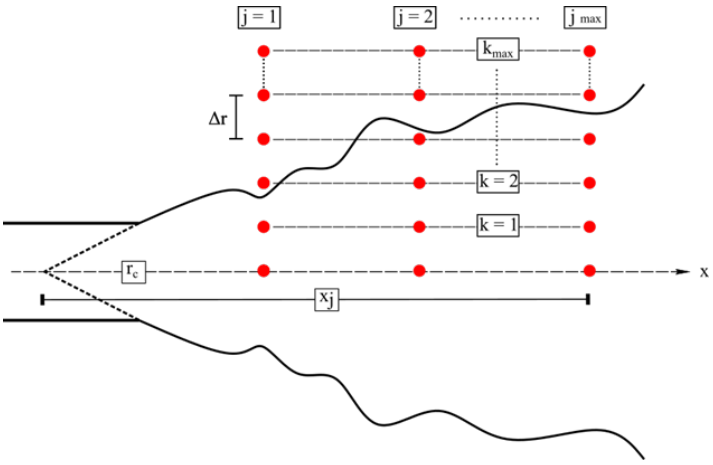


Figure 2. Velocity measurement locations.

The effective number of times each pair of points is measured (N_{eff}) must be sufficiently high since the product of the streamwise velocity fluctuations will be time averaged to calculate the corresponding cross-correlation. Therefore the turbulence intensity of the flow (Tu), the integral time scale of the flow (I), and the desired precision error (ε) will be approximated such that the total sample time (T), and the corresponding optimal effective sampling rate ($f_{s,eff}$) can be calculated:

$$T = \frac{2ITu^2}{\epsilon^2} \tag{1-6}$$

$$N_{eff} = \frac{T}{2I} \tag{1-7}$$

$$f_{s,eff} = \frac{N_{eff}}{T}. \tag{1-8}$$

The number of streamwise locations considered (j_{max}) will not need to be very high. This is because a high-resolution profile of the integral length scale along the streamwise direction is not necessary. Rather, data at just a few streamwise locations will allow conclusions to be drawn about the general behavior of the integral length scale at the centerline of the jet as distance from the nozzle increases.

The number of points along the radius that will be considered (k_{max}) will need to be high enough such that the profile of the cross-correlation along the radius of the jet at a given streamwise location will be sufficiently fine to determine where it approximately goes to zero per the established criteria, but low enough such that the total amount of data that needs to be taken does not exceed the time constraints of the project. Some intermediate post-processing will therefore likely be necessary in order to determine how many radial locations need to be measured. For example, one possibility may be to take course measurements ranging from very close to the centerline to very far from the centerline (ensuring that the cross-correlation has indeed converged to zero), and then to go back and measure more points close to where the cross-correlation appears to go to zero, allowing for a more precise estimate of the integral length scale at that streamwise location to be made. Similarly, the selection of the radial spacing (Δr) will be a function of how many measurements can be taken in the allotted time, as well as the physical constraints of the instrumentation and experimental setup.

The velocity correlation is expected to decrease as the radial distance increases and the integral length scale is expected to increase as the streamwise position increase in distance from the jet. This study will provide a great deal of insight into the nature of turbulence in a free jet.

2. Experimental Setup

2.1 Test Facility, Hardware, and Methods

The facility utilized for this experiment was the Optics and Instrumentation Lab on the second floor of Engineering. The room was large enough that the walls' influence on the pressure distribution within the room was deemed negligible. This allows for the assumption that the jet exits into a quiescent environment.

Equipment used for producing and taking data from the jet was provided by Dr. Naughton's graduate laboratory. The jet itself had a circular nozzle of diameter of 1cm and could maintain back pressures of up to approximately 100psi. This pressure was easily adjusted by a valve on the jet, meaning the exit velocity of the jet could be adjusted to the desired test condition. In this study, the testing conditions included a jet exit velocity of U = 51.56 m/s (driven by a 1.3 kPa pressure difference between the inside and outside of the jet) which corresponds to a Reynolds Number of $Re_D = \rho UD/\mu \approx 28{,}000$ where D = 1cm is the diameter of the nozzle, ρ = 0.9773 kg/m^3 and $\mu = 1.81 \times 10^{-5}$ kg/ms (properties at elevation of Laramie, WY). The jet was mounted on a traversal system which could be manually adjusted along the streamwise axis.

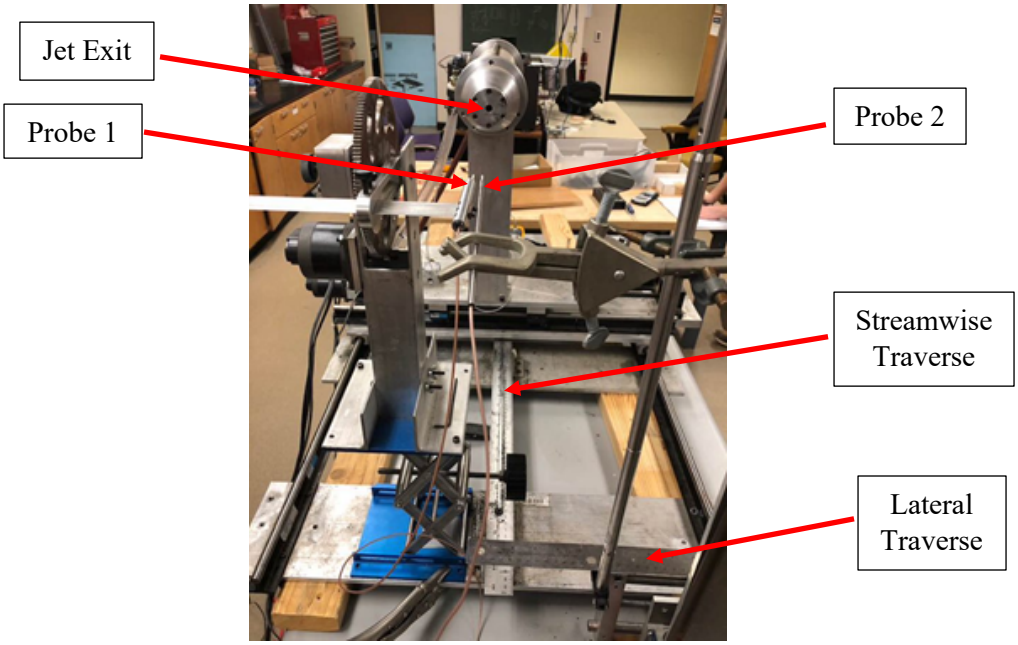


Figure 3. Front view of jet and HWA configuration.

Two constant temperature hot-wire anemometers (HWAs) were used (see ***Models and Instrumentation*** for more details) to analyze the flow. It was desired to keep one probe fixed at the centerline of the jet, and to place the other probe at varying radial distances for each streamwise distance tested. Probe 1 (centerline probe) was held in place by an adjustable sheath. Probe 2 (variable probe) was held by a ring stand clamp, which in turn was attached to a manual traverse

such that it could be moved laterally away from the jet centerline. Fig. 3 and Fig. 4 show the described experimental setup.

Measurements were taken at three streamwise distances: 30 cm ($30D_{jet}$), 39.5 cm, and 49 cm. These values are evenly spaced at 9.5 cm apart; it would have been desirable to have an even 10 cm spacing, but the streamwise traverse maxed out at 49 cm. For each streamwise location, the probes began with a separation of 5 mm, as this was the distance between the wires when the probe bodies were nearly in contact. Probe 2 was then moved radially outward in 1mm increments until the lateral traverse was maxed out and data was collected at each radial position. In total, Probe 2 was moved a total distance of 47 mm, or 52 mm from Probe 1. For a visual representation of the measurement locations, please refer to Fig. 2, with $j = 3, k_{max} = 47$, and $\Delta x = 1$ mm.

2.2 Instrumentation and Sampling

Constant temperature hot wire anemometry (HWA) was used to simultaneously determine the streamwise velocity of the jet flow at two different radial points having the same streamwise location. Constant temperature HWA was chosen as the measurement technique because it had a high signal to noise ratio, was sensitive to a wide range of flows, had a well understood calibration and the wire burns out infrequently compared to constant current HWA. Two single-wire constant temperature HWA probes operated by a single Hot Wire and Film Anemometer device made by A. A. Lab Systems LTD. were utilized in this study in order to simultaneously measure streamwise velocities. The output signal of each HWA probe wire in Volts was filtered first prior to sampling at 400 Hz to avoid aliasing (< fs/2) using a Stanford Research Systems, Inc. Model SR640 Lowpass Filter. The filtered probe signals were then simultaneously sampled using a National Instruments (NI) BNC-2110 board and LabVIEW at a sampling frequency f_s = 1 kHz for a sampling period T = 10 s. Using the sampling frequency and sampling period according to Eqn. 2-1 through Eqn. 2-3, the number of independent samples was chosen to be N_{eff} = 10,000 for each probe, resulting in a mean velocity precision uncertainty of 0.2 % for an estimated turbulence intensity of 20 %. The samples were independent because the sampling frequency necessary to obtain independent samples was determined to be 2.578 kHz, and was calculated as

$$f_{s_{required}} = \frac{1}{2I} = 2.578 \text{ kHz} \tag{2-1}$$

where I, the integral time scale, was approximated to be

$$I = \frac{D}{U} = \frac{1\ cm}{51.56\frac{m}{s}} = 1.94 \times 10^{-4} \text{ s} \tag{2-2}$$

where D = 1 cm is the jet outlet diameter and U = 51.56 m/s is the mean jet velocity at the outlet during testing determined from Bernoulli's principle. After acquiring 10,000 samples at each measurement location using the prescribes sampling frequency, statistics were computed on the signals in order to quantify further results such as cross-correlations and integral length scales.

A single DAQ assistant was used in LabVIEW to simultaneously acquire signals from two channels while the NI DAQ board was run in 'Ground Source' mode. Because a single DAQ assistant was used to simultaneously acquire signals from two channels, the signals of the two channels were multiplexed and demultiplexed prior to being displayed in LabVIEW. See Appendix

6.2 for the LabVIEW block diagram used for data acquisition, and Fig. 4 below for a view of the instrument setup.

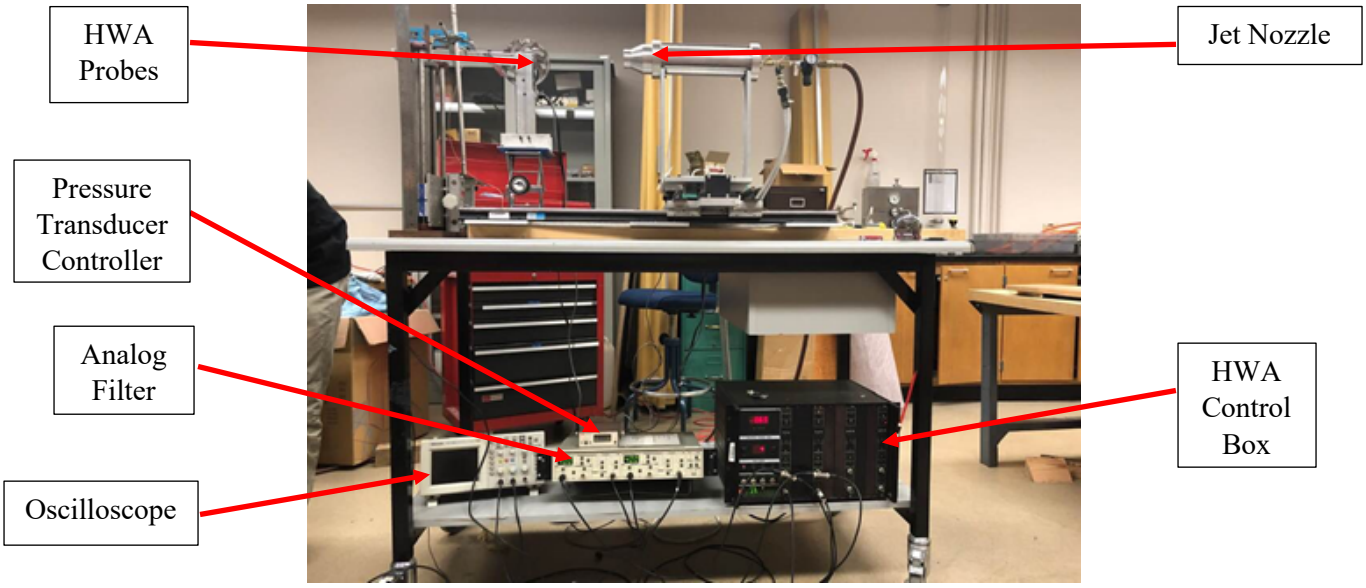


Figure 4. Side view of instrument setup, computer not pictured.

2.3 Instrument Calibration

The two probes used in this work were tuned and calibrated in order to relate the measured voltages of the hot wires to velocity measurements. During the probe tuning processes, the cable and wire resistances of each channel were accounted for, the desired overheat ratio of 1.6 was applied, the damping response was adjusted to maximize frequency response, and the gain and DC offset of the system were set. The ambient wire resistances were determined to be 4.60 Ω and 4.70 Ω for probe 1 and probe 2 respectively. So, the corresponding operating resistances of the wires determined from the overheat ratio for constant temperature HWA were 7.36 Ω and 7.52 Ω for the fixed and moving probes respectively. The damping response of each probe was adjusted to minimize the response time of each wire, however, for the purposes of this study the exact damping response time was not of interest due to the slow nature of data acquisition herein. The DC offset of each channel was set so that the 'ambient' wire voltage reading while in operating mode was 0 ± 0.5 V, and the gain multiplier setting was set to "off" to ensure that the output voltage of each wire did not exceed -5 V (minimum voltage accepted by Stanford Research Systems, Inc. filter) during testing conditions.

