The Design and Characterization of a 3D Printed Venturi Tube

Joshua E. Hrisko[*]
Principal Engineer, Maker Portal
New York, NY 10025

Abstract

A 3D-printed venturi tube is presented as a rapid method for measuring flow rates of small fans commonly used in electronics cooling. The fluid dynamical theory, design, and testing behind the venturi device are introduced, strictly adhering to the performance test code on flow measurement, PTC 19.5-2004, prescribed by the American Society of Mechanical Engineers (ASME). The diameter of the venturi inlet was designed to fit a small 80mm fan blowing axially down the tube. The throat of the tube was designed to be 57mm, resulting in a diameter ratio of 0.75. The Arduino platform serves as the control and acquisition points for the ensuing analysis, where the fan speed was changed based on a pulse-width modulated duty cycle. Barometric pressure, differential pressure, and ambient temperature were all acquired using two external sensors. A computational fluid dynamics (CFD) simulation was used to compare flow rates across similar pressure differentials of the venturi tube. The incompressible venturi equation agreed with the CFD model to within 0.7% across the range of Reynolds numbers and pressure differentials recorded during the experiments. The error established between the CFD model and the venturi equation were far below the inherent error of the measurement devices, indicating that the venturi equation suffices for the current tube geometry and range of pressures. The maximum volumetric flow rate measured using the 3D printed venturi tube was found to be within 4% of the manufacturer's cited flow rate, indicating that the 3D printed venturi tube is an accurate instrument for determining flow rate.

Keywords Venturi · Fluid · 3D-Printing · Flow Measurement · CFD · Arduino · Differential Pressure

1 Introduction

The ASME performance test code standard on flow measurement, PTC 19.5-2004, outlines specifications for measuring fluid flow with a variety of differential pressure meters [1]. One particular measurement device, the ASME venturi tube, has been widely tested and calibrated under strict laboratory conditions, where errors can be less than 0.5% under optimal design conditions [2]. The ASME venturi design specification is applied to the current application using fused deposition modeling—a common form of 3D printing. Commonly, steel or other hardened metals are used in the design of venturi tubes; however, for the low velocity conditions of our experiments, pressures are expected to be far below the stress limits observed for 3D printed materials [3, 4].

The venturi tube introduced here is designed to fit within the limits of many common desktop 3D printers that often have vertical printing capabilities limited to 300mm or less (see popular printers by Creality, Ultimaker, Monoprice, Makerbot, Prusa). Another design constraint is the size of the fan being used at the inlet. The fan diameter will dictate the size of both the inlet and the tapered section of the venturi tube. Luckily, the perfect combination of fan size and geometry for the venturi tube were found for the 3D printing domain and are introduced in Section 3.1.

The complete background of compressible and incompressible flow through a venturi tube is given in the theoretical background in Section 2. The fluid dynamics theory is introduced as it relates to ideal gas flow through a venturi tube. Then, the instrumentation used in the experimental process is introduced, which encompasses the venturi design as well as the instruments used to test the tube. An Arduino board is used to both control the fan rotation speed and record data measured by a pair of sensors. The first sensor measures barometric pressure and ambient temperature, while the

[*]`engineer@makersportal.com | https://makersportal.com`

second measures the pressure differential across the venturi tube. The wiring of the sensors, fan, and Arduino board is also introduced.

Lastly, a computational fluid dynamics (CFD) simulation is introduced as a comparison point against the incompressible flow equations developed in the theoretical section of this paper and the ASME performance test code. The CFD model will demonstrate the accuracy of the ASME venturi tube approximations and establish the error associated with the 3D printed venturi tube. Moreover, the fan being used has an associated manufacturer's maximum flow rate, which will be compared against our venturi tube approximations. The CFD model, empirical and theoretical venturi tube equations, and the comparison with the manufacturer's flow rate will aid in the characterization of the 3D printed device, ultimately indicating whether a 3D printed tube is a valid way of measuring flow rates of fans.

2 Theoretical Background

2.1 Fluid Dynamics

Venturi tubes take advantage of flow continuity and energy conservation along orifices, where changes in area affect the pressure and mass of enclosed flow. The flow within a venturi tube can be modeled first by employing the mass conservation equation:

$$\frac{\partial \rho}{\partial t} + \frac{\partial(\rho u_i)}{\partial x_i} = 0 \tag{1}$$

where ρ is the fluid density and u_i is a velocity component. Similarly, the conservation equation for energy can be written:

$$\frac{\partial}{\partial t}\left(\rho E + \frac{\rho u_j^2}{2} + \rho g z\right) dV + \frac{\partial}{\partial x_i}\left[\left(E + \frac{u_i^2}{2} + gz\right)(\rho u_i)\right] dV = -\frac{\partial(p u_i)}{\partial x_i} dV + \delta\dot{W} + \delta\dot{Q} \tag{2}$$

where E is the internal energy, z if the height relating to the potential energy equating to zero, dV is the differential volume element, $\delta\dot{Q}$ represents the heat transfer rate, and $\delta\dot{W}$ is the rate of work done across the meter. Under assumptions of no heat transfer, no work being done across the system, and no internal energy change—the energy equation can be simplified [5]:

$$\frac{\partial}{\partial t}\left(\frac{\rho u_j^2}{2} + \rho g z\right) + \frac{\partial}{\partial x_i}\left[\left(\frac{u_i^2}{2} + gz\right)(\rho u_i)\right] = -\frac{\partial(p u_i)}{\partial x_i} \tag{3}$$

In addition to the assumptions above, steady-state conditions can be applied, resulting in the following simplifications of the continuity and energy equations, respectively:

$$\frac{\partial(\rho u_i)}{\partial x_i} = 0 \tag{4}$$

$$\frac{\partial}{\partial x_i}\left[\left(\frac{u_j^2}{2} + gz\right)(\rho u_i)\right] = -\frac{\partial(p u_i)}{\partial x_i} \tag{5}$$

collecting terms and taking advantage of the product rule and continuity:

$$\frac{\partial}{\partial x_i}\left[\left(\frac{u_j^2}{2} + gz\right)(\rho u_i) + (p u_i)\right] = 0 \tag{6}$$

$$(\rho u_i)\frac{\partial}{\partial x_i}\left[\left(\frac{u_j^2}{2} + gz\right) + \frac{p}{\rho}\right] = 0 \tag{7}$$

This leads to the final form of the thermodynamic energy per unit mass equation without heat transfer, work being done on the system, or internal energy change:

$$\frac{\partial u_j^2}{2} + g dz + \frac{dp}{\rho} + p d\left(\frac{1}{\rho}\right) = 0 \tag{8}$$

The last term on the right-hand side of Eqn. 8 is often neglected under the isentropic assumption—where the density changes must be a result of changes in pressure alone, since there is no heat transfer, external work being done on the system, or mass being added to the system.

2.2 Thermodynamics and Ideal Gases

In the case of a DC fan blowing air, the equation of state can be written as:

$$p = \rho R_s T \tag{9}$$

where p is pressure, ρ is the gas density, R_s is the specific gas constant, and T is the temperature. The gas flowing through the venturi tube is assumed isentropic, where the process is reversible and adiabatic ($\delta Q = 0$). In terms of the first law of thermodynamics, this can be written as:

$$dU = \delta Q - \delta W = -\delta W \tag{10}$$

The work can be written as a function of the pressure and volume of the system, in the absence of pressure changes without changes in volume [6]:

$$dU = -\delta(pV) = -pdV - Vdp = -pdV \tag{11}$$

Additionally, enthalpy is often used to describe the heat content of the system:

$$h = U + pV \tag{12}$$

where h represents the enthalpy in the system. Differentiating enthalpy leads to:

$$dh = dU + pdV + Vdp \tag{13}$$

substituting Eqn. 10 into Eqn. 13:

$$dh = -pdV + pdV + Vdp = Vdp \tag{14}$$

Stating the expressions for specific heat capacity at constant pressure and volume [7]:

$$c_v = \frac{dU}{dT} \tag{15}$$

$$c_p = \frac{dh}{dT} \tag{16}$$

The relationships can be substituted back into the equations for isentropic gases (Eqn. 14 and Eqn. 11):

$$c_v dT = -pdV \tag{17}$$

$$c_p dT = Vdp \tag{18}$$

Employing the equation of state:

$$c_v dT = -\frac{mR_sT}{V}dV \tag{19}$$

$$c_p dT = \frac{mR_sT}{p}dp \tag{20}$$

Integrating both equations over a fixed range leads to:

$$c_v \ln\left(\frac{T_2}{T_1}\right) = mR_s \ln\left(\frac{V_1}{V_2}\right) \tag{21}$$

$$c_p \ln\left(\frac{T_2}{T_1}\right) = mR_s \ln\left(\frac{p_2}{p_1}\right) \tag{22}$$

Dividing the specific heat at constant pressure by the specific heat at constant volume leads to the following:

$$\frac{c_p}{c_v} = \frac{\ln\left(\frac{p_2}{p_1}\right)}{\ln\left(\frac{V_1}{V_2}\right)} \tag{23}$$

In this context, the inverse ratio of densities can be substituted for the ratio of volumes. Moreover, denoting the ratio of specific heats as $\gamma = c_p/c_v$ gives the following isentropic relationship between pressure and density for an ideal gas:

$$\frac{p_2}{p_1} = \left(\frac{\rho_2}{\rho_1}\right)^{\gamma} \tag{24}$$

The ratio of the specific heat capacities in differential form is also useful:

$$\frac{c_p}{c_v} = \frac{\frac{mR_sTdp}{p}}{\frac{mR_sTdV}{V}} \tag{25}$$

By substituting density and mass for volume:

$$\frac{c_p}{c_v} = -\frac{V}{p}\frac{dp}{dV} = \frac{1}{\rho p}\frac{dp}{d\left(\frac{1}{\rho}\right)} \tag{26}$$

substituting the inverse of density as ν, for simplifying the notation:

$$\frac{c_p}{c_v} = \frac{\nu}{p}\frac{dp}{d\nu} \tag{27}$$

Integrating:

$$\int c_p \frac{d\nu}{\nu} = \int c_v \frac{dp}{p} \tag{28}$$

result in the following, where the ratio of specific heats is substituted as γ:

$$\gamma \ln \nu = \ln p + K \tag{29}$$

The constant, K, is a result of the integration. The density can be substituted back in for ν and the above can be simplified:

$$\ln p - \gamma \ln \rho = K \tag{30}$$

and in its final form, a simple relationship between pressure and density emerges:

$$p = K\rho^{\gamma} \tag{31}$$

differentiating this equation also becomes useful later:

$$\frac{dp}{d\rho} = K\gamma\rho^{\gamma-1} \tag{32}$$

2.3 Compressible Flow for Ideal Gas

Returning to the conservation of energy in Eqn. 8 and implementing the isentropic relationship between density and pressure (Eqn. 31), the conservation of energy can be rewritten:

$$\frac{\partial u_j^2}{2} + gdz + K\gamma\rho^{\gamma-2}d\rho + pd\left(\frac{1}{\rho}\right) = 0 \tag{33}$$

As stated in Section 2.1, the last term in the equation above can be assumed zero for the isentropic process, due to the lack of external heat or work being done on the system, and no added mass. Thus, integrating Eqn. 33 under the conditions of an ideal gas and one-dimensional flow results in the following:

$$\frac{U_2^2 - U_1^2}{2} + g(z_2 - z_1) + K\left(\frac{\gamma}{\gamma - 1}\right)(\rho_2^{\gamma-1} - \rho_1^{\gamma-1}) = 0 \tag{34}$$

substituting the relationship between pressure and density from Eqn. 31:

$$\frac{U_2^2 - U_1^2}{2} + g(z_2 - z_1) + \left(\frac{\gamma}{\gamma - 1}\right)\left(\frac{p_2}{\rho_2} - \frac{p_1}{\rho_1}\right) = 0 \tag{35}$$

Now, a relationship has been established between the component velocity in one direction, U, gravity, pressure, density, and the gas being used (γ). Equation 35 is used for applications in compressible flows under varying velocities, potential heights, pressures, and densities. Returning to the continuity equation allows some simplification of Eqn. 35. Integrating over a control volume allows Eqn. 4 to be written as a mass flow rate:

$$\int \frac{\partial(\rho u_i)}{\partial x_i} dV = 0 \tag{36}$$

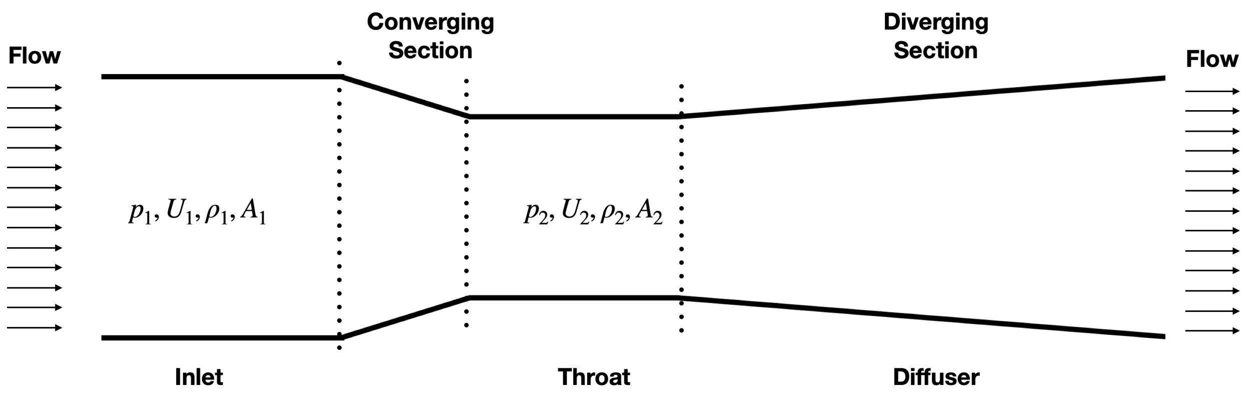


Figure 1: Differential pressure meter diagram showing the inlet and throat as the test points for the variables used for quantifying velocities and flow rates.

If one-dimensional flow is prescribed, the mass flow rate can be simplified:

$$\int d(\rho U A) = \rho_2 U_2 A_2 - \rho_1 U_1 A_1 = 0 \tag{37}$$

If the first velocity is solved for:

$$U_1 = \frac{\rho_2}{\rho_1}\frac{A_2}{A_1}U_2 \tag{38}$$

Inserting Eqn. 38 into Eqn. 35:

$$\frac{U_2^2 - \left(\frac{\rho_2}{\rho_1}\frac{A_2}{A_1}U_2\right)^2}{2} + g(z_2 - z_1) + \left(\frac{\gamma}{\gamma - 1}\right)\left(\frac{p_2}{\rho_2} - \frac{p_1}{\rho_1}\right) = 0 \tag{39}$$

By solving for the resultant velocity, the theoretical compressible flow equation through varying cross-sectional areas is given:

$$U_2 = \left[\left(\frac{1}{1 - \left(\frac{\rho_2 A_2}{\rho_1 A_1}\right)^2}\right)\left(\left(\frac{2\gamma}{\gamma - 1}\right)\left(\frac{p_1}{\rho_1} - \frac{p_2}{\rho_2}\right) - 2g(z_2 - z_1)\right)\right]^{0.5} \tag{40}$$

Some assumptions need to be made regarding the application in order to make further simplifications. For example, in the next section, flow through a differential pressure meter will be explored where simplifications will be made that fit those particular conditions.