After tuning each probe, a pressure calibration was performed for the transducer in order to relate transducer voltage signals to known pressure differences applied to the transducer. This calibration was required to determine the mean jet velocity at the outlet of the jet for calibration of the hot wires where hot wire voltage was related to fluid velocity in the streamwise direction. Prior to performing the pressure transducer calibration, the pressure transducer was zeroed and spanned

for a 0-2 kPa pressure range. This range was determined from a desired maximum velocity at the jet outlet of about 60 m/s. Once zeroed and spanned, known pressure differences were applied to the pressure transducer using a FLUKE 719-30G pressure calibrator. The average output voltage of the transducer for an applied pressure was acquired in LabVIEW by taking the average of 1,000 samples sampled at 1 kHz. This process was repeated for several pressure differences between 0-2 kPa and the voltage measurements were plotted vs. their corresponding applied pressure differences. These measurements, as well as the corresponding linear fit, are shown in Fig. 5. The slope of the curve was then used to relate pressure transducer voltages to mean velocities at the jet outlet.

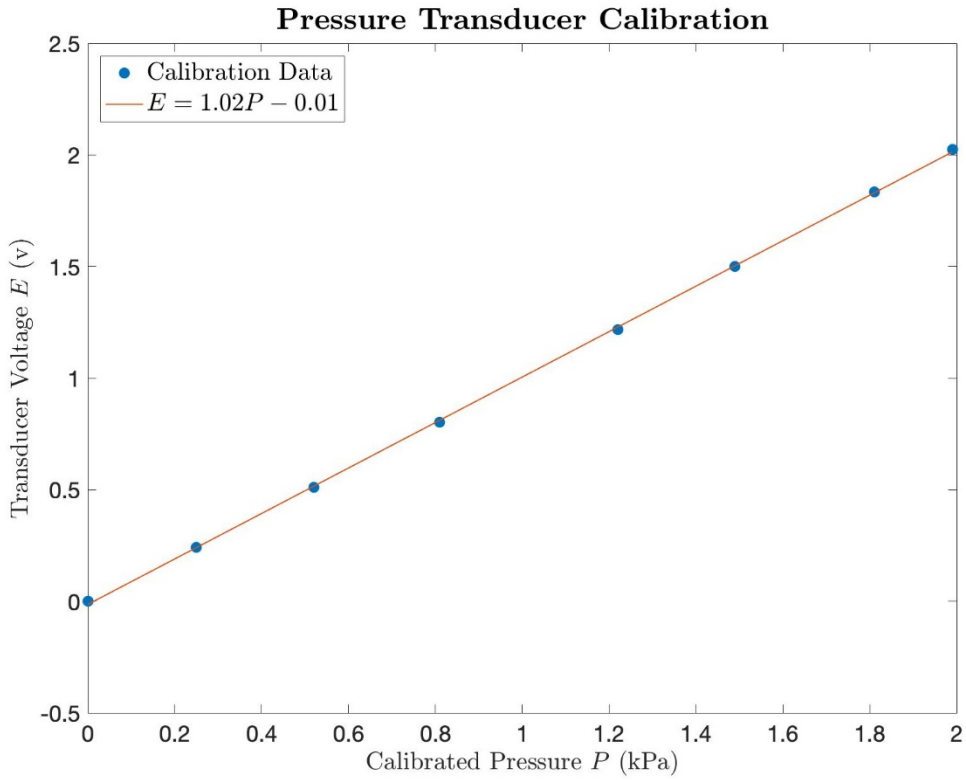
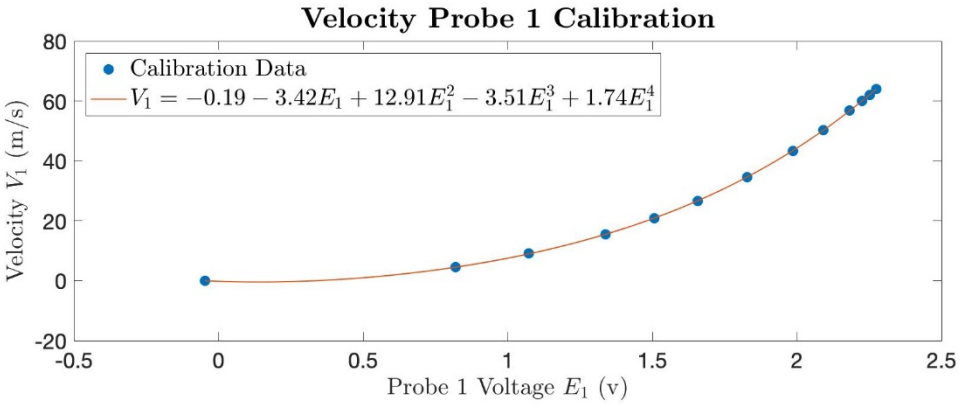


Figure 5. Pressure transducer calibration.

Next, the voltage responses of each constant temperature hot wire probe were related to velocity measurements via a polynomial calibration approach. The probe to be calibrated was placed directly at the midpoint of the jet outlet. The jet was turned on and the pressure difference between the inside of the jet and atmospheric pressure was registered by the pressure transducer. Using the previous pressure transducer calibration, the voltage measurement of the pressure transducer was related to the corresponding mean velocity at the jet outlet. The mean velocity at the jet outlet was then related to the corresponding hot wire probe voltage measurement. The voltage output by the hot wire probe was sampled at 1 kHz for 1s and the average was computed. The average voltage was then plotted vs the mean velocity applied for a range of mean fluid velocities. A fourth order polynomial fit was applied to the established Voltage vs. Velocity curve for each probe and the deduced polynomial was used to relate voltage measurements to velocity measurements for each

respective probe. The calibration plots for Probe 1 and Probe 2 are shown in Fig. 6 a) and b) respectively.

a)



b)

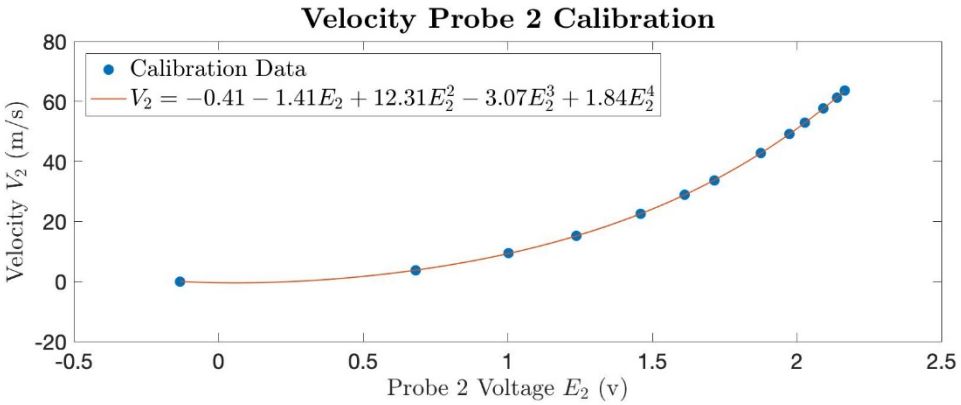


Figure 6. HWA velocity calibration for a) Probe 1 and b) Probe 2.

3. Results and Discussion

3.1 Analysis of Results

After completing the necessary calibration procedures described in section 2.3, velocity data was taken at the three streamwise distances of $x = 30, 39.5,$ and 49(cm) from the jet. At each streamwise location, simultaneous pairs of velocity measurements were taken according to Eqn. 1-3 and Fig 2. with $j = 3, k_{max} = 47$, and $\Delta r = 1$ (mm), after the initial 5 (mm) spacing of the two hot wire probes. For each pair of points, $N_{eff} = 10{,}000$ effective samples were taken at a sampling frequency of $f_s = 1\ kHz$ in order to meet the statistical requirements and precisions discussed in section 2.2.

The first analysis that can be considered with the resulting velocity data is the resulting radial velocity profile at each streamwise distance. This is easily found by averaging the result of U_r/U_c over all 10,000 measurements for each pair of points, where U_r is the velocity at the radial distance $r = k\Delta r$ and $U_c$is the velocity at the jet centerline, i.e. $r = 0$. Figure 7 shows the resulting profiles of streamwise velocity in the radial direction, at each streamwise location.

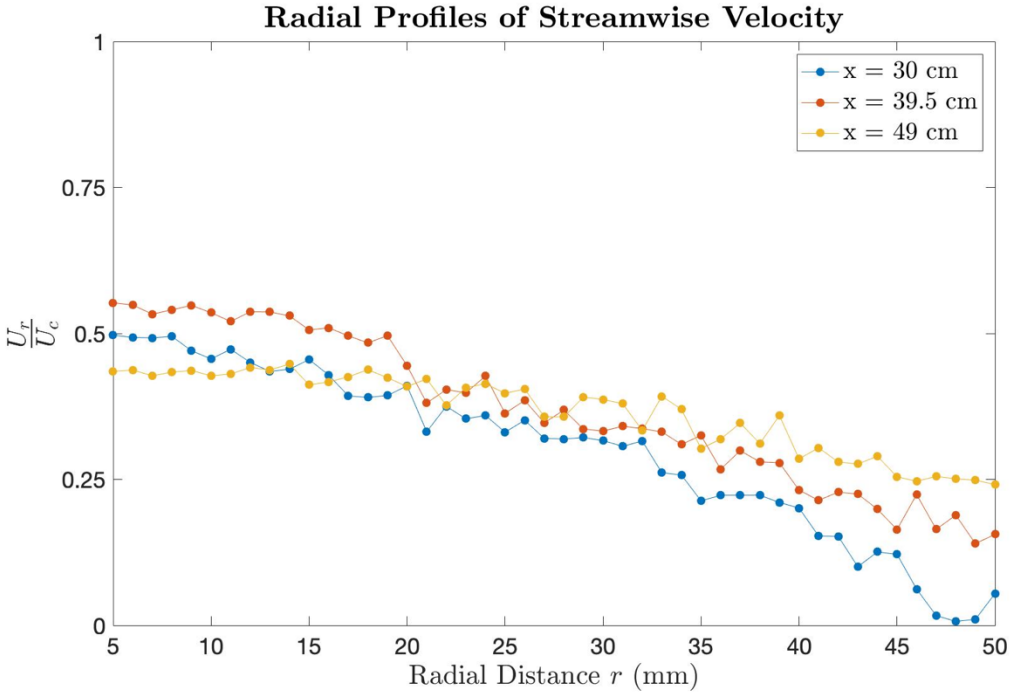


Figure 7. Radial profiles of streamwise velocity.

Two main conclusions can be drawn from Fig 7. The first is that, in general, the jet is behaving as expected. That is to say, it is expanding as it progresses downstream. This is seen by the steeper gradient of the profile at $x = 30$ (cm) whereas the velocity gradient is more gradual at $x = 49$ (cm). The second important conclusion from these velocity profiles is that, at all of the streamwise distances considered, the streamwise velocity already drops to around half of the corresponding streamwise velocity at the jet centerline within the first 5 (mm) of the jet radius. Recall that this is the zone that cannot be measured due to the discussed physical constraints of the dual hot wire

probe approach. This therefore provides an initial warning that there will be a corresponding gap in the cross-correlation profile, which could make it difficult to decide where the cross-correlation converges to 0 and thus approximate the integral length scale. However, it is shown below that this is ultimately not a significant problem, since the cross-correlation clearly does not converge to 0 until well after the initial 5 (mm) measurement gap.

The cross-correlation of each pair of points was computed via a "brute force" approach. That is to say, each pair of $N_{eff} = 10{,}000$ effective samples result in a single cross-correlation by simply applying the definition shown in Eqn. 1-4. The cross-correlation is also normalized by the product of the standard deviation of each velocity measurement ($\sigma_1\sigma_2$) such that the absolute value of the cross-correlation is always ≤ 1. This process was repeated for all radial pairs and at all streamwise locations.

The resulting integral length scale is then approximated by the radial distance where the cross-correlation has "converged" to 0. It is known however that the cross-correlation will continue to slightly fluctuate around 0 at distances well beyond the integral length scale. Therefore, the convergence criterion was simply defined as the first location where the cross-correlation becomes negative. Numerically, this was found by linearly interpolating between the first instance where the cross-correlation becomes negative and the immediately preceding positive cross-correlation. The corresponding results for each streamwise location are shown in Fig. 8.