2.4 Differential Pressure Meter

Figure 1 is a simplified drawing of the typical geometry used for a differential flow meter. The terms used in Fig. 1 are used in the subsequent derivation of venturi meter relationships and components of their design. First, we look at the incompressible fluid scenario in a typical venturi tube, where Eqn. 8 becomes the simplified Bernoulli equation:

$$\frac{\partial u_j^2}{2} + g dz + \frac{dp}{\rho} = 0 \tag{41}$$

which can be integrated directly, under constant density and 1D flow, to result in the following:

$$\frac{U_2^2}{2} + g z_2 + \frac{p_2}{\rho} = \frac{U_1^2}{2} + g z_1 + \frac{p_1}{\rho} \tag{42}$$

Using the result in Eqn. 38, assuming constant density, and inserting back into the integrated Bernoulli equation gives the following:

$$\frac{U_2^2}{2} + g z_2 + \frac{p_2}{\rho} = \frac{1}{2}\left(\frac{A_2}{A_1}U_2\right)^2 + g z_1 + \frac{p_1}{\rho} \tag{43}$$

Solving for velocity:

$$U_2 = \left[\frac{2(p_1 - p_2) + 2\rho g(z_1 - z_2)}{\rho\,(1-\beta^4)}\right]^{0.5} \tag{44}$$

where β^2 has been substituted for the ratio of areas. Additionally, the effects due to gravity are often calibrated out of the pressure measurement due to the linear relationship between the pressure and gravitational component—assuming the height between pressure measurement points is unchanging—which leads to the very simple and classic incompressible velocity approximation for a differential pressure scenario:

$$U_2 = \left(\frac{2\Delta p}{\rho(1-\beta^4)}\right)^{0.5} \tag{45}$$

which collapses the pressure difference $(p_1 - p_2)$ into the Δp term. This classic form will be used to shape the more complex compressible scenario discussed as follows. For a differential pressure meter with an ideal gas flowing, gravity is neglected [5], resulting in the simplified form of Eqn. 40:

$$U_2 = \left[\left(\frac{1}{1-\left(\frac{\rho_2 A_2}{\rho_1 A_1}\right)^2}\right)\left(\frac{2\gamma}{\gamma-1}\right)\left(\frac{p_1}{\rho_1} - \frac{p_2}{\rho_2}\right)\right]^{0.5} \tag{46}$$

Before the velocity can be approximated for a given scenario, the second density must be substituted, as it is difficult to quantify in an experimental setting. Using the isentropic relationship between pressure and density (Eqn. 24), the second density term can be solved for:

$$\rho_2 = \rho_1\left(\frac{p_2}{p_1}\right)^{1/\gamma} \tag{47}$$

Substituted Eqn. 47 into Eqn. 40 results in the following:

$$U_2 = \left[\left(\frac{1}{1-\left(\left(\frac{p_2}{p_1}\right)^{1/\gamma}\left(\frac{A_2}{A_1}\right)\right)^2}\right)\left(\frac{2\gamma}{\gamma-1}\right)\left(\frac{p_1}{\rho_1} - \frac{p_2}{\left(\rho_1\left(\frac{p_2}{p_1}\right)^{1/\gamma}\right)}\right)\right]^{0.5} \tag{48}$$

making the following substitutions for simplification:

$$r = \frac{p_2}{p_1}$$

$$\beta^2 = \frac{A_2}{A_1}$$

$$\xi = \frac{\gamma}{\gamma-1}$$

substituting back into Eqn. 48:

$$U_2 = \left[\left(\frac{1}{1-\left(r^{2/\gamma}\beta^4\right)}\right)\frac{2\xi}{\rho_1}\left(p_1 - \left(\frac{p_2}{r^{1/\gamma}}\right)\right)\right]^{0.5} \tag{49}$$

Recalling the form of Eqn. 45 for the incompressible case, the ideal gas compressible case can be forced into a similar form:

$$U_2 = \left[\frac{2\Delta p}{(1-\beta^4)}\left(\frac{(1-\beta^4)}{1-\left(r^{2/\gamma}\beta^4\right)}\right)\frac{\xi}{\rho_1}\left(\frac{1}{\Delta p}\left(p_1 - \left(\frac{p_2}{r^{1/\gamma}}\right)\right)\right)\right]^{0.5} \tag{50}$$

separating the incompressible portion from the rest:

$$U_2 = \left(\frac{2\Delta p}{\rho_1(1-\beta^4)}\right)^{0.5}\cdot\left[\left(\frac{(1-\beta^4)}{1-\left(r^{2/\gamma}\beta^4\right)}\right)\xi\left(\frac{1}{\Delta p}\left(p_1 - \left(\frac{p_2}{r^{1/\gamma}}\right)\right)\right)\right]^{0.5} \tag{51}$$

One final simplification can be implemented by looking at the pressure variables in the last term:

$$\frac{p_1 - \left(\frac{p_2}{r^{1/\gamma}}\right)}{\Delta p} = \frac{p_1 - \left(\frac{p_2}{r^{1/\gamma}}\right)}{(p_1 - p_2)} = \frac{1 - r^{(1-(1/\gamma))}}{1-r} = \frac{1 - r^{(\gamma-1)/\gamma}}{1-r} \tag{52}$$

This gives the final compressible velocity equation for an ideal gas flowing through a differential pressure meter:

$$U_2 = \left(\frac{2\Delta p}{\rho_1(1-\beta^4)}\right)^{0.5} \cdot \left[\frac{\gamma}{\gamma - 1}\left(\frac{(1-\beta^4)}{1 - \left(r^{2/\gamma}\beta^4\right)}\right)\left(\frac{1 - r^{(\gamma-1)/\gamma}}{1-r}\right)\right]^{0.5} \tag{53}$$

Equation 53 can be used for both compressible and incompressible scenarios, where the right-hand square root is approximated as 1.0 for incompressible scenarios. In the next section, the design for the 3D printed venturi tube will be introduced, followed by a discussion of the sensors used to acquire the parameters referenced in Eqn. 53.

3 Instrumentation

3.1 Venturi tube Design

The venturi tube presented here falls within the guidelines prescribed in section 5-5 of the ASME PTC 19.5-2004 [1]. The primary design constraint is that it must be printed within the limited area of common desktop fused filament fabrication 3D printers. The second design constraint requires that the inlet be dimensionally consistent with a common DC fan diameter that is used for electronics cooling. This consistency allows for affordable and accessible experimentation of the venturi tube, while also serving the purpose of characterizing airflow of a fan. The 3D printed design will also have taps in the recommended locations, which allows for pressure measurement using a differential pressure sensor.

The Ender 3 desktop 3D printer is chosen as the printer for the venturi tube, as it has been widely applied to prototyping in robotics, design of medical devices, and manufacturing [8, 9, 10, 11, 12, 13, 14]. The Ender 3 has a print volume of 220mm x 220mm x 250mm (length, width, height), marking the upper limit on the length of the tube at 250mm. A fan diameter of 80mm was chosen based on this length limit and the commonality of DC fan diameters ranging from 40mm - 120mm in the electronics consumer market.

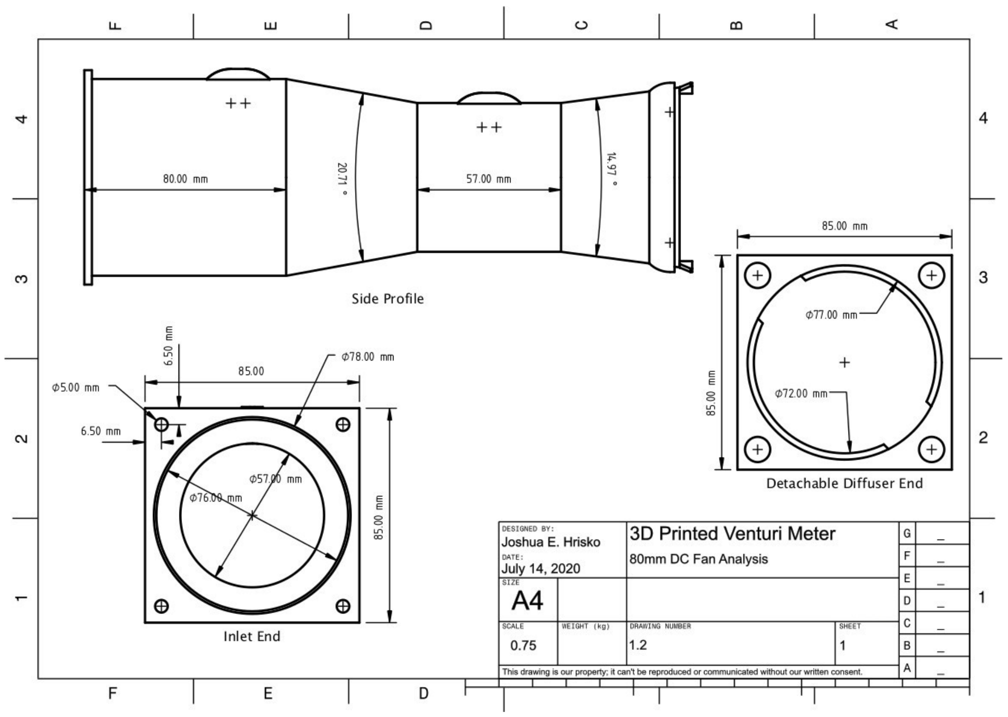
The actual inlet diameter was found to be 76mm, once the thickness of the fan walls were included. As per the guidelines of the ASME test code, the throat must fall within $0.25D$ - $0.75D$ [15, 16]. Given that we are limited by a length constraint, the diameter of the throat is chosen as 57mm, which happens to be the upper limit of the inlet diameter ratio, $0.75D$. The converging section from the 76mm inlet, D, to the 57mm throat, d, must be $21° \pm 1°$. Thus, after some calculations, the venturi tube from inlet to end of throat is already nearly 200mm, which led to the truncation modification of the outlet section of the venturi tube.

The outlet section (diffuser) is chosen to expand at an angle of $15°$ and is permitted a truncation of up to 35%—both constraints that are outlined in the ASME test code. Again, after some calculations the truncation of 35% results in an approximate length of 235mm, which falls within the printing height of the Ender 3 printer. Additional constraints imposed are that the inlet length is at least the diameter of the inlet (80mm was chosen here, in place of 76mm - which is still permitted), and the throat length is limited to the diameter of the throat (57mm). Lastly, the pressure taps are located and sized as required by ASME: the inlet tap is placed at a quarter of the distance from the start of the converging section and is less than $0.1D$, and the throat tap is placed halfway from the beginning of the throat, but is sized slightly smaller than the recommended $0.13d$. These specifications completely constrain the venturi tube and allow it to be printed at the recommendation of the ASME design.

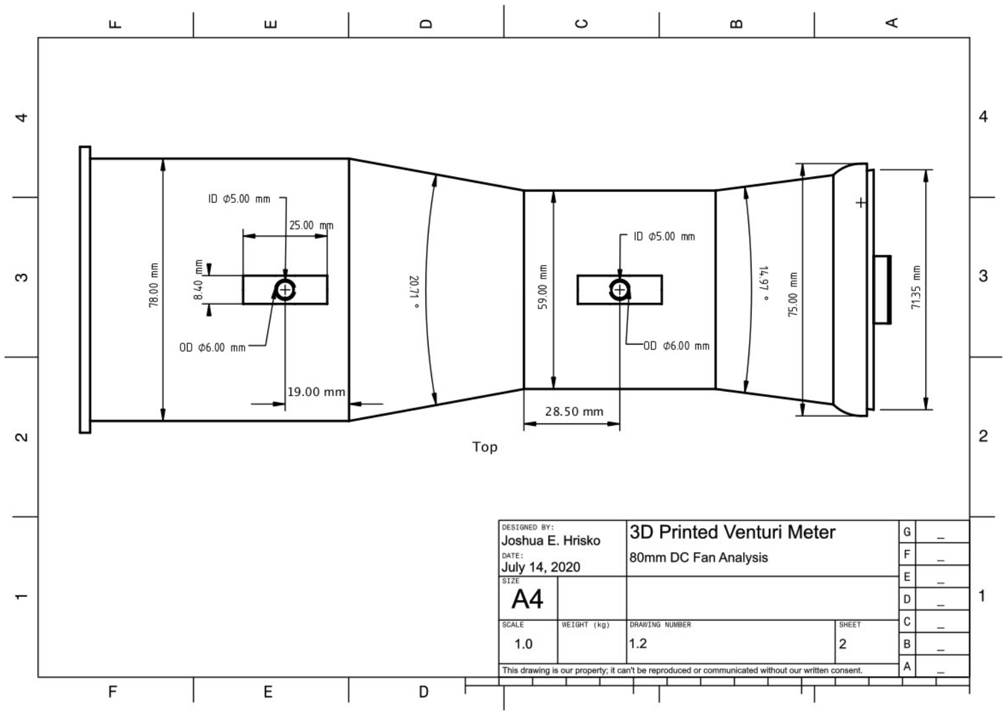
The technical drawing for the 76mm venturi tube can be found in Figs. 2a and 2b. The designs closely follow the design constraints outlined for the ASME venturi, while also being optimized for 3D printing. The particular difficulty in designing the meter was in the printing speed. The meter was designed to be squared at both ends to facilitate the shape of the DC fan; however, this results in a *much* longer print time due to the machine's need to create supports beneath the diffuser end. Consequently, a removable diffuser stand was developed to speed-up print times. With a 0.6mm nozzle affixed to the extruder of the Ender 3, and printing at a layer height of 0.32mm with a print speed of 50mm$\cdot$s^{-1}, the print time for the venturi fabrication is roughly 5.5 hours. Other configurations can produce more accurate and smoother prints (smaller nozzle and layer height), but at the detriment of longer print times.

3.2 Measurement and Control Devices

An Arduino board will serve as the DC fan controller and data acquisition device for the ensuing experimentation. Arduino is particularly specialized for this application due to the platform's integration of on-board analog-to-digital



(a)



(b)

Figure 2: Technical drawing for 3D printed venturi tube. (a) Profile drawing of the 76mm 3D printed venturi tube. An axial view of the inlet is also given, showing the radii for both the inlet and throat. Additionally, the detachable diffuser end is shown, which speeds up the printing time. (b) Top view of the 3D printed venturi tube, showing the tap locations and dimensions.

converters, complex digital protocol libraries, and the ability to use pulse-width modulation to control motors (a DC fan in our case). The open-source Arduino platform has been widely recognized as a useful educational tool for undergraduate engineering, likely due to its simplified approach to electronics software and hardware integration [17, 18, 19, 20]. Fortunately, the sensors used to measure differential pressure, barometric pressure, and temperature have simple libraries that make the data acquisition easy and straightforward.

The variables required for quantifying the velocity of the inlet and throat are displayed in Eqn. 53. The sensors chosen for the experiment must collectively measure: barometric pressure, differential pressure between the throat and inlet, and ambient temperature. These three measurements will allow the near complete characterization of the venturi tube across a range of velocities (Reynolds numbers). The pressure differential can be measured using an MPXV7002DP sensor, which converts the pressure differential between two nodes to an analog signal from -2kPa to +2kPa. The barometric pressure and standard temperature can both be acquired using one common sensor produced by Sensortec, Bosch: the BME280 [21].

The wiring between the BME280, MPXV7002DP, DC motor, and Arduino board is given diagrammatically in Fig. 3 and by component pins in Table 1. Note that the DC fan operates at 12V but shares a ground with the Arduino for PWM purposes. Pin 3 on the Arduino controls the speed of the fan using pulse-width modulation (PWM), altering the velocity through the venturi tube. The fan is an 80mm x 38mm (diameter, thickness) 12V DC, dual ball bearing fan that alters its rotation speed (RPM) according to different input duty cycles (PWM pulses). The variations in speed result in changes in pressure differential across the inlet and throat pressure taps of the meter, which is measured by the MPXV7002DP.

The MPXV7002DP only measures the pressure differential between the inlet and tap. It is powered by the 3.3V pin on the Arduino, and its analog signal is measured by the Arduino's first analog input pin, A0. Lastly, the barometric pressure and ambient temperature are captured by the BME280, which communicates via I2C protocol with the Arduino board on pins A4/A5. The BME280 is placed outside the venturi tube, which means that the air density will always be slightly lower than that inside the venturi tube inlet (ρ_1 in Eqn. 53). However, the inherent error of the BME280 device ($\approx 100Pa$)—in conjunction with the error associated with 3D printing—far exceed that of the pressure differential between the venturi inlet and atmospheric conditions. Thus, the location of the BME280 has a negligible impact on calculations, particularly in the region of low velocities presented in this study.

One important note is that the MPXV7002DP pressure nodes must be recorded properly in association with the respective venturi tube taps. This means that running a test with the meter tubing in one configuration is important for understanding how each node is connected. For example, if the pressure differential reads negative, then the nodes are reversed (assuming the fan is blowing from inlet to throat). This relates to a subtraction of the inlet node from the throat node—a higher pressure being subtracted from a low pressure. If the differential is positive, then it is in the correct configuration, i.e. the throat node is being subtracted from the inlet node.