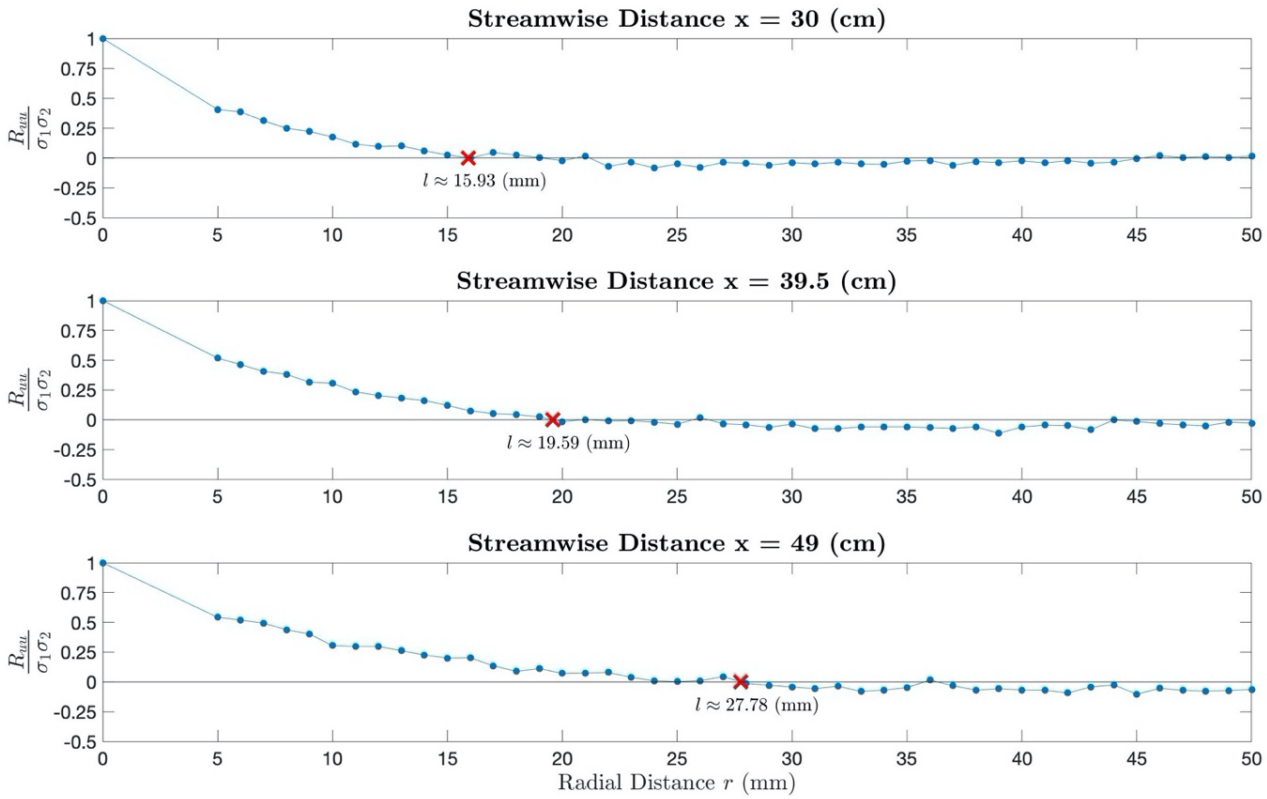


Figure 8. Cross-correlation profiles.

As expected, these profile shows that the cross-correlation between the velocity measurements of the two probes do decrease as the distance between the measurement locations increases. Also, as previously discussed, the cross-correlation profile has a gap in the first 5 (mm) of the jet radius due to the physical constraints of the two hot-wire probes. However, it is possible to add the initial point where the normalized cross-correlation is exactly equal to 1 at the centerline. This is a direct result from the definition shown in Eqn. 1-4. In other words, this additional point is a consequence of the cross-correlation between probe 1 and itself being exactly equal to 1. This additional point, combined with the original profile that has a resolution of 1 (mm) beginning at 5 (mm), strongly suggests that the cross-correlation does not become negative within the first 5 (mm) of the jet radius. This indirectly shows that the utilized dual hot wire probe approach was appropriate for this test, as the 5 (mm) gap does not prevent an approximation of the integral length scale from being made.

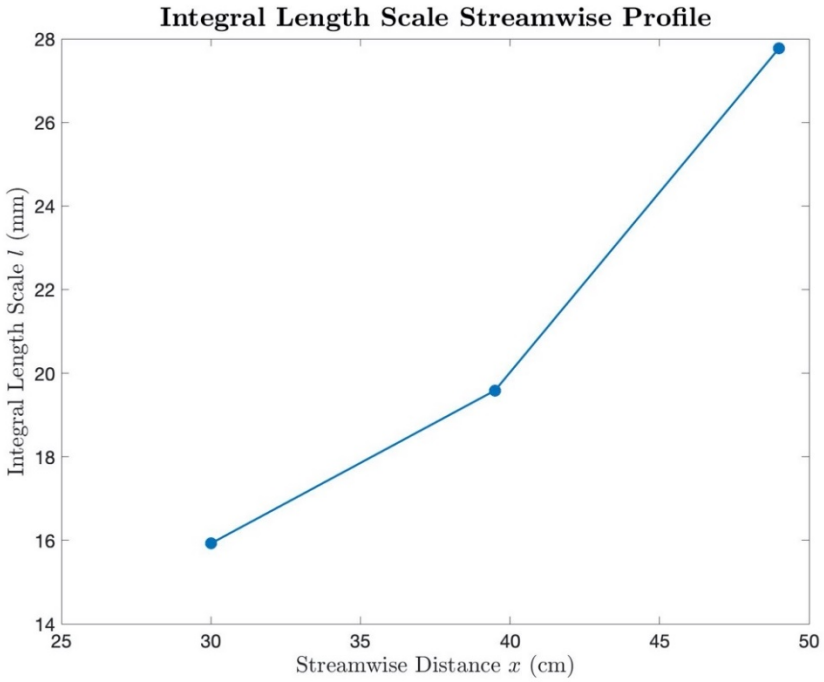


Figure 9. Integral length scale profile.

The major conclusion that can be made from Fig. 8 is simply that the resulting approximate integral length scale increases as streamwise distance from the jet nozzle increases. This is emphasized in Fig. 9 by plotting the resulting integral length scales along their corresponding streamwise distances. Although only three points are available in this study, it is clear that the cross-correlation increases with streamwise distance, even it if the exact behavior of the relationship is not clear. This conclusion will be validated with a comparison to existing literature in the next section. However, qualitatively speaking, it is easy to understand that this is the expected result. Fig. 7 verified that the jet diameter is increasing as the streamwise distance increases. Therefore, the turbulent structures themselves are also expanding as they propagate downstream. This means that for two simultaneous velocity measurements to be independent, the distance that they must be

separated by (i.e. the integral length scale) will increase as the streamwise distance from the jet nozzle increases.

3.2 Comparison to Literature

An important part of any experiment is to compare the results obtained with results of prior studies. Such a comparison will either provide strong validation for the present experimental results or highlight some discrepancies which should be addressed further. The two main results from this study were the radial velocity profiles at three streamwise distances, and the evolution of the integral length scale in the radial direction at the centerline of the jet. Cushman-Roisin (2008) showed the radial velocity profile for a round turbulent jet decays quickly when near to the jet exit, and more gradually as the measurement location is moved downstream [12]. See Fig. 10 below. Although this result is not directly comparable to the velocity profile results found in this study (Fig. 7) due to different normalization methods and greatly differing Reynolds numbers, the qualitative meaning behind both results is still the same. That is to say, in both results, the gradient of the velocity becomes more gradual as the streamwise distance increases, thus verifying the expanding behavior of the jet.

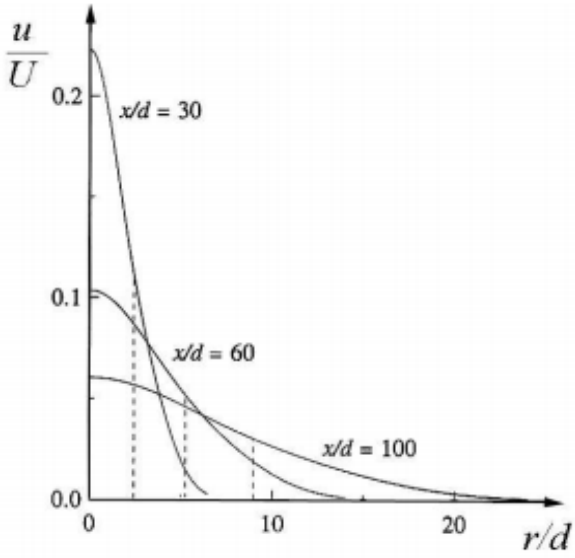


Figure 10. Radial profiles of mean axial velocity in a turbulent round jet. [12]

Khashehchi et al. (2013) showed that the integral length scale in the radial direction at the centerline of a round, turbulent jet increases with increasing streamwise distance from the jet exit. See Fig. 11 below. Again, there are several key differences between the present study and the work of Khashechi et al., namely in Reynolds number (28,000 vs. 3,000) and in streamwise measurement locations ($^{x}/_{d} = 30$ to 49 vs $^{x}/_{d} = 1$ to 25). Khashehchi et al. was also specifically interested in considering the region of transitional turbulence, whereas this study considered fully develop turbulence. However, comparing with this study is still valuable in that it confirms, at the minimum, that the integral length scale does increase with increasing streamwise

distance. Khashehchi et al. is also able to provide data at more than 3 streamwise locations, and the resulting data is still a relatively linear relationship comparable to what was found in Fig. 9.

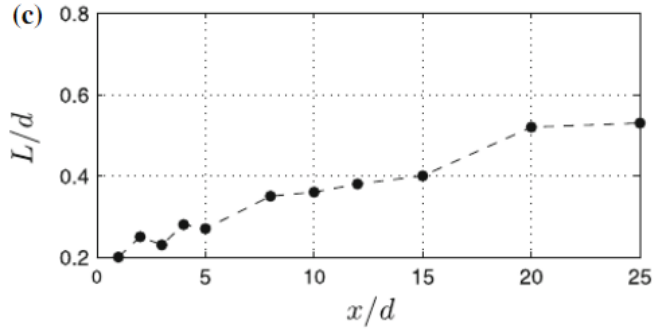


Figure 11. Streamwise evolution of the integral length scale, L, for turbulent round jet. [13]

3.3 Discussion of Errors and Uncertainty

The uncertainty and error arising in measurements in this study were due to the combination of bias errors and precision errors. Bias errors that arose were due to errors in calibrations, hot wire tuning, imperfect probe alignment to the direction of the flow and imperfect centering of the fixed probe at the centerline of the flow. The HWA calibration processes introduced bias errors by relating hot wire voltage measurements to velocity measurements with a polynomial fit having a corresponding fit accuracy, as well as by using the calibration in streamwise locations other than the exact position where it was calibrated. Bias errors also could have arisen from imperfect balancing of the Wheatstone bridge for each hot wire probe. Normal alignment of the probes to the flow direction was strived for, however, imperfect probe alignment likely occurred and contributed to the bias error. Similarly, the fixed probe (Probe 1) was intended for perfect alignment at the centerline of the jet flow, however, if imperfect centering of Probe 1 occurred then an additional bias error was contributed to the total error in the measurements.

Precision errors that contributed to the total error in the measurements made in this study included digitizing errors, room air fluctuations, and pressure variations in the building air handling unit which provided the lab room with pressurized air used in the jet. The digitizer had 16-bit resolution which contributed to the precision error of the measurements by discretizing continuous signal measurements into discrete signal bins. The air handling of the lab room where tests were conducted was controlled by a centralized HVAC unit which induced minor air currents in the room. These undesired air currents contributed to the precision bias of the measurements made. Additionally, the building air handling unit which supplied pressurized air to the jet would intermittently turn on and off throughout the duration of the tests performed herein. This caused the applied pressure difference between the room and inside of the jet to fluctuate (±0.05 kPa), and resulted in additional precision errors in the velocity measurements. Although not every source of bias and precision error was identified, the errors were accounted for in the statistics computed for each velocity measurement at various radial positions.

4. Conclusions and Future Work

4.1 Conclusions

In conclusion, the absolute value of the cross-correlation of centerline and radial streamwise velocity measurements decreases as the radial position of the unfixed probe increases. The correlation starts high when the two probes are nearly in the same location, and decays rapidly as the unfixed probe is traversed radially. An approximation of the length scale in the radial direction can therefore be made by determining the radial distance at which the cross-correlation between the two velocity measurements has sufficiently converged. The resulting length scale is seen to increase as the distance of the probes from the jet increases in the streamwise position. This observation of length scale increasing with increasing streamwise position agrees with the similar (but not identical) turbulent jet studies considered [13]. This behavior is ultimately explained by the fact that the turbulent jet is expanding as it propagates down the streamwise direction. Consequently, the turbulent structures themselves are expanding, and thus the integral length scale also increases.

4.2 Computational Considerations

The computational problem for turbulent shear flows is to represent a wide enough range of scales in the computer to resolve the underlying physical mechanisms. To conduct the numerical simulation of a round jet flow, an appropriate CFD model representing the same physical problem in the experiment must be chosen first, especially a proper turbulence model. We may choose RANS models (k-e, k-w) or LES models for this specific problem.

We need to consider the spatial and temporal resolutions based on how small the size of the eddy interested is and how accurately we want to capture the evolution of turbulence structures. Experimental data can help us get the eddy scale and time scale, then we can decide the time step and space size to discretize the governing equations and geometry domain. Meanwhile, the mesh quality can be increased by refining or adjusting the mesh of the zones with high gradients based on the experimental data. In addition, the computational domain must be large enough to ensure that the boundaries are far away from and do not affect the free jet flow significantly. For the round jet problem, we can convert the 3D problem to a 2D one by taking advantage of the axis-symmetry of the domain.

Experiments can provide boundary conditions for the numerical simulation. For instance, if the inlet flow is not uniform, we can use the experimental data to set the inlet condition. Pressure boundary conditions may also be set from experimental data.

Verification and validation are very important to do a successful CFD simulation and must always be addressed with an emphasis on the quantification of the uncertainties due to the model assumptions (either physical or geometrical), and to the numerical and experimental approximations. Verification assessment determines how accurately a computational simulation

represents a given conceptual model. It examines the mathematics in the models through comparison to experimental results. Validation assessment determines if the computational simulation agrees with physical reality. It examines the science in the models through comparison to experimental results.

4.3 Future Work

A reasonable extension to this study would be to repeat a similar process where the velocity measurement location at the boundary of the jet is held stationary, and the second simultaneous velocity measurement is gradually moved towards the centerline of the jet. This would provide information about the integral length scale at the boundary of the jet, which could also be compared at multiple streamwise locations and to the centerline integral length scale. Additionally, more than three streamwise distances could be tested, such that a more quantitative relationship between integral length scale and streamwise distance could be determined. It would also be possible to study the integral length scales in the streamwise and radial directions, though this would require a velocity measurement instrument other than HWA, such that simultaneous velocity measurements are not being taken in the wake of other instrumentation.