The Arduino Uno board is shown in Fig. 3. It is capable of 8-bit PWM output, allowing for control in intervals from 0-255, or 0%-100% in steps of 0.4%. Additionally, the Arduino has an analog-to-digital converter (ADC) with a resolution of 10-bits. When considering the MPXV7002DP, it operates between 0.33V-2.97V over a pressure range of 4kPa, giving an approximate measurement resolution of 5Pa/bit [22]. Therefore, the Arduino is only capable of resolving pressure changes of 5Pa between the venturi inlet and throat taps.

4 Experimental Procedure

The process of testing the venturi tube involves sweeping over a range of fan rotation speeds (PWM pulses, or RPM values) and collecting several data points at each of those RPM speeds. In Appendix I, the Arduino code is given that loops through a range of PWM values and acquires data from each sensor. The Arduino code fully automates the process and repeats it indefinitely. Another code, given in Appendix II, gives the Python code used to acquire and save data in real-time from the Arduino outputs. The Python code is responsible for reading the PWM value, the pressure differential from the MPXV7002DP, and the barometric pressure and ambient temperature from the BME280 sensor. The Python code then exits the acquisition and saves the data upon the user pressing CTRL+C on the keyboard. Finally, the Python code plots the differential pressure and barometric pressure as a function of PWM. This allows users to see the results directly after the experiment to determine whether the tests were successful.

The general procedure of the Arduino automation code is as follows:

1. Turn the DC fan off, wait 2 seconds
2. Print out the headers to notify the data acquisition system (Python) that the experiment has started
3. Read 20 analog signals from the MPXV7002DP sensor and average them

Table 1: Wiring connections for each sensor to the Arduino Uno board.

Arduino Pin	MPXV7002DP	BME280	DC Fan	12V Supply
3.3V	3.3V	VCC		
GND	GND	GND	GND	GND
A0	ANALOG			
A4		SDA		
A5		SCL		
D3			PWM (Blue)	
			VCC (Red)	+12V
			GND (Black)	GND
*3.3V to AREF				

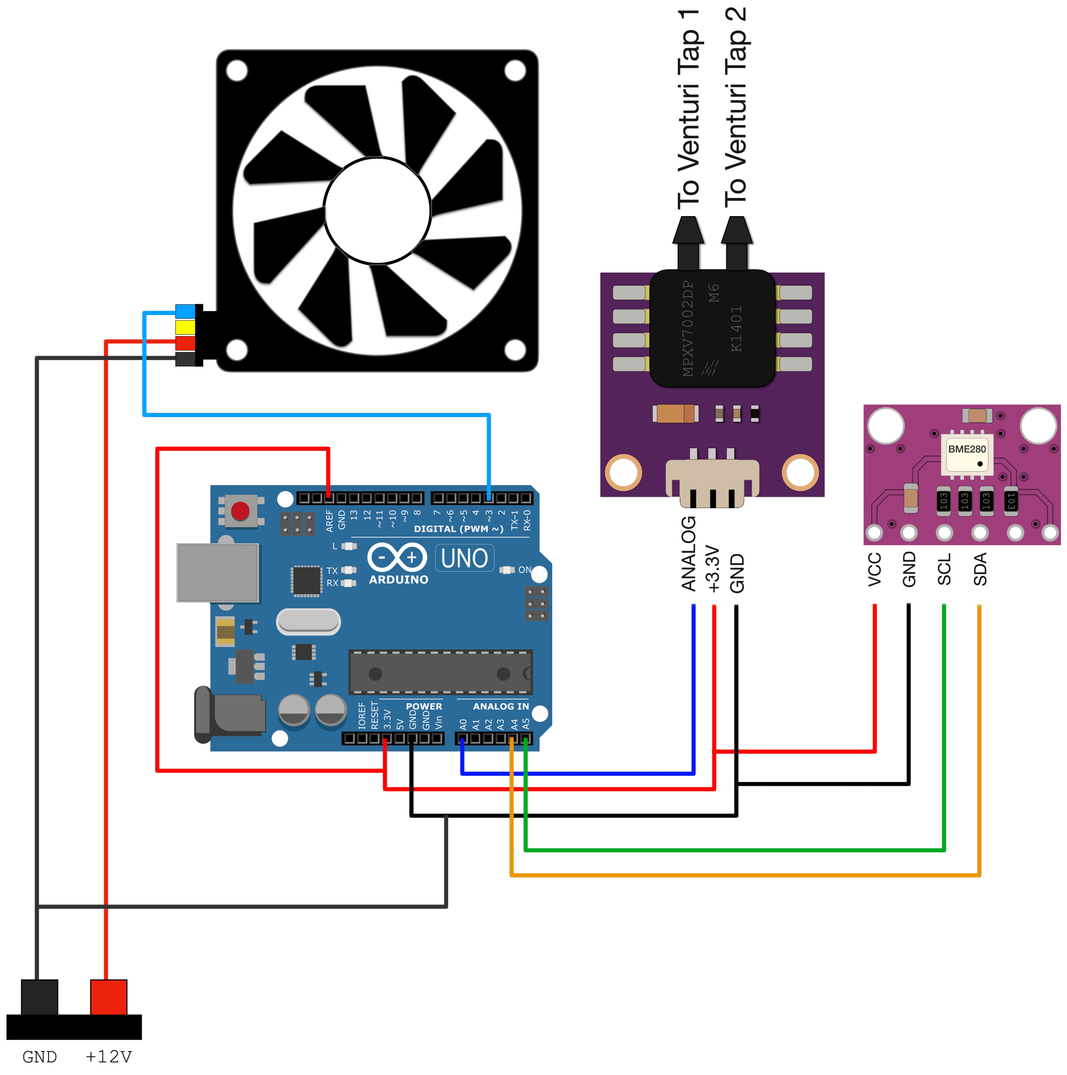


Figure 3: Wiring between Arduino and the 12V DC fan, BME280, MPXV7002DP and the 12V supply.

4. Convert the MPXV7002DP signal to differential pressure
5. Print out values for PWM, MPXV7002DP pressure differential
6. Read and print values from BME280 (pressure, temperature)
7. Wait 100 milliseconds, repeat steps 3 - 6 for 20 iterations (2s)
8. Increase PWM value by 10 (4%) on fan if below maximum (255), otherwise go back to zero
9. Wait 1 second to allow fan to stabilize, repeat steps 3-8

The process loop given above will run indefinitely until the user disconnects the Arduino board. What is most straightforward for an acquisition loop is to run the Python code, wait for the PWM pulses to span 0-255, then exit the program and unplug the Arduino board. This will give one full loop of the DC fan from off to maximum. After taking a few measurements and observing similar results over multiple runs, the data can be analyzed for quantification of the performances of the venturi tube and DC fan.

5 Results

5.1 Volumetric Flow Rate of the 80mm DC Fan

Python is again used to analyze the data acquired from the Arduino board and its sensors. There are four variables of interest, namely: differential pressure from the venturi tube, barometric pressure and ambient temperature measured by the BME280, and PWM pulse or duty cycle controlled by the Arduino board. These four are the primary variables used to analyze both the venturi tube and fan performance.

Using the headers printed out in the comma-separated values (CSV) file, the variables can be plotted and investigated. Equation 45 and 53 can be used to approximate the velocities through the throat of the venturi tube. For the DC fan used in our experiments, the velocity through the nozzle was found to range 5.5 - 15.0 m·s^{-1}. Therefore, calculation of the compressibility factor, ϵ, for our data shows that it never falls below 0.999, which is approximated to 1 at a greater degree of accuracy than our measurement error. Thus, Eqn. 45, the incompressible assumption, suffices for calculations going forward. After determining the range velocities above, the Reynolds number can also be calculated for flow through the venturi:

$$Re_d = \frac{\rho U_2 d}{\mu} \tag{54}$$

where ρ and μ are determined using the pressure and ambient temperature measured by the BME280. The diameter of the throat is again, d, and U_2 is the velocity approximated using Eqn. 45. The expressions used for density is based on the ideal gas law (Eqn. 9):

$$\rho = \frac{p}{R_s T} \tag{55}$$

where the pressure and temperature from the BME280 are used for p and T, respectively. Similarly, the dynamic viscosity is determined using the temperature from the BME280 and Sutherland's law [23]:

$$\mu = 1.716 \times 10^{-5} \left(\frac{T}{273.15}\right)^{3/2} \left(\frac{273.15 + 110.4}{T + 110.4}\right) \tag{56}$$

For the velocity ranges calculated in our experiments, the Reynolds number spans 20k - 55k for the venturi tube and 80mm DC fan flow rate. This range, according to the ASME performance test code, is still well within the laminar boundary layer region. An added correction factor called the discharge coefficient must be added to the calculation of velocity in any venturi tube, which is simplified for laminar boundary layers as follows:

$$C = 1.0054 - 6.88(\mathrm{Re}_d)^{-0.5} \tag{57}$$

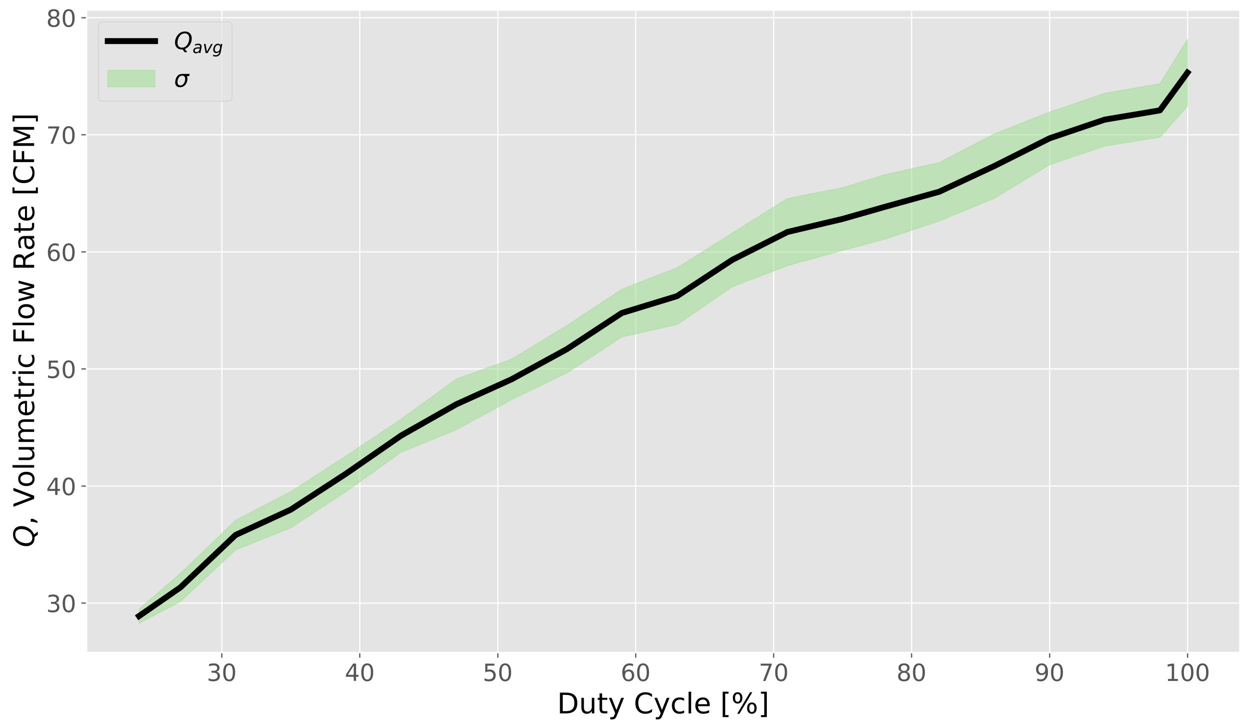


Figure 4: Volumetric flow rate of the 80mm DC fan as a function of duty cycle. The approximate maximum flow rate was found to be 75.3 cubic feet per minute (CFM).

where C is the effective discharge coefficient for the low velocity laminar boundary layer of our 3D printed venturi tube. Finally, the volumetric flow rate can be calculated using the conversion to cubic feet per minute (CFM) and the measured and derived variables of the experiment:

$$Q = 2118.88\pi \left(\frac{d}{2}\right)^2 \left(1.0054 - 6.88 \left(\frac{\rho d}{\mu}\right)^{-0.5} \left(\frac{2\Delta p}{\rho(1-\beta^4)}\right)^{-0.25}\right) \left(\frac{2\Delta p}{\rho(1-\beta^4)}\right)^{0.5} \tag{58}$$

Equation 58 is the final expression for approximating the volumetric flow rate, Q, in units of cubic feet per minute (CFM), for the 80mm DC fan being tested. The units of CFM are common for centrifugal devices, and is used here to verify the manufacturer's reported maximum flow rate.

The pulse-width modulation controlled by the Arduino board spans 0 - 255, which translates to a duty cycle of 0-100%. The conversion from PWM bits to duty cycle can be written as follows:

$$DT = 100 \cdot \frac{\text{PWM}}{255} \tag{59}$$

The duty cycle, DT, will be used to determine the relationship between duty cycle and airflow in CFM - a common pairing when comparing the efficiency of a fan.

The fan used for the experimentation has dimensions of 80mm x 80mm x 38mm. There was no datasheet available for the fan used in our experiments, however, similar dual ball bearing fans cite volumetric flow rates ranging from 60-80 CFM for their 80mm fans [24, 25, 26]. Accordingly, we expect to see a similar maximum flow rate for our 80mm fan being tested.

Figure 4 shows the Arduino duty cycle as a function of volumetric flow rate calculated using Eqn. 58. The plot shows the average and standard deviation for each duty cycle value ranging from 24% - 100%. The average maximum value of Q was calculated as 75.3 CFM, slightly higher than the cited 72.6 CFM by the manufacturer, but within the bounds of standard deviation. Figure 4 also shows a somewhat linear relationship between duty cycle and Q for duty cycles below 60%, then the slope appears to flatten perhaps to an exponential or logarithmic profile. Another important note is that the fan does not turn when the duty cycle is below 24%.

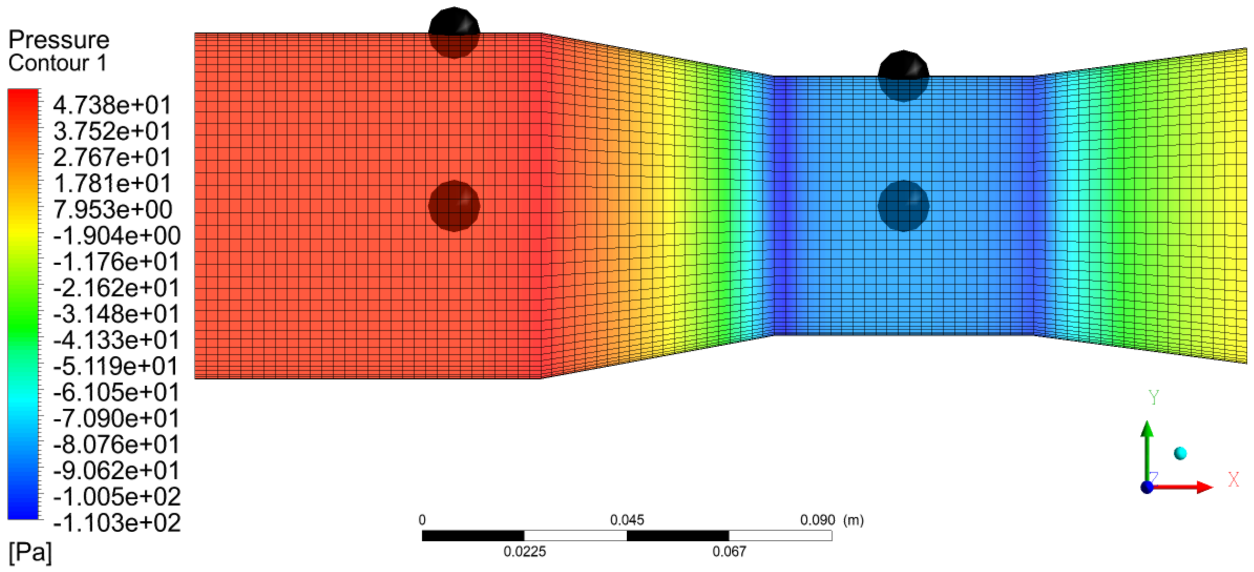


Figure 5: Pressure contour taken from the Ansys Fluent simulation at the peak inlet velocity, 9.5 m·s^{-1}. The two top-most black balls represent the recorded pressure locations, while the two axial midpoint black balls mark the velocity recording locations.