5. References

1) Nicholas J. Georgiadis, James R. DeBonis. **Navier–Stokes analysis methods for turbulent jet flows with application to aircraft exhaust nozzles**. Progress in Aerospace Sciences, Volume 42, Issues 5–6, July–August 2006, Pages 377-418.
2) A.V.J. Edwards, C.L. Morfey. **A computer simulation of turbulent jet flow.** Computers & Fluids. Volume 9, Issue 2, June 1981, Pages 205-221.
3) Christophe Bailly, Geneviève Comte-Bellot. **Turbulence (Experimental Fluid Mechanics).** Springer, 2015.
4) Raul Payria, Jose M. García-Oliver, Tiemin Xuan, Michele Bardi. **A study on diesel spray tip penetration and radial expansion under reacting conditions.** Applied Thermal Engineering. Volume 90, 5 November 2015, Pages 619-629.
5) Yasuhiro Kamotani, Isaac Greber. **Experiments on a Turbulent Jet in a Cross Flow.** AIAA Journal Vol. 10, No. 11, November 1972, Pages1425-1430.
6) Prasanth Prabhakaran, Sachin Shinde, Roddam Narasimha. **A DNS Study of entrainment in an axisymmetric turbulent jet as an episodic process.** arXiv:1907.05421v1 [physics.flu-dyn], 2019.
7) Susan Werner Kieffer, Bradford Sturtevant. **Laboratory Studies of Volcanic Jets.** Journal of Geophysical Research, Vol. 89, No. B10, Pages 8253-8268, September 30, 1984.
8) Darcy E. Ogden, Kenneth H. Wohletz, Gary A. Glatzmaier. Emily E. Brodsky. **Numerical simulations of volcanic jets: Importance of vent overpressure.** Journal of Geophysical Research: Solid Earth, Volume 113, Issue B2, 2008.
9) Vanesa Durán-Grados, Javier Mejías, Liliya Musina, JuanMoreno-Gutiérrez. **The influence of the waterjet propulsion system on the ships' energy consumption and emissions inventories.** Science of The Total Environment, Volumes 631–632, 1 August 2018, Pages 496-509
10) D. Ewing, B.Frohnapfel, W.K.George, J. M. Pedersen, J.Westerweel. **Two-point similarity in the round jet.** J. Fluid Mech. (2007), vol.577, pp.309–330.
11) Chang, P. H P; Adrian, Ronald; Jones, B. G. **Two-point pressure-velocity correlations, conditional average and linear estimation in a round jet flow.** IN: FLUID CONTROL AND MEASUREMENT, (TOKYO, JAPAN: SEP. 2-6, 1985), M. HARADA (ED.). Vol. 1, Oxford, U.K., Pergamon Press, 1986, p.481-491. Pergamon Press, 1986.
12) Cushman-Roisin, B., **Turbulent Jets**. Environmental Transport and Fate, Dartmouth College, February 2008.
13) Khashehchi, M., Ooi, A., Soria, J., Marusic, I. **Evolution of the turbulent/non-turbulent interface of an axisymmetric turbulent jet.** Experiments in Fluids, January 2013.
14) M. Oberlack. **Analysis of the two-point velocity correlations in turbulent boundary layer.** Center for Turbulence Research Annual Research Briefs, 1995, Pages 209-220.

6. Appendices

6.1 Simplified RANS Equations for Round Jet

In a turbulent flow, the velocity field is random. To study the free shear turbulence flow, we start from the Navier-Stokes equations, i.e., the mass conservation and momentum balance equations; then the Reynolds averaging technique is introduced to decompose the instantaneous flow into mean and fluctuating components. This means the Reynolds mean equations (RANS equations) are obtained with additional turbulent stress terms, and the additional stress terms introduce a closure problem. To gain insight from the RANS equations, simplification of the flow must be made. The simplifications of turbulent jet flow are made below, following a discussion of flow assumptions.

Flow Assumptions

For many fluid dynamics experiments, it is necessary to make some simplifying assumptions, and then confirm the validity of said assumptions. The following assumptions will be made about the flow for this experiment:

Newtonian: It is commonly accepted that air, the fluid with which the experiment will be performed, is assumed to be Newtonian for practical calculations under ordinary conditions.

Incompressible: The fundamental requirement for incompressible flow is that the density is constant within a small elemental volume dV, which moves at the flow velocity u. This assumption can be validated by checking if the divergence of the flow velocity is zero, or if the Mach number of the flow is less than 0.3.

Statistically stationary: While turbulent flows are unsteady by definition, turbulent flow can be statistically stationary. A flow is statistically stationary if all statistics are invariant under a shift in time. Since data will be collected over a certain time interval, calculating statistics at different moments in time will determine if this assumption is valid.

Axisymmetric: An axisymmetric flow is a flow in which the streamlines are symmetrically located around an axis, i.e. every longitudinal plane through this axis exhibits the same streamline pattern. This assumption is typically applied to cylindrical or round geometries, such as a jet from a round nozzle. This assumption can be validated by analyzing and comparing the flow at different radial angles around the center axis of the jet.

Homogeneous: In homogeneous turbulence all derivatives with respect to any direction of the correlations vanish, i.e., all the quantities $\overline{u_r'^2}, \overline{u_\theta'^2}, \overline{u_x'^2}, \overline{u_r'u_\theta'}, \overline{u_\theta'u_x'}, \overline{u_r'u_\theta'u_x'}$, ... are invariant under arbitrary translations of the coordinate axes. This is equivalent to assuming that the statistics of the turbulent flow are not a function of space. In the case of the turbulent jet, the turbulence is assumed to be homogeneous in the azimuthal direction, and thus $\overline{u_r'u_\theta'} = \overline{u_x'u_\theta'} = 0$. This assumption can be validated with measurements and is a reasonable assumption if the turbulence is at Kolmogorov scale and the Reynolds number is large enough. In this situation the characteristics are only determined by dissipation ratio but not relevant to viscosity.

Quiescent environment outside of jet: Though inviscid flow does not exist, in terms of calculations, we tend to neglect freestream viscous effects at Mach numbers below 0.3 because Bernoulli's principle can then be applied, which seems to provide answers with enough accuracy. This essentially means that the fluid surrounding the jet is initially at rest, so that it will not interfere with the natural formation of the jet. This assumption can be validated, before the jet is created, by ensuring the velocity of the air in the region the jet will occupy is nearly zero.

Full steady RANS equations in cylindrical coordinates

The full steady RANS equations in cylindrical coordinates (r, θ, x) are given by:

Averaged Continuity Equation:

$$\frac{\partial \bar{u}_x}{\partial x} + \frac{1}{r}\frac{\partial (r\bar{u}_r)}{\partial r} + \frac{\partial \bar{u}_\theta}{\partial \theta} = 0 \tag{A-1}$$

Averaged Momentum Equation in Radial Direction:

$$\left[\bar{u}_r \frac{\partial \bar{u}_r}{\partial r} + \frac{\bar{u}_\theta}{r}\frac{\partial \bar{u}_r}{\partial \theta} - \frac{\bar{u}_\theta{}^2}{r} + \bar{u}_x \frac{\partial \bar{u}_r}{\partial x}\right] = -\frac{1}{\rho}\frac{\partial \bar{P}}{\partial r} - \left[\frac{1}{r}\frac{\partial}{\partial r}\left(r\,\overline{u_r'^2}\right) + \frac{1}{r}\frac{\partial}{\partial \theta}\left(\overline{u_r' u_\theta'}\right) + \frac{\partial}{\partial z}\left(\overline{u_r' u_z'}\right) - \frac{\overline{u_\theta'^2}}{r}\right] + \nu\left(\nabla^2 \bar{u}_r - \frac{\bar{u}_r}{r^2} - \frac{2}{r^2}\frac{\partial \bar{u}_\theta}{\partial \theta}\right) \tag{A-2}$$

Averaged Momentum Equation in Azimuthal Direction:

$$\left[\bar{u}_r \frac{\partial \bar{u}_\theta}{\partial r} + \frac{\bar{u}_\theta}{r}\frac{\partial \bar{u}_\theta}{\partial \theta} + \bar{u}_x \frac{\partial \bar{u}_\theta}{\partial x} + \frac{\bar{u}_r \bar{u}_\theta}{r}\right] = -\frac{1}{\rho r}\frac{\partial \bar{P}}{\partial \theta} - \left[\frac{1}{r^2}\frac{\partial}{\partial r}\left(r^2 \overline{u_\theta' u_r'}\right) + \frac{1}{r}\frac{\partial}{\partial \theta}\left(\overline{u_\theta'^2}\right) + \frac{1}{r}\frac{\partial}{\partial x}\left(\overline{u_\theta' u_x'}\right)\right] + \nu\left(\nabla^2 \bar{u}_\theta - \frac{\bar{u}_\theta}{r^2} + \frac{2}{r^2}\frac{\partial \bar{u}_r}{\partial \theta}\right) \tag{A-3}$$

Averaged Momentum Equation in Axial Direction:

$$\left[\bar{u}_r \frac{\partial \bar{u}_x}{\partial r} + \frac{\bar{u}_\theta}{r}\frac{\partial \bar{u}_x}{\partial \theta} + \bar{u}_x \frac{\partial \bar{u}_x}{\partial x}\right] = -\frac{1}{\rho}\frac{\partial \bar{P}}{\partial x} - \left[\frac{1}{r}\frac{\partial}{\partial r}\left(r\overline{u_r' u_x'}\right) + \frac{1}{r}\frac{\partial}{\partial \theta}\left(\overline{u_\theta' u_x'}\right) + \frac{\partial}{\partial x}\left(\overline{u_x'^2}\right)\right] + \nu\nabla^2 \bar{u}_x \tag{A-4}$$

where $\nabla^2 = \frac{1}{r}\frac{\partial}{\partial r}\left(r\frac{\partial}{\partial r}\right) + \frac{1}{r^2}\frac{\partial^2}{\partial \theta^2} + \frac{\partial^2}{\partial x^2}$.

Simplification of the cylindrical RANS equations

The instantaneous quantities can be decomposed into the mean and fluctuating part, that is $u_r = \bar{u}_r + u_r'$, $u_\theta = \bar{u}_\theta + u_\theta'$, $u_x = \bar{u}_x + u_x'$, $p = \bar{P} + p'$. For a round jet, one has $\bar{u}_\theta = 0$ but $u_\theta' \neq 0$, therefore, the averaged mass conservation equation (A-5) is obtained by simplifying Eq (A-1) by dropping out the term $\frac{\partial \bar{u}_\theta}{\partial \theta}$.

The simplifications of momentum equations are primarily based on dimensional analysis. Length $(L, \delta_l)$and velocity (U, V) scales in the axial and radial directions respectively, are introduced to examine the relative amplitude of each term. The scale of the fluctuating velocity vector $\boldsymbol{u}'$ is represented by a scalar u. The boundary-layer approximation requires that $\delta_l/L < 1$. Variations of the mean flow are smaller in the axial direction than in the radial one, $\frac{\partial}{\partial x} \ll \frac{\partial}{\partial r}$. The equation of mass conservation (A-1) implies that

$$\frac{1}{r}\frac{\partial (r\bar{u}_r)}{\partial r} + \frac{\partial \bar{u}_x}{\partial x} = 0 \quad \Longrightarrow \quad \frac{V}{\delta_l} \sim \frac{U}{L} \quad or \quad V \sim \frac{\delta_l}{L}U. \tag{A-5}$$

Meanwhile, the gradient and Laplace operators can be simplified to $\bar{\boldsymbol{u}} \cdot \nabla = \bar{u}_r \frac{\partial}{\partial r} + \bar{u}_x \frac{\partial}{\partial x}$ and $\nabla^2 = \frac{1}{r}\frac{\partial}{\partial r}\left(r\frac{\partial}{\partial r}\right) + \frac{\partial^2}{\partial x^2}$ by dropping out the derivatives of $\frac{\partial}{\partial \theta}$; owning to the symmetry of the mean

turbulent flow, the turbulence must be homogeneous in the azimuthal direction and thus, $\overline{u_r' u_\theta'} = \overline{u_x' u_\theta'} = 0$.

Considering the averaged Naiver-Stokes equation in the axial direction (all the terms with $\frac{\partial}{\partial \theta}$ and $\bar{u}_\theta$ are dropped out with the expansions of $\bar{\boldsymbol{u}} \cdot \nabla$ and $\nabla^2 \bar{u}_x$, one has

$$\left[\bar{u}_r \frac{\partial \bar{u}_x}{\partial r} + \bar{u}_x \frac{\partial \bar{u}_x}{\partial x}\right] = -\frac{1}{\rho}\frac{\partial \bar{P}}{\partial z} - \frac{1}{r}\frac{\partial\left(r\overline{u_r' u_x'}\right)}{\partial r} - \frac{\partial \overline{u_x'^2}}{\partial x} + \nu\left[\frac{1}{r}\frac{\partial}{\partial r}\left(r\frac{\partial \bar{u}_x}{\partial r}\right) + \frac{\partial^2 \bar{u}_x}{\partial x^2}\right], \tag{A-6}$$

$$\left[\frac{U^2}{L}; \quad \frac{U^2}{L}\right] \quad \frac{U^2}{L}\left(\frac{\delta_l}{L}\right) \quad \frac{U^2}{L}\left(\frac{\delta_l}{L}\right)^2 \quad \nu\left(\frac{U}{\delta_l^2}; \quad \frac{U}{L^2}\right).$$

Divided by U^2/L, the magnitudes of these terms are

$$[1; \quad 1] \quad \frac{\delta_l}{L} \quad \left(\frac{\delta_l}{L}\right)^2 \quad \left(\left(\frac{L}{\delta_l}\right)^2 \frac{1}{Re}; \quad \frac{1}{Re}\right).$$

(1) (2) (3) (4) (5) (6)

Term (3) is negligible with respect to terms (3) since $\delta_l < L$. Similarly, only the first viscous term (5) could be retained,

$$\nu \frac{U}{\delta_l^2}\frac{L}{U^2} = \left(\frac{L}{\delta_l}\right)^2 \frac{\nu}{UL} = \left(\frac{L}{\delta_l}\right)^2 \frac{1}{Re} \sim (5), \text{ where } Re = \frac{UL}{\nu}. \tag{A-7}$$