5.2 Computational Fluid Dynamics Model

The computational fluid dynamics (CFD) software, Ansys Fluent, was used to model and simulate the 3D printed venturi tube across the range of velocities generated by the 80mm fan. First, the 76mm venturi tube was modeled in its simplest form as a 3-dimensional pipe with the same dimensions as described in 3.1. The state conditions for temperature, atmospheric pressure, and density were inputted into the CFD model based on the measurements in the experiment. A shear stress transport (SST) k-ω model is used to determine the flow rate across the inlet and throat, and is based on transport equations for the turbulent kinetic energy (k, or TKE) and specific dissipation rate (ω) [27]. SST $k-\omega$ models can be found applied to venturi tubes across the literature, at much lower β values and higher Reynolds numbers, thus, it is applied here without concern for misapplication [28, 29, 30, 31].

The venturi tube was modeled with the dimensions identical to those given in the technical drawing in Figs. 2a and 2b, excluding the pressure taps, inlet frame, and diffuser end's asymmetric components. This resulted in a clean model with four sections: the inlet, converging section, throat, and the truncated diffuser section. The simplified model of the 3D printed venturi tube was meshed in 3 dimensions using 408k nodes and 99k elements. The orthogonal quality of the mesh ranged from 0.54 - 0.99, with an average of 0.98—indicating very high quality meshing [32].

The wall roughness was also modified to 32 μm in accordance with observations made by Alsoufi and Elsayed for 3D printed nozzle diameters and layer heights [33]. The resulting unsteady turbulent CFD model ran for 100 time cycles with a user-defined velocity function that linearly incremented under the following condition:

$$u(t) = \frac{6.68t}{10} + 2.82 \tag{60}$$

where t is the time vector that incremented from 0-10s at intervals of 0.1s. The resulting velocity spanned 2.82 m·s^{-1} - 9.5 m·s^{-1} to mirror the approximate inlet velocities derived from the values of Q in Fig. 4. The residuals of the 3 Navier-Stokes, k (TKE), and ω equations all converged to 10^{-6}, while the continuity converged to 10^{-4}—all of which are more than acceptable convergence conditions [34, 35, 36]. The time steps were saved after each convergence, which resulted in 100 points in time at 0.1s intervals. Pressures were recorded at the throat and inlet taps (see Fig. 2b for reference), whereas the velocities were recorded along the centerline of the venturi tube, under the pressure tap locations. Figure 5 shows an example pressure contour with the location of each measurement point marked using a black ball.

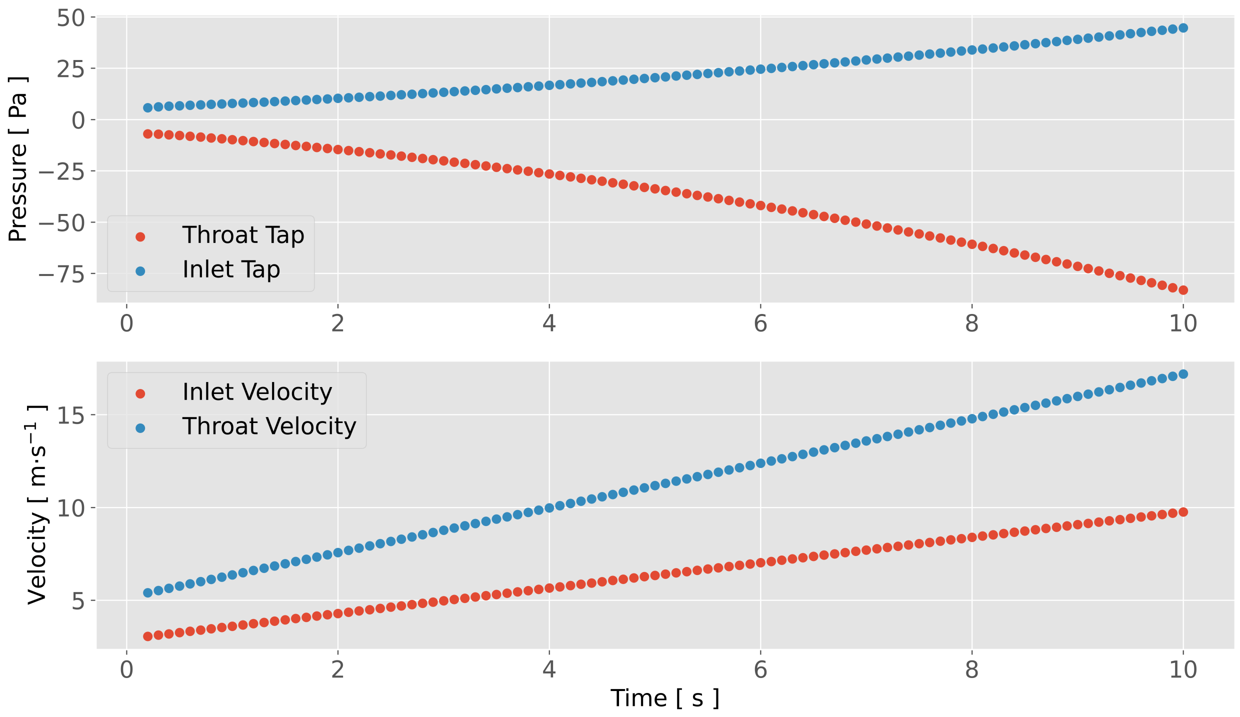


Figure 6: Timeseries profiles taken for inlet and throat for pressure and velocities recorded from the Ansys simulation results.

5.3 Comparison Between Theory and CFD Model

The alignment of pressure tap locations and velocity ranges allowed for direct comparison between the flow rate calculated from the venturi tube and the flow rate simulated and recorded by the CFD simulation. It is immediately clear from the contour in Fig. 5 that the tap locations are well placed, as there appears to be very consistent pressure distributions at those axial positions. If the inlet tap were to be placed closer to the converging section, the pressure measurement may not be indicative of the inlet section, and the same can be said for the throat tap.

Figure 6 shows the timeseries plots for pressure and velocity at the four probe points for the Ansys simulation. The velocity at the inlet can be seen to follow the approximate velocity measured in the Arduino experiments, just as intended. Figure 7 shows the direct comparison between 3D printed venturi tube experiments and the Ansys simulation results for volumetric flow rate. The red line represents the volumetric flow rate, Q, computed using the velocity probe located near the inlet, the blue line represents Q calculated using the velocity measured at the throat probe, and the dotted black line represents Q derived using the Reynolds-corrected, incompressible assumption in Eqn. 58.

The calculation of Q from the probed Ansys inlet velocity is used as the true value for volumetric flow rate; the reason being, this value is the closest to the value outputted by the DC fan at the inlet. Thus, the mean absolute errors are given in Fig. 7 by direct comparison with the inlet simulation results. The calculated error between the incompressible venturi assumption can be associated with the approximate error of the 3D printed venturi results given in Fig. 4. Another conclusion that can be drawn here is that the combined error between 3D printing dimensions and Arduino analog-to-digital conversion for the MPXV7002DP is far above the error found between the Ansys simulation and the incompressible venturi tube correction given for the ASME performance test code.