But viscous effects are very small for usual Reynolds numbers so term (5) can also be neglected. This is a reasonable assumption that can be applied to developing free shear flows far from any wall. The averaged Navier-Stokes equation along x is thus simplified as

$$\bar{u}_r \frac{\partial \bar{u}_x}{\partial r} + \bar{u}_x \frac{\partial \bar{u}_x}{\partial x} = -\frac{1}{\rho}\frac{\partial \bar{P}}{\partial x} - \frac{1}{r}\frac{\partial\left(r\overline{u_r' u_x'}\right)}{\partial r}. \tag{A-8}$$

(1) (2) (3)

Moreover, for a turbulent flow, it is necessary that the terms (1) and (3) have the same order of magnitude. If (3) $\ll$ (1), the flow is laminar and the case (3) $\gg$ (1) is not physically acceptable. Therefore,

$$\frac{U^2}{L} \sim \frac{u^2}{\delta_l} \quad \text{or} \quad \frac{u^2}{U^2} \sim \frac{\delta_l}{L}. \tag{A-9}$$

A similar analysis can be conducted for the averaged Navier-Stokes equation in the radial direction. In terms of order of magnitude, one has

$$\left[\bar{u}_r \frac{\partial \bar{u}_r}{\partial r} + \bar{u}_x \frac{\partial \bar{u}_r}{\partial x}\right] = -\frac{1}{\rho}\frac{\partial \bar{P}}{\partial r} - \left[\frac{\partial \overline{u_r'^2}}{\partial r} + \frac{\partial}{\partial z}\left(\overline{u_r' u_z'}\right) - \frac{\overline{u_\theta'^2} - \overline{u_r'^2}}{r}\right] + \nu\left(\frac{1}{r}\frac{\partial}{\partial r}\left(r\frac{\partial \bar{u}_r}{\partial r}\right) + \frac{\partial^2 \bar{u}_r}{\partial x^2} - \frac{\overline{u_r}}{r^2}\right), \tag{A-10}$$

$$\left[\frac{\delta_l}{L}\frac{U^2}{L} \quad \frac{\delta_l}{L}\frac{U^2}{L}\right] \quad \left[\frac{u^2}{\delta_l} \sim \frac{U^2}{L} \quad \frac{\delta_l}{L}\frac{U^2}{L} \quad \frac{\delta_l}{L}\frac{U^2}{L} \cdot 0\right] \quad \nu\left(\frac{U}{\delta_l^2}; \quad \frac{U}{L^2}; \quad \frac{U}{\delta_l^2}\right).$$

(1) (2) (3) (4) (5) (6) (7)

Divided by U^2/L, the magnitudes of these terms are

$$\left[\frac{\delta_l}{L} \quad \frac{\delta_l}{L}\right] \qquad \left[1 \quad \frac{\delta_l}{L}\right] \qquad \left(\left(\frac{\delta_l}{L}\right)^2 \frac{1}{Re};\right. \qquad \frac{1}{Re}; \qquad \left.\left(\frac{\delta_l}{L}\right)^2 \frac{1}{Re}\right).$$

(1) (2) (3) (4) (5) (6) (7)

Viscous terms are again negligible, and the order of magnitude of the term (3) is larger than all the other terms, and it can be only balanced by the pressure term. Hence,

$$\frac{1}{\rho}\frac{\partial \bar{P}}{\partial r} + \frac{\partial}{\partial r}\left(\overline{u_r'^2}\right) = 0 \tag{A-11}$$

and by integration in the radial direction from the jet axis to the ambient medium, one obtains $\bar{P} + \rho\overline{u_r^2} = P_\infty$. This relation can then be used to estimate the order of magnitude of the longitudinal pressure gradient in Eq. (A-12),

$$-\frac{1}{\rho}\frac{\partial \bar{P}}{\partial \mathrm{x}} = \frac{\partial \overline{u_r'^2}}{\partial \mathrm{x}} \quad \sim \quad U^2\frac{\delta l}{L}\frac{1}{L} = \frac{\delta_l}{L}\frac{U^2}{L}. \tag{A-12}$$

The pressure term (with non-dimensional magnitude $\frac{\delta_l}{L}$) is found to be negligible compared with other terms in Eq. (A-8).

Finally, the governing equations for the mean flow of a round jet are given by

$$\frac{1}{r}\frac{\partial (r\bar{u}_r)}{\partial r} + \frac{\partial \bar{u}_x}{\partial \mathrm{x}} = 0 \tag{A-13}$$

$$\frac{1}{\rho}\frac{\partial \bar{P}}{\partial r} + \frac{\partial}{\partial r}\left(\overline{u_r'^2}\right) = 0 \tag{A-14}$$

$$\bar{u}_r\frac{\partial \bar{u}_x}{\partial r} + \bar{u}_x\frac{\partial \bar{u}_x}{\partial x} = -\frac{1}{r}\frac{\partial\left(\overline{r u_r' u_z'}\right)}{\partial r}. \tag{A-15}$$

Using the equation of mass conservation (A-13), the left-hand side of (A-15) can be rewritten as

$$\bar{u}_r\frac{\partial \bar{u}_x}{\partial r} + \bar{u}_x\frac{\partial \bar{u}_x}{\partial x} + \bar{u}_x\left(\frac{1}{r}\frac{\partial (r\bar{u}_r)}{\partial r} + \frac{\partial \bar{u}_x}{\partial x}\right) = \frac{1}{r}\frac{\partial (r\bar{u}_r\bar{u}_x)}{\partial r} + \frac{\partial (\bar{u}_x\bar{u}_x)}{\partial x}$$

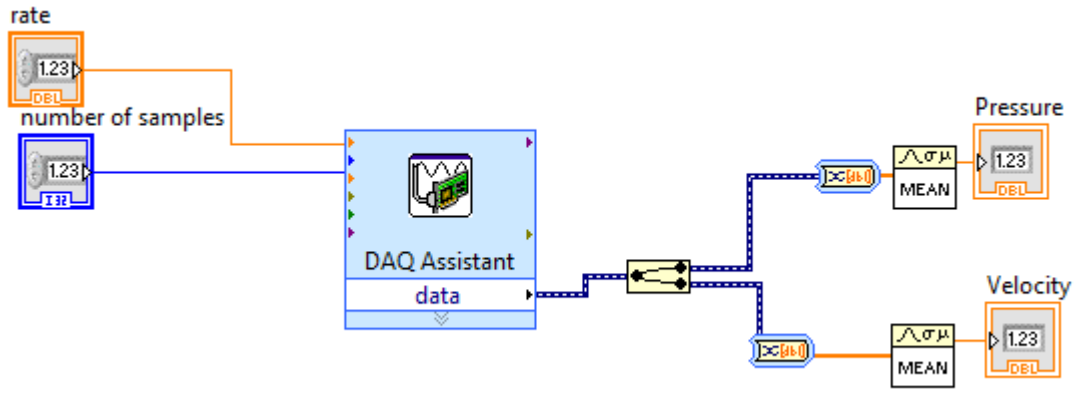
Therefore, (A-15) becomes

$$\frac{1}{r}\frac{\partial (r\bar{u}_r\bar{u}_x)}{\partial r} + \frac{\partial (\bar{u}_x\bar{u}_x)}{\partial x} = -\frac{1}{r}\frac{\partial\left(\overline{r u_r' u_z'}\right)}{\partial r} \tag{A-16}$$

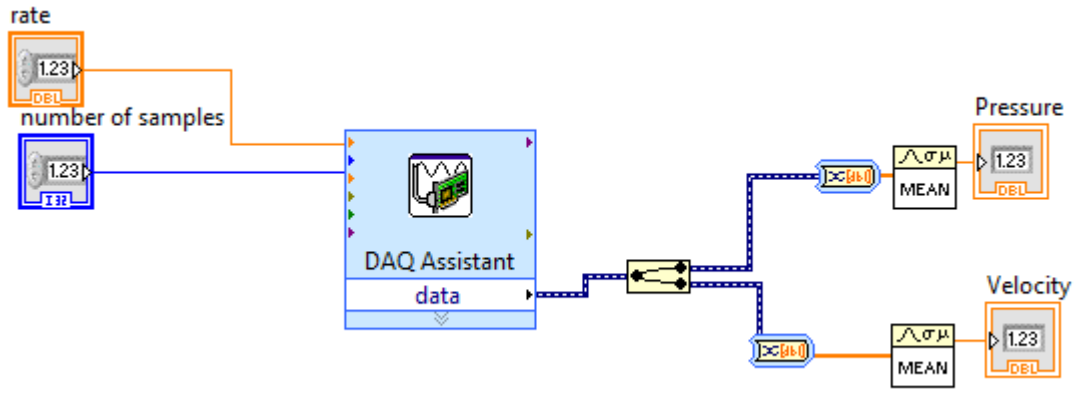
6.2 LabVIEW Block Diagrams

Rudimentary LabVIEW programs were used to simply acquire voltage readings. Block diagrams of these simple programs are provided below. All other calibrations, conversion, calculations, etc. were then performed in MATLAB (Appendix 6.3).

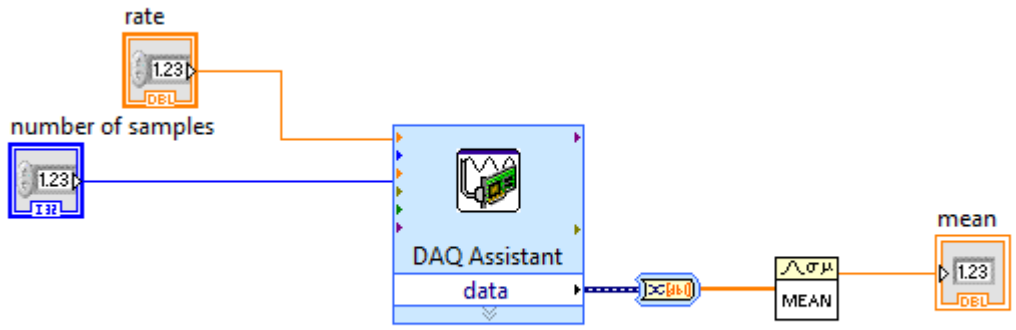
Probe 1 Calibration LabVIEW Code:



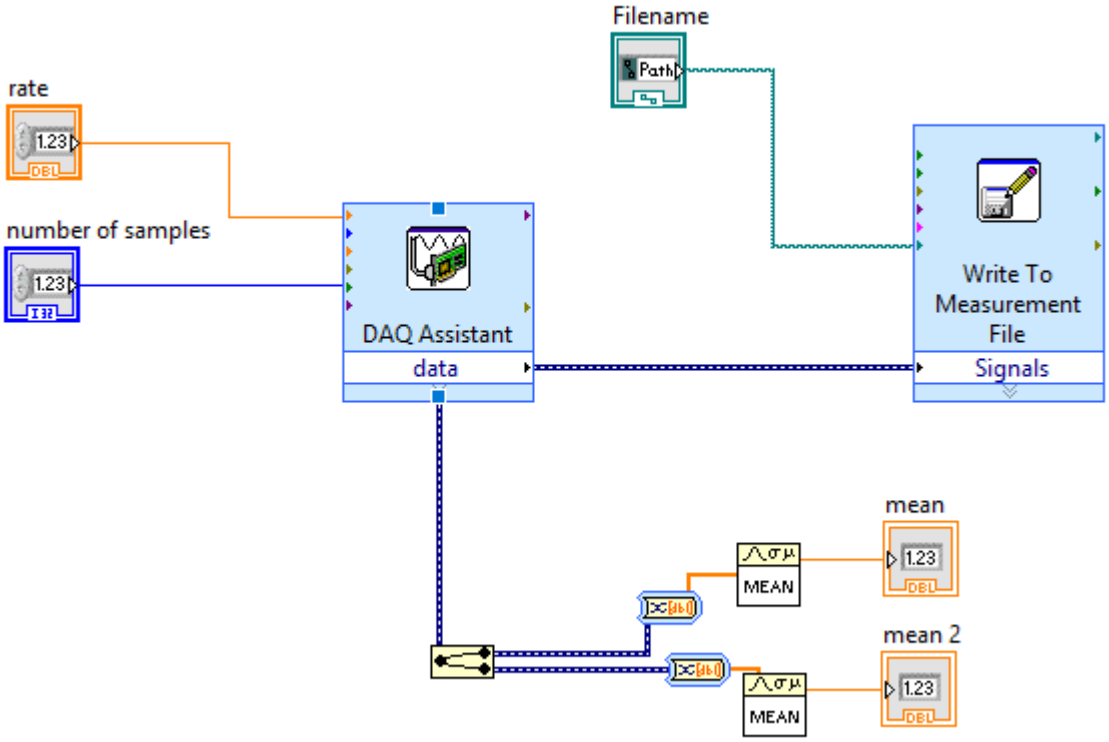
Probe 2 Calibration LabVIEW Code:



Pressure Transducer Calibration LabVIEW Code:



Velocity Measurement Acquisition LabVIEW Code:



6.3 List of Attached Matlab Programs

Main Script File:

- DataAnalysis.m

Included Function Files:

- lvm_import.m
- PressureCal.m
- VelocityCal.m
- VelocityStats.m

Table of Contents

EFD Project Data Analysis

```
%David Tobin
%November 2019

%Code for all included functions is provided below

clear,clc
set(0,'defaultAxesFontSize',18)
```

Manual Calibration Inputs

```
%Last Done on 12/2/19 @ 2:00pm

%Manually Enter P0-Pv0 Data
P0 = [0 0.25 0.52 0.81 1.22 1.49 1.81 1.99]; %kPa
Pv0 = [0 0.2403 0.5097 0.8032 1.2174 1.4994 1.8348 2.0234]; %Volts

%Manually Enter Pv1-Vv1 Data
Pv1 = [0 0.01 0.041 0.12 0.218 0.355 0.595 0.937 1.262 1.610...
    1.796 1.920 2.034]; %Volts
Vv1 = -[0.047 -0.82 -1.074 -1.339 -1.508 -1.657 -1.828 -1.986...
    -2.092 -2.183 -2.226 -2.252 -2.275]; %Volts

%Manually Enter Pv2-Vv2 Data
Pv2 = [0 0.007 0.044 0.115 0.253 0.416 0.567 0.911 1.204 1.390...
    1.656 1.870 2.012]; %Volts
Vv2 = -[0.132 -0.682 -1.004 -1.238 -1.460 -1.613 -1.714 -1.876...
    -1.974 -2.027 -2.092 -2.139 -2.165]; %Volts
```

Flow Parameters

```
rho = 0.9773; %kg/m^3
```

Pressure Calibration

```
%Call PressureCal Function
```

```
[m,b] = PressureCal(P0,Pv0);

%Plot Pressure Calibration Results
figure(1)
p11 = plot(P0,Pv0,'o','MarkerSize',8);
set(p11, 'markerfacecolor', get(p11, 'color'));
hold on
P_line = min(P0):.001:max(P0);
E_line = m.*P_line+b;
p12 = plot(P_line,E_line,'LineWidth',1);
xlabel('Calibrated Pressure $P$ (kPa)','interpreter',...
    'latex','Fontsize',20)
ylabel('Transducer Voltage $E$ (v)','interpreter','latex',...
    'Fontsize',20)
title(sprintf('\\bf{Pressure Transducer Calibration}'),...
    'interpreter','latex','Fontsize',24)
leg1 = legend('Calibration Data',sprintf('$E = %0.2fP %+0.2f$',m,b));
set(leg1,'Interpreter','latex');
set(leg1,'FontSize',20);
set(leg1,'location','northwest')
hold off
```

Calibrate Probe 1 Velocity

```
%Call VelocityCal Function
[a1,V1_cal] = VelocityCal(Pv1,Vv1,m,rho);

%Plot Probe 1 Velocity Calibration Results
figure(2)
subplot(2,1,1)
p21 = plot(Vv1,V1_cal,'o','MarkerSize',8);
set(p21, 'markerfacecolor', get(p21, 'color'))
hold on
Vol_line1 = min(Vv1):.001:max(Vv1);
Vel_line1 = polyval(a1,Vol_line1);
plot(Vol_line1,Vel_line1,'LineWidth',1);
xlabel('Probe 1 Voltage $E_1$ (v)','interpreter','latex'...
    ,'Fontsize',20)
ylabel('Velocity $V_1$ (m/s)','interpreter','latex'...
    ,'Fontsize',20)
title(sprintf('\\bf{Velocity Probe 1 Calibration}'),...
    'interpreter','latex','FontSize',24)
leg2 = legend('Calibration Data',sprintf(['$V_1 = %0.2f '...
    ' %+0.2fE_1 %+0.2fE_1^2 %+0.2fE_1^3 %+0.2fE_1^4$'],...
    a1(5),a1(4),a1(3),a1(2),a1(1)));
set(leg2,'Interpreter','latex');
set(leg2,'location','northwest');
set(leg2,'FontSize',20);
hold off
```

Calibrate Probe 2 Velocity

```
%Call VelocityCal Function
```

```
[a2,V2_cal] = VelocityCal(Pv2,Vv2,m,rho);

%Plot Probe 2 Velocity Calibration Results
figure(2)
subplot(2,1,2)
p22 = plot(Vv2,V2_cal,'o','MarkerSize',8);
set(p22, 'markerfacecolor', get(p22, 'color'))
hold on
Vol_line2 = min(Vv2):.001:max(Vv2);
Vel_line2 = polyval(a2,Vol_line2);
plot(Vol_line2,Vel_line2,'LineWidth',1);
xlabel('Probe 2 Voltage $E_2$ (v)','interpreter','latex'...
    ,'Fontsize',20)
ylabel('Velocity $V_2$ (m/s)','interpreter','latex'...
    ,'Fontsize',20)
title(sprintf('\\bf{Velocity Probe 2 Calibration}'),...
    'interpreter','latex','FontSize',24)
leg3 = legend('Calibration Data',sprintf(['$V_2 = %0.2f '...
    ' %+0.2fE_2 %+0.2fE_2^2 %+0.2fE_2^3 %+0.2fE_2^4$'],...
    a2(5),a2(4),a2(3),a2(2),a2(1)));
set(leg3,'Interpreter','latex');
set(leg3,'location','northwest');
set(leg3,'FontSize',20);
hold off
```

Data Analysis Over All Streamwise and Radial Locations

```
%Calculate Ruu's and Approximate ILS at x = 30cm
distances1 = 5:1:51; %mm
N_pts1 = length(distances1);
r1 = zeros(N_pts1,1);
for i = 1:N_pts1
    filename = sprintf(['/Users/davidtobin/Desktop/UW/Masters'...
        ' II/EFD/Project/30cm round 2/%d.lvm'],i);
    r1(i) = VelocityStats(filename,a1,a2);
end
for i = 1:N_pts1
    if (r1(i) <= 0)
        ILS1 = interp1(r1(i-1:i),distances1(i-1:i),0,'linear');
        break
    end
end

%Calculate Ruu's and Approximate ILS at x = 39.5cm
distances2 = 5:1:51; %mm
N_pts2 = length(distances2);
r2 = zeros(N_pts2,1);
for i = 1:N_pts2
    filename = sprintf(['/Users/davidtobin/Desktop/UW/Masters'...
        ' II/EFD/Project/39.5cm round 2/%d.lvm'],i);
    r2(i) = VelocityStats(filename,a1,a2);
```

```
end
for i = 1:N_pts2
    if (r2(i) <= 0)
        ILS2 = interp1(r2(i-1:i),distances2(i-1:i),0,'linear');
        break
    end
end

%Calculate Ruu's and Approximate ILS at x = 49cm
distances3 = 5:1:51; %mm
N_pts3 = length(distances3);
r3 = zeros(N_pts3,1);
for i = 1:N_pts3
    filename = sprintf(['/Users/davidtobin/Desktop/UW/Masters'...
        ' II/EFD/Project/49cm/%d.lvm'],i);
    r3(i) = VelocityStats(filename,a1,a2);
end
for i = 1:N_pts3
    if (r3(i) <= 0)
        ILS3 = interp1(r3(i-1:i),distances3(i-1:i),0,'linear');
        break
    end
end
```

Final Results Plots

```
%Plot Ruu's and Approximate ILS at x = 30cm
figure(3)
subplot(3,1,1)
p31 = plot([0 distances1],[1 r1'],'-o');
set(p31, 'markerfacecolor', get(p31, 'color'))
hold on
plot(ILS1,0,'xr','Markersize',18,'MarkerFaceColor','red',...
    'LineWidth',20)
ylabel(sprintf('$\\frac{R_{uu}}{\\sigma_1 \\sigma_2}$'),...
    'interpreter','latex','Fontsize',30)
line([0,50],[0,0],'color','k')
title(sprintf('\\bf{Streamwise Distance x = 30 (cm)}'),...
    'interpreter','latex','FontSize',24)
xlim([0 50])
yticks(-0.5:.25:1)
ylim([-0.5 1])
text(ILS1-2,-0.2,sprintf('$l \\approx %3.2f$ (mm)',ILS1),...
    'interpreter','latex','Fontsize',18)
hold off

%Plot Ruu's and Approximate ILS at x = 39.5cm
subplot(3,1,2)
p32 = plot([0 distances1],[1 r2'],'-o');
set(p32, 'markerfacecolor', get(p32, 'color'))
hold on
plot(ILS2,0,'xr','Markersize',18,'MarkerFaceColor','red',...
    'LineWidth',20)
```

```
ylabel(sprintf('$\\frac{R_{uu}}{\\sigma_1 \\sigma_2}$'),...
    'interpreter','latex','Fontsize',30)
line([0,50],[0,0],'color','k')
title(sprintf('\\bf{Streamwise Distance x = 39.5 (cm)}'),...
    'interpreter','latex','FontSize',24)
xlim([0 50])
yticks(-0.5:.25:1)
ylim([-0.5 1])
text(ILS2-2,-0.2,sprintf('$l \\approx %3.2f$ (mm)',ILS2),...
    'interpreter','latex','Fontsize',18)
hold off

%Plot Ruu's and Approximate ILS at x = 49cm
subplot(3,1,3)
p33 = plot([0 distances1],[1 r3'],'-o');
set(p33, 'markerfacecolor', get(p33, 'color'))
hold on
plot(ILS3,0,'xr','Markersize',18,'MarkerFaceColor','red',...
    'LineWidth',20)
ylabel(sprintf('$\\frac{R_{uu}}{\\sigma_1 \\sigma_2}$'),...
    'interpreter','latex','Fontsize',30)
line([0,50],[0,0],'color','k')
title(sprintf('\\bf{Streamwise Distance x = 49 (cm)}'),...
    'interpreter','latex','FontSize',24)
xlim([0 50])
yticks(-0.5:.25:1)
ylim([-0.5 1])
text(ILS3-2,-0.2,sprintf('$l \\approx %3.2f$ (mm)',ILS3),...
    'interpreter','latex','Fontsize',18)
xlabel ('Radial Distance $r$ (mm)','interpreter','Latex',...
    'FontSize',22)
hold off

%Plot Approximate ILS's vs Streamwise Distance
figure(4)
p4 = plot([30 39.5 49],[ILS1 ILS2 ILS3],'-o','MarkerSize',10,...
    'LineWidth',2);
set(p4, 'markerfacecolor', get(p4, 'color'));
xlabel(sprintf('Streamwise Distance $x$ (cm)'),...
    'interpreter','latex','Fontsize',22)
ylabel(sprintf('Integral Length Scale $l$ (mm)'),...
    'interpreter','latex','Fontsize',22)
title(sprintf('\\bf{Integral Length Scale Streamwise Profile}'),...
    'interpreter','latex','Fontsize',26)
xlim([25 50])
```

Confirmation of Velocity Profiles

```
%Calculate Uc/Ur at x = 30cm
distances1 = 5:1:51; %mm
N_pts1 = length(distances1);
Ur_Uc1 = zeros(N_pts1,1);
for i = 1:N_pts1
```

```
    filename = sprintf(['/Users/davidtobin/Desktop/UW/Masters'...
        ' II/EFD/Project/30cm round 2/%d.lvm'],i);
    dat = lvm_import(filename,0);
    Ur_Uc1(i) = mean(-dat(:,2))./mean(-dat(:,1));
end


%Calculate Uc/Ur at x = 39.5cm
distances2 = 5:1:51; %mm
N_pts2 = length(distances2);
Ur_Uc2 = zeros(N_pts2,1);
for i = 1:N_pts2
    filename = sprintf(['/Users/davidtobin/Desktop/UW/Masters'...
        ' II/EFD/Project/39.5cm round 2/%d.lvm'],i);
    dat = lvm_import(filename,0);
    Ur_Uc2(i) = mean(-dat(:,2))./mean(-dat(:,1));
end

%Calculate Uc/Ur at x = 49cm
distances3 = 5:1:51; %mm
N_pts3 = length(distances3);
Ur_Uc3 = zeros(N_pts3,1);
for i = 1:N_pts3
    filename = sprintf(['/Users/davidtobin/Desktop/UW/Masters'...
        ' II/EFD/Project/49cm/%d.lvm'],i);
    dat = lvm_import(filename,0);
    Ur_Uc3(i) = mean(-dat(:,2))./mean(-dat(:,1));
end

%Velocity Profile Plot
figure(5)
p51 = plot(distances1,Ur_Uc1,'-o');
set(p51, 'markerfacecolor', get(p51, 'color'))
hold on
p52 = plot(distances2,Ur_Uc2,'-o');
set(p52, 'markerfacecolor', get(p52, 'color'))
hold on
p53 = plot(distances3,Ur_Uc3,'-o');
set(p53, 'markerfacecolor', get(p53, 'color'))
hold on
ylabel(sprintf('$\\frac{U_r}{U_c}$'),'interpreter','latex',...
    'Fontsize',30)
xlabel ('Radial Distance $r$ (mm)','interpreter','Latex',...
    'FontSize',22)
title(sprintf('\\bf{Radial Profiles of Streamwise Velocity}'),...
    'interpreter','latex','FontSize',24)
xlim([5 50])
yticks(0:.25:1)
ylim([-0 1])
leg5 = legend('x = 30 cm','x = 39.5 cm','x = 49 cm');
set(leg5,'Interpreter','latex');
set(leg5,'location','northeast');
set(leg5,'FontSize',20);
hold off
```

Manual Data / Notes Archive

```
%{

11/20-------------------------------------------------------------------

Pressure:
P0 = [0 0.21 0.40 0.60 0.80 1.00 1.20 1.40 1.63 1.80 2.00]; %kPa
Pv0 = [-0.0016 0.2031 0.3965 0.5882 0.7927 1.0030 1.2016 1.4057 ...
    1.6538 1.8269 2.0400];

Probe 2:
Pv1 = [-0.0036 0.0195 0.0437 0.0625 0.1601 0.3100 0.5272 0.9308...
    1.2143 1.8851]; %Volts
Vv1 = [0.0556 -0.9515 -1.1166 -1.2005 -1.4523 -1.6502 -1.8221 ...
    -2.0159 -2.1118 -2.1878]; %Volts


11/21-------------------------------------------------------------------

Pressure:
P0 = [0.00 0.30 0.83 0.90 1.21 1.51 2.01 2.91]; %kPa
Pv0 = [-0.1492 0.0432 0.4007 0.4542 0.6633 0.8627 1.2090 ...
    1.8445]; %Volts

Probe 1:
Pv1 = [514 504 353 351 387 322 306 794 1296 1663 2270 2787...
    3185 3385]*1E-3; %Volts
Vv1 = [4120 2283 1792 1158 889 533 006 -436 -709 -853 -1058...
    -1191 -1285 -1328]*1E-3; %Volts

!Realized multiplexing isssue that results in non-zero Pv/V at zero V!

Plan: Make sure no DC offset. Also, don't subtract y-intercept during
pressure conversion. Trying again...

Probe 1: (0.0 DC offset)
Pv1 = [-36 -509 -675 -672 -689 -590 -604 -569 -336 10 414 ...
    748 1101]*1E-3; %Volts
Vv1 = [85 -542 -964 -1182 -1328 -1465 -1578 -1661 -1877 -2073 ...
    -2216 -2306 -2378]*1E-3; %Volts

Probe 2: (-0.88 DC offset)
Pv2 = [-496 -592 -602 -576 -431 -276 70 471 1047 1281 1429 ...
    1339]*1E-3; %Volts
Vv2 = [-842 -1662 -2079 -2292 -2556 -2707 -2897 -3063 -3225...
    -3254 -3267 -3280]*1E-3; %Volts

!Negative pressure voltages? Multiplexing with negative HWA signal?

Doing pressure calibration again...

P0 = [0.00 0.10 0.30 0.51 0.70 0.83 0.96]; %kPa
```

```
Pv0 = [0.0115 0.2036 0.5983 1.0025 1.3802 1.6472 1.9185]; %Volts

Pv1 = [-39 -56 -567 -533 -473 -390]*1E-3; %Volts
Vv1 = [75 -872 -992 -1163 -1296 -1403]*1E-3; %Volts

Still negative Pv...

Found it... SWITCH ON NI BOARD MUST BE BE ON GS!!!!!

New test:

Pressure:
P0 = [0.00 0.28 0.41 0.61 1.05 1.36 1.57 1.73 1.97]; %kPa
Pv0 = [-0.0017 0.2686 0.4012 0.6063 1.0562 1.3816 1.6012 ...
    1.7629 2.0089]; %Volts

Probe 1: (0.0 DC offset, Gain = 1)
Pv1 = [0.002 0.0 0.004 0.02 0.074 0.169 0.262 0.513 0.945 ...
    1.362 1.639 1.996]; %Volts
Vv1 = [0.1 -0.458 -0.624 -0.869 -1.168 -1.390 -1.519 -1.732...
    -1.940 -2.069 -2.137 -2.211]; %Volts

Probe 2: (0.0 DC Offset, Gain = 1)
%Manually Enter Pv2-Vv2 Data
Pv2 = [0.001 0.002 0.016 0.056 0.144 0.293 0.614 1.006 1.431 ...
    1.661 2.019]; %Volts
Vv2 = [0.036 -0.557 -0.858 -1.121 -1.365 -1.572 -1.805 -1.973...
    -2.099 -2.153 -2.225]; %Volts

Test Analysis:

% time = 0:0.0001:0.9999;
% figure(2)
% subplot(2,1,1)
% plot(time,V1_test);
% ylabel('Velocity (m/s)')
% title('Probe 1')
%
% subplot(2,1,2)
% plot(time,V2_test);
% ylabel('Velocity (m/s)')
% title('Probe 2')
% xlabel('t')

12/2----------------------------------------------------------------

Final Calibration for Final Data:

%Manually Enter P0-Pv0 Data
P0 = [0 0.25 0.52 0.81 1.22 1.49 1.81 1.99]; %kPa
Pv0 = [0 0.2403 0.5097 0.8032 1.2174 1.4994 1.8348 2.0234]; %Volts

%Manually Enter Pv1-Vv1 Data
Pv1 = [0 0.01 0.041 0.12 0.218 0.355 0.595 0.937 1.262 1.610 1.796...
```

```
    1.920 2.034]; %Volts
Vv1 = -[0.047 -0.82 -1.074 -1.339 -1.508 -1.657 -1.828 -1.986 ...
    -2.092 -2.183 -2.226 -2.252 -2.275]; %Volts

%Manually Enter Pv2-Vv2 Data
Pv2 = [0 0.007 0.044 0.115 0.253 0.416 0.567 0.911 1.204 1.390 ...
    1.656 1.870 2.012]; %Volts
Vv2 = -[0.132 -0.682 -1.004 -1.238 -1.460 -1.613 -1.714 -1.876...
    -1.974 -2.027 -2.092 -2.139 -2.165]; %Volts

distances = [0.3175 1.0175 1.5175 2.0175 2.5175 3.5175 ...
    4.5175 4.9175];%cm;

%}
```

Published with MATLAB® R2017b

```
%lvm_import
%M.A. Hopcraft
%October 2017
%Mathworks File Exchange

%Edited by David Tobin
%November 2019

function data = lvm_import(filename,verbose)
%LVM_IMPORT Imports data from a LabView LVM file
% DATA = LVM_IMPORT(FILENAME,VERBOSE) returns the data from a LVM
 (.lvm)
%  ASCII text file created by LabView.
%
% FILENAME    The name of the .lvm file, with or without ".lvm"
 extension
%
% VERBOSE     How many messages to display. Default is 1 (few
 messages),
%              0 = silent, 2 = display file information and all
 messages
%
% DATA        The data found in the LVM file. DATA is a structure with
%              fields corresponding to the Segments in the file (see
 below)
%              and LVM file header information.
%
%
% This function imports data from a text-formatted LabView Measurement
 File
%  (LVM, extension ".lvm") into MATLAB. A LVM file can have multiple
%  Segments, so that multiple measurements can be combined in a single
%  file. The output variable DATA is a structure with fields named
%  'Segment1', 'Segment2', etc. Each Segment field is a structure with
%  details about the data in the Segment and the actual data in the
 field
%  named 'data'. The column labels and units are stored as cell arrays
 that
%  correspond to the columns in the array of data.
% The size of the data array depends on the type of x-axis data that
 is
%  stored in the LVM file and the number of channels (num_channels).
%  There are three cases:
%  1) No x-data is included in the file ('No')
%   The data array will have num_channels columns (one column per
 channel
%   of data).
%  2) One column of x-data is included in the file ('One')
%   The first column of the data array will be the x-values, and the
 data
%   array will have num_channels+1 columns.
%  3) Each channel has its own x-data ('Multi')
```

```
%   Each channel has two columns, one for x-values, and one for data.
 The
%   data array will have num_channels*2 columns, with the x-values and
%   corresponding data in alternating columns. For example, in a
 Segment
%   with 4 channels, columns 1,3,5,7 will be the x-values for the data
 in
%   columns 2,4,6,8.
%
% Note: because MATLAB only works with a "." decimal separator,
 importing
%  large LVM files that use a "," (or other character) will be
 noticeably
%  slower. Use a "." decimal separator to avoid this issue.
%
% The LVM file specification is available at:
%   http://zone.ni.com/devzone/cda/tut/p/id/4139
%
%
% Example:
%
%  Use the following command to read in the data from a file
 containing two
%   Segments:
%
% >> d=lvm_import('testfile.lvm');
%
% Importing testfile.lvm:
%
% Import complete. 2 Segments found.
%
% >> d
% d =
%       X_Columns: 'One'
%            user: 'hopcroft'
%     Description: 'Pressure, Flowrate, Heat, Power, Analog Voltage,
 Pump on, Temp'
%            date: '2008/03/26'
%            time: '12:18:02.156616'
%           clock: [2008 3 26 12 18 2.156616]
%        Segment1: [1x1 struct]
%        Segment2: [1x1 struct]
%
% >> d.Segment1
% ans =
%            Notes: 'Some notes regarding this data set'
%     num_channels: 8
%          y_units: {8x1 cell}
%          x_units: {8x1 cell}
%               X0: [8x1 double]
%          Delta_X: [8x1 double]
%    column_labels: {9x1 cell}
%             data: [211x9 double]
%          Comment: 'This data rulz'
```

```
%
% >> d.Segment1.column_labels{2}
% ans =
% Thermocouple1
%
% >> plot(d.Segment1.data(:,1),d.Segment1.data(:,2));
% >> xlabel(d.Segment1.column_labels{1});
% >> ylabel(d.Segment1.column_labels{2});
%
%
%
% M.A. Hopcroft
%      < mhopeng at gmail.com >
%

% MH Sep2017
% v3.12 fix bug for importing data-only files
%  (thanks to Enrique Alvarez for bug reporting)
% MH Mar2017
% v3.1  use cellfun to vectorize processing of comma-delimited data
%       (thanks to Victor for suggestion)
% v3.0  use correct test for 'tab'
% MH Aug2016
% v3.0  (BETA) fixes for files that use comma as delimiter
%       improved robustness for files with missing columns
% MH Sep2013
% v2.2  fixes for case of comma separator in multi-segment files
%       use cell2mat for performance improvement
%       (thanks to <die-kenny@t-online.de> for bug report and testing)
% MH May2012
% v2.1  handle "no separator" bug
%       (thanks to <adnan.cheema@gmail.com> for bug report and
 testing)
%       code & comments cleanup
%       remove extraneous column labels (X_Value for "No X" files;
 Comment)
%       clean up verbose output
%       change some field names to NI names
 ("Delta_X","X_Columns","Date")
% MH Mar2012
% v2.0  fix "string bug" related to comma-separated decimals
%       handle multiple Special Headers correctly
%       fix help comments
%       increment version number to match LabView LVM writer
% MH Sep2011
% v1.3  handles LVM Writer version 2.0 (files with decimal separator)
%       Note: if you want to work with older files with a non-"."
 decimal
%       separator character, change the value of
 "data.Decimal_Separator"
% MH Sep2010
% v1.2  bugfixes for "Special" header in LVM files.
%        (Thanks to <bobbyjoe23928@gmail.com> for suggestions)
% MH Apr2010
```

```
% v1.1  use case-insensitive comparisons to maintain compatibility
 with
%        NI LVM Writer version 1.00
%
% MH MAY2009
% v1.02 Add filename input
% MH SEP2008
% v1.01 Fix comments, add Cells
% v1.00 Handle all three possibilities for X-columns (No,One,Multi)
%       Handle LVM files with no header
% MH AUG2008
% v0.92 extracts Comment for each Segment
% MH APR2008
% v0.9  initial version
%

%#ok<*ASGLU>

% message level
if nargin < 2, verbose = 1; end % use 1 for release and 2 for BETA
if verbose >= 1, fprintf(1,'\nlvm_import v3.1\n'); end

% ask for filename if not provided already
if nargin < 1
    filename=input(' Enter the name of the .lvm file: ','s');
    fprintf(1,'\n');
end


% Open the data file
% open and verify the file
fid=fopen(filename);
if fid ~= -1 % then file exists
    fclose(fid);
else
    filename=strcat(filename,'.lvm');
    fid=fopen(filename);
    if fid ~= -1 % then file exists
        fclose(fid);
    else
        error(['File not found in current directory! (' pwd ')']);
    end
end

% open the validated file
fid=fopen(filename);

if verbose >= 1, fprintf(1,' Importing "%s"\n\n',filename); end

% is it really a LVM file?
linein=fgetl(fid);
if verbose >= 2, fprintf(1,'%s\n',linein); end
% Some LabView routines create an LVM file with no header; just a text
 file
```

```
%  with columns of numbers. We can try to import this kind of data.
if ~contains(linein,'LabVIEW')
    try
        data.Segment1.data = dlmread(filename);
        if verbose >= 1, fprintf(1,'This file appears to be an LVM
 file with no header.\n'); end
        if verbose >= 1, fprintf(1,'Data was copied, but no other
 information is available.\n'); end
        return
    catch fileEx
        error('This does not appear to be a text-format LVM file (no
 recognizeable header or data).');
    end
end


% Process file header
% The file header contains several fields with useful information

% default values
data.Decimal_Separator = '.';
text_delimiter={',',' ','\t'};
data.X_Columns='One';

% File header contains date, time, etc.
% Also the file delimiter and decimal separator (LVM v2.0)
if verbose >= 2, fprintf(1,' File Header Contents:\n\n'); end
while 1

    % get a line from the file
    linein=fgetl(fid);
    % handle spurious carriage returns
    if isempty(linein), linein=fgetl(fid); end
    if verbose >= 3, fprintf(1,'%s\n',linein); end
    % what is the tag for this line?
    t_in = textscan(linein,'%s','Delimiter',text_delimiter);
    if isempty(t_in{1}{1})
        tag='notag';
    else
        tag = t_in{1}{1};
    end
    % exit when we reach the end of the header
    if contains(tag,'***End_of_Header***')
        if verbose >= 2, fprintf(1,'\n'); end
        break
    end

    % get the value corresponding to the tag
%     if ~strcmp(tag,'notag')
%         v_in = textscan(linein,'%*s
 %s','delimiter','\t','whitespace','','MultipleDelimsAsOne', 1);
        if size(t_in{1},1)>1 % only process a tag if it has a value
%             val = v_in{1}{1};
            val = t_in{1}{2};
```

```
            switch tag
                case 'Date'
                    data.Date = val;
                case 'Time'
                    data.Time = val;
                case 'Operator'
                    data.user = val;
                case 'Description'
                    data.Description = val;
                case 'Project'
                    data.Project = val;
                case 'Separator'
                    % v3 separator sanity check
                    if strcmpi(val,'Tab')
                        text_delimiter='\t';
                        if contains(linein,',')
                            fprintf(1,'ERROR: File header reports
 "Tab" but uses ",". Check the file and correct if necessary.\n');
                            return
                        end
                    elseif strcmpi(val,'Comma') || strcmpi(val,',')
                        text_delimiter=',';
                        if contains(linein,sprintf('\t'))
                            fprintf(1,'ERROR: File header reports
 "Comma" but uses "tab". Check the file and correct if necessary.\n');
                            return
                        end
                    end

                case 'X_Columns'
                    data.X_Columns = val;
                case 'Decimal_Separator'
                    data.Decimal_Separator = val;
            end
            if verbose >= 2, fprintf(1,'%s: %s\n',tag,val); end
        end
%     end

end

% create matlab-formatted date vector
if isfield(data,'time') && isfield(data,'date')
    dt = textscan(data.Date,'%d','Delimiter','/');
    tm = textscan(data.Time,'%d','Delimiter',':');
    if length(tm{1})==3
        data.clock=[dt{1}(1) dt{1}(2) dt{1}(3) tm{1}(1) tm{1}(2) tm{1}
(3)];
    elseif length(tm{1})==2
        data.clock=[dt{1}(1) dt{1}(2) dt{1}(3) tm{1}(1) tm{1}(2) 0];
    else
        data.clock=[dt{1}(1) dt{1}(2) dt{1}(3) 0 0 0];
    end
end
```

```
if verbose >= 3, fprintf(1,' Text delimiter is "%s":\n
\n',text_delimiter); end


% Process segments
% process data segments in a loop until finished
segnum = 1;
val=[]; tag=[]; %#ok<NASGU>
while 1
    %segnum = segnum +1;
    fieldnm = ['Segment' num2str(segnum)];

    % - Segment header
    if verbose >= 1, fprintf(1,' Segment %d:\n\n',segnum); end
    % loop to read segment header
    while 1
        % get a line from the file
        linein=fgetl(fid);
        % handle spurious carriage returns/blank lines/end of file
        while isempty(linein), linein=fgetl(fid); end
        if feof(fid), break; end
        if verbose >= 3, fprintf(1,'%s\n',linein); end

        % Ignore "special segments"
        % "special segments" can hold other types of data. The type
 tag is
        % the first line after the Start tag. As of version 2.0,
        % LabView defines three types:
        %  Binary_Data
        %  Packet_Notes
        %  Wfm_Sclr_Meas
        % In theory, users can define their own types as well.
 LVM_IMPORT
        %  ignores any "special segments" it finds.
        % If special segments are handled in future versions,
 recommend
        %  moving the handler outside the segment read loop.
        if contains(linein,'***Start_Special***')
            special_seg = 1;
            while special_seg

                while 1 % process lines until we find the end of the
 special segment
                    % get a line from the file
                    linein=fgetl(fid);
                    % handle spurious carriage returns
                    if isempty(linein), linein=fgetl(fid); end
                    % test for end of file
                    if linein==-1, break; end
                    if verbose >= 2, fprintf(1,'%s\n',linein); end
                    if contains(linein,'***End_Special***')
                        if verbose >= 2, fprintf(1,'\n'); end
                        break
```

```
                    end
                end

                % get the next line and proceed with file
                %  (there may be additional Special Segments)
                linein=fgetl(fid);
                % handle spurious carriage returns/blank lines/end of
file
                while isempty(linein), linein=fgetl(fid); end
                if feof(fid), break; end
                if ~contains(linein,'***Start_Special***')
                    special_seg = 0;
                    if verbose >= 1, fprintf(1,' [Special Segment
ignored]\n\n'); end
                end
            end
        end % end special segment handler


        % what is the tag for this line?
        t_in = textscan(linein,'%s','Delimiter',text_delimiter);
        if isempty(t_in{1}{1})
            tag='notag';
        else
            tag = t_in{1}{1};
            %disp(t_in{1})
        end
        if verbose >= 3, fprintf(1,'%s\n',linein); end
        % exit when we reach the end of the header
        if contains(tag,'***End_of_Header***')
            if verbose >= 3, fprintf(1,'\n'); end
            break
        end

        % get the value corresponding to the tag
        % v3 assignments use dynamic field names
        if size(t_in{1},1)>1 % only process a tag if it has a value
            switch tag
                case 'Notes'
   %                 %d_in = textscan(linein,'%*s
%s','delimiter','\t','whitespace','');
   %                 d_in = linein;
                    data.(fieldnm).Notes = t_in{1}{2:end};
                case 'Test_Name'
   %                 %d_in = textscan(linein,'%*s
%s','delimiter','\t','whitespace','');
   %                 d_in = linein;
                    data.(fieldnm).Test_Name = t_in{1}{2:end};
 %d_in{1}{1};
                case 'Channels'
   %                 numchan = textscan(linein,sprintf('%%*s%s%
%d',text_delimiter),1)
   %                 data.(fieldnm).num_channels = numchan{1};
                    data.(fieldnm).num_channels = str2num(t_in{1}{2});
```

```
                case 'Samples'
   %                  numsamp =
textscan(linein,'%s','delimiter',text_delimiter);
   %                  numsamp1 = numsamp{1};
                    numsamp1 = t_in{1}(2:end);
   %                  numsamp1(1)=[]; % remove tag "Samples"
                    num_samples=[];
                    for k=1:length(numsamp1)
                        num_samples = [num_samples
sscanf(numsamp1{k},'%f')]; %#ok<AGROW>
                    end
                    %numsamp2=str2num(cell2mat(numsamp1));
                  %#ok<ST2NM>
                    data.(fieldnm).num_samples = num_samples;
                case 'Y_Unit_Label'
   %                  Y_units =
textscan(linein,'%s','delimiter',text_delimiter);
   %                  data.(fieldnm).y_units=Y_units{1}';
                    data.(fieldnm).y_units=t_in{1}';
                    data.(fieldnm).y_units(1)=[]; % remove tag
                case 'Y_Dimension'
   %                  Y_Dim =
textscan(linein,'%s','delimiter',text_delimiter);
   %                  data.(fieldnm).y_type=Y_Dim{1}';
                    data.(fieldnm).y_type=t_in{1}';
                    data.(fieldnm).y_type(1)=[]; % remove tag
                case 'X_Unit_Label'
   %                  X_units =
textscan(linein,'%s','delimiter',text_delimiter);
   %                  data.(fieldnm).x_units=X_units{1}';
                    data.(fieldnm).x_units=t_in{1}';
                    data.(fieldnm).x_units(1)=[];
                case 'X_Dimension'
   %                  X_Dim =
textscan(linein,'%s','delimiter',text_delimiter);
   %                  data.(fieldnm).x_type=X_Dim{1}';
                    data.(fieldnm).x_type=t_in{1}';
                    data.(fieldnm).x_type(1)=[]; % remove tag
                case 'X0'
                    %[Xnought, val]=strtok(linein);
                    val=t_in{1}(2:end);
                    if ~strcmp(data.Decimal_Separator,'.')
                        val = strrep(val,data.Decimal_Separator,'.');
                    end
                    X0=[];
                    for k=1:length(val)
                        X0 = [X0 sscanf(val{k},'%e')]; %#ok<AGROW>
                    end
                    data.(fieldnm).X0 = X0;
                    %data.(fieldnm).X0 = textscan(val,'%e');
                case 'Delta_X' %,
                    %[Delta_X, val]=strtok(linein);
                    val=t_in{1}(2:end);
                    if ~strcmp(data.Decimal_Separator,'.')
```

```
                        val = strrep(val,data.Decimal_Separator,'.');
                    end
                    Delta_X=[];
                    for k=1:length(val)
                        Delta_X = [Delta_X
 sscanf(val{k},'%e')]; %#ok<AGROW>
                    end
                    data.(fieldnm).Delta_X = Delta_X;
                end
            end

        end % end reading segment header loop
        % Done reading segment header

        % after each segment header is the row of column labels
        linein=fgetl(fid);
        Y_labels = textscan(linein,'%s','delimiter',text_delimiter);
        data.(fieldnm).column_labels=Y_labels{1}';
        % The X-column always exists, even if it is empty. Remove if not
 used.
        if strcmpi(data.X_Columns,'No')
            data.(fieldnm).column_labels(1)=[];
        end
        % remove empty entries and "Comment" label
        if any(strcmpi(data.(fieldnm).column_labels,'Comment'))
            data.(fieldnm).column_labels=data.
(fieldnm).column_labels(1:find(strcmpi(data.
(fieldnm).column_labels,'Comment'))-1);
        end
        % display column labels
        if verbose >= 1
            fprintf(1,' %d Data Columns:\n | ',length(data.
(fieldnm).column_labels));
            for i=1:length(data.(fieldnm).column_labels)
                fprintf(1,'%s | ',data.(fieldnm).column_labels{i});
            end
            fprintf(1,'\n\n');
        end



        % - Segment Data
        % Create a format string for textscan depending on the number/type
 of
        %  channels. If there are additional segments, texscan will quit
 when
        %  it comes to a text line which does not fit the format, and the
 loop
        %  will repeat.
        if verbose >= 1, fprintf(1,' Importing data from Segment
 %d...',segnum); end

        % How many data columns do we have? (including X data)
        switch data.X_Columns
```

```
            case 'No'
                % an empty X column exists in the file
                numdatacols = data.(fieldnm).num_channels+1;
                xColPlural='no X-Columns';
            case 'One'
                numdatacols = data.(fieldnm).num_channels+1;
                xColPlural='one X-Column';
            case 'Multi'
                numdatacols = data.(fieldnm).num_channels*2;
                xColPlural='multiple X-Columns';
        end

        % handle case of not using periods (aka "dot" or ".") for decimal
    point separators
        %  (LVM version 2.0+)
        if ~strcmp(data.Decimal_Separator,'.')
            if verbose >= 2, fprintf(1,'\n  (using decimal separator
    "%s")\n',data.Decimal_Separator); end

            % create a format string for reading data as numbers
            fs = '%s'; for i=2:numdatacols, fs = [fs ' %s']; end
         %#ok<AGROW>
            % add one more column for the comment field
            fs = [fs ' %s'];
        %#ok<AGROW>
            % v3.1 - use cellfun to process data
            % Read columns as strings
            rawdata = textscan(fid,fs,'delimiter',text_delimiter);
            % Convert ',' decimal separator to '.' decimal separator
            rawdata = cellfun(@(x) strrep(x,data.Decimal_Separator,'.'),
    rawdata, 'UniformOutput', false);
     % save first row comment as The Comment for this segment
            data.(fieldnm).Comment = rawdata{size(rawdata,2)}{1};
            % Transform strings back to numbers
            rawdata = cellfun(@(x) str2double(x),
    rawdata, 'UniformOutput', false);

        % else is the typical case, with a '.' decimal separator
        else
            % create a format string for reading data as numbers
            fs = '%f'; for i=2:numdatacols, fs = [fs ' %f']; end
              %#ok<AGROW>
            % add one more column for the comment field
            fs = [fs ' %s'];
              %#ok<AGROW>
            % read the data from file
            rawdata = textscan(fid,fs,'delimiter',text_delimiter);
            % save first row comment as The Comment for this segment
            data.(fieldnm).Comment = rawdata{size(rawdata,2)}{1};
        end

        % v2.2 use cell2mat here instead of a loop for better performance
        % consolidate data into a simple array, ignore comments
        data.(fieldnm).data=cell2mat(rawdata(:,1:numdatacols));
```

```
    % If we have a "No X data" file, remove the first column (it is
 empty/NaN)
    if strcmpi(data.X_Columns,'No')
        data.(fieldnm).data=data.(fieldnm).data(:,2:end);
    end

    if verbose >= 1, fprintf(1,' complete (%g data points (rows)).\n
\n',length(data.(fieldnm).data)); end

    % test for end of file
    if feof(fid)
        if verbose >= 2, fprintf(1,' [End of File]\n\n'); end
        break;
    else
        segnum = segnum+1;
    end


end % end process segment


if verbose >= 1
    if segnum > 1, segplural='Segments';
    else segplural='Segment'; end %#ok<SEPEX>
    fprintf(1,'\n Import complete. File has %s and %d Data %s.\n
\n',xColPlural,segnum,segplural);
end

data = data.Segment1.data; %RETURN ONLY DATA, NOT STRUCTURE

% close the file
fclose(fid);
return
```

Published with MATLAB® R2017b

```
%Pressure Calibration Function

%David Tobin
%November 2019

%This function provides the linear parameters (m,b) for the
%relationship between measured pressure (kPa) and voltage from the
%pressure transducer (V) given two sufficently long vectors of these
%quantities obtained manually using the pressure calibrator.

function [m,b] = PressureCal(P,Pv)

coeffs = polyfit(P,Pv,1);
m = coeffs(1);
b = coeffs(2);

end
```

Published with MATLAB® R2017b

```
%Velocity Calibration Function

%David Tobin
%November 2019

%This function provides the 4th order polynomial coefficients for the
%relationship between probe velocity (m/s) and probe voltage (V), as
%well as the corresponding velocity measurements (m/s) that
%correspond to the calibration velocity voltages, given sufficently
%long vectors of the voltage from the pressure transducer (V) and
%the voltage from the probe (V) when the probe is placed at the
%centerline of the exit of the jet at incrimental jet speeds,
%the linear slope parameter of the pressure transducer, and the air
%density (kg/m^3).

function [a,V] = VelocityCal(Pv,Vv,m,rho)

P = (Pv)./m; %kPa   !DON'T INCLUDE Y-INTERCEPT!
V = sqrt(2.*P*1000/rho); %m/s
a = polyfit(Vv,V,4);

end
```

Published with MATLAB® R2017b

```
%Velocity Statistics Function

%David Tobin
%November 2019

%This function provides the cross-correlation for a set of data, given
%the Labview text file data (sampling frequency, Probe 1 voltage
%record, Probe 2 voltage record), and the calibration polynomial
%coefficients for each probe.

function [r] = VelocityStats(filename,a1,a2)

dat = lvm_import(filename,0);
Vv1 = -dat(:,1);
Vv2 = -dat(:,2);

V1 = polyval(a1,Vv1);
V2 = polyval(a2,Vv2);

m1 = mean(V1);
m2 = mean(V2);

fluc1 = V1 - m1;
fluc2 = V2 - m2;

SD1 = sqrt(mean(fluc1.^2));
SD2 = sqrt(mean(fluc2.^2));

r = mean(fluc1.*fluc2)/SD1/SD2;

end
```

Published with MATLAB® R2017b