6 Conclusion

A 3D printed venturi tube was designed under the specifications outlined in the ASME performance test code on flow measurement. The intention was to use the dimensions of a common DC cooling fan to develop an inexpensive solution to testing flow rates across several operational modes. This would allow for more efficient control and cooling of electronic devices. An inlet diameter of 76mm was chosen for the venturi tube, with a throat diameter of 57mm

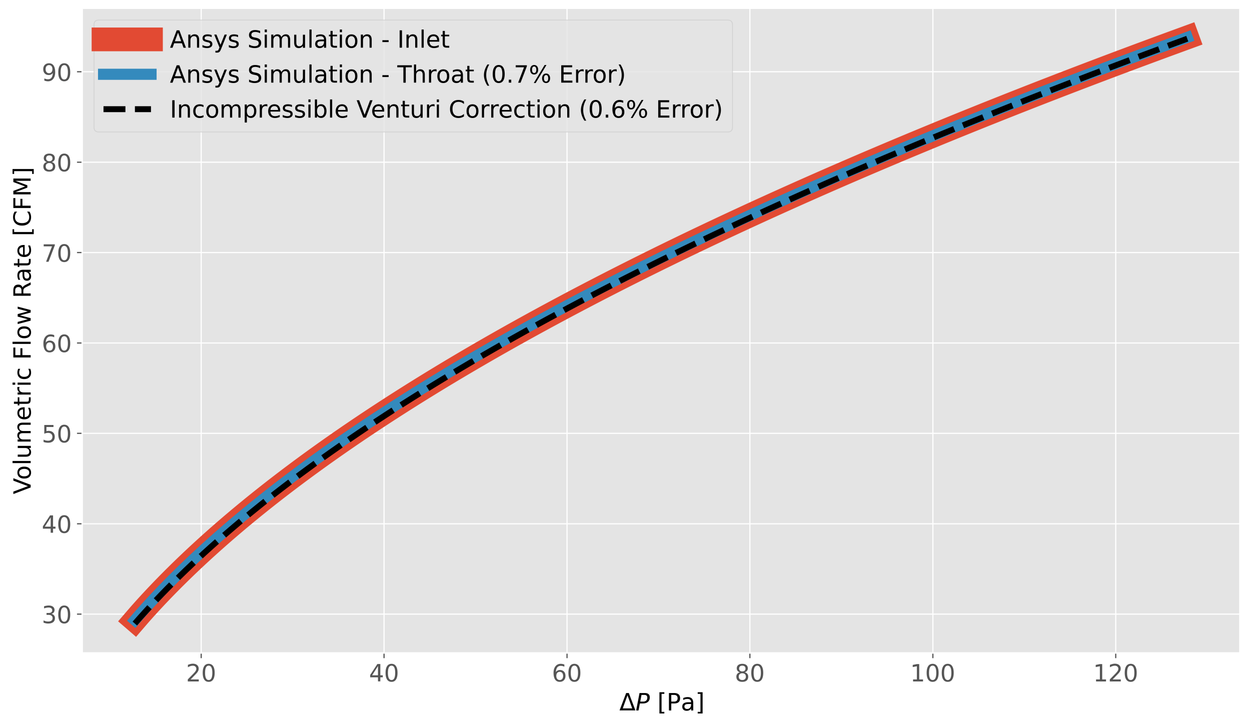


Figure 7: Comparison between Ansys Fluent CFD model results for the venturi tube and the incompressible venturi tube equation. The error between the incompressible venturi correction and the Ansys inlet simulation results can be associated with the 3D printed venturi experiments, indicating that error of the incompressible venturi equation is much lower than the error associated with the dimensions of the 3D printing or even the MPXV7002DP pressure sensor and Arduino's analog-to-digital converter.

($\beta = 0.75$), which were specifically chosen to minimize the height of the venturi tube in order to fit within the printing volume of most desktop 3D printers.

The resulting device was tested using an 80mm x 38mm DC fan attached at the inlet section of the meter, along with 5mm tubing attached to the inlet and throat taps. The opposite ends of the taps were then attached to the MPXV7002DP differential pressure sensor that was wired to an Arduino Uno board. Using the Arduino microcontroller, the fan rotation speed was controlled using pulse-width modulation. Two sensors were also connected to the Arduino board, which measured barometric pressure and ambient temperature (BME280), and differential pressure between the inlet and throat taps (MPXV7002DP).

The sensor data was recorded from the Arduino board and saved using a Python script for later analysis. Following the equations prescribed in the ASME standard on flow measurement, the volumetric flow rate, Q, was approximated. The resulting flow rate was plotted against the fan duty cycle, where the maximum duty cycle flow rate was compared directly with the DC fan's manufacturer's cited flow rate. The two values agreed within 4%, with the venturi tube measuring a maximum flow rate of 75.3 CFM, and the manufacturer citing 72.6 CFM.

Lastly, an Ansys Fluent computational fluid dynamics (CFD) model was introduced as a way of verifying the calculated flow rate of the 80mm fan and to verify the range of velocities measured by the venturi tube. Using a shear stress transport $k - \omega$ turbulence model, a simplified geometry of the venturi tube was simulated across a range of velocities from 2.8 - 9.5 m$\cdot$s^{-1}. The resulting pressure distribution was probed at inlet and throat taps to compare directly with the pressures measured by the 3D printed venturi tube. The flow rate probed at the inlet of the CFD simulation was used as the comparison point for the simulated flow rate in the throat and the approximated flow rate based on the pressure differential at the meter taps.

The incompressible flow approximation of the volumetric flow rate agreed with the CFD simulation within 0.7% for both the inlet and throat, indicating that the incompressible venturi approximation—with the Reynolds number-dependent discharge coefficient—is very accurate across the range of flow rates and pressure conditions analyzed. Another thing to note is that the error associated with the CFD model and incompressible venturi equation are far below the error associated with the Arduino analog-to-digital converter and the MPXV7002DP differential pressure sensor. The conclusion that can be drawn from this observation is that the flow rate measured by the 3D printed venturi tube is well within the modeled values and agrees with the manufacturer's cited value to within one standard deviation of measurement. Consequently, a 3D printed venturi tube is an affordable, accurate, and rapid way of characterizing a low-speed DC fan.

Acknowledgement

Recognition to Matthew Hrisko for editorial comment and review.

Appendix I - Arduino Automation Code

Code 1: Arduino code to control 12V DC fan via PWM from 0 - 100%, while simultaneously measuring the differential pressure of the inlet and throat using the MPXV7002DP sensor, and the barometric pressure and ambient temperature using the BME280 sensor.

```
// Pressure differential from MPXV7002DP
// with DC Fan controller via PWM, and a
// BME280 taking pressure, temp measurements
//
#include <Wire.h>
#include <Adafruit_Sensor.h>
#include <Adafruit_BME280.h>

Adafruit_BME280 bme; // I2C for BME280

// pin allocations for MPXV7002DP and fan PWM
int mpxv_pin = A0; // mpxv analog input pin
int fan_pin = 3; // PWM pin for dc fan

float V_0 = 3.3; // supply voltage to the pressure sensors

// parameters for averaging and offset
int offset = 0;
int offset_size = 10;
int mean_size = 20;
int zero_span = 2;

// dc fan pwm value
int pwm_val = 0; // PWM value for controlling fan speed
int iter_change = 0; // iterations to keep fan at specific speed
bool max_val = false; // when max PWM is reached (255), this sets it back to 0
int pwm_min = 0; // min pwm for loop
int pwm_max = 255; // max pwm for loop

// setup and calculate offset
void setup() {
  analogReference(EXTERNAL); // 3.3V instead of 5V for analog values
  pinMode(fan_pin,OUTPUT); // setup PWM for fan

  digitalWrite(fan_pin,0); // make sure fan is off
  delay(2000);
  Serial.begin(9600); // start serial port
  delay(100);
  Serial.println("Acquisition Start"); // data acquisition start phrase
  Serial.println("PWM [0-255],dP[Pa],Pressure [Pa],Temp. [C]"); //header
  for (int ii=0;ii<offset_size;ii++){
    offset += analogRead(mpxv_pin)-(1023/2); // offset for MPXV7002DP
  }
  offset /= offset_size;

  unsigned status;
  status = bme.begin(); // make sure BME280 was wired/is working properly
    if (!status) {
        Serial.println("No BME280 sensor available");
        while (1); // if no BME280, stay in infinite loop
    }
}

void loop() {
  float adc_avg = 0; float delta_P = 0.0; // preallocations for variables

// average a few ADC readings for stability
  for (int ii=0;ii<mean_size;ii++){
    adc_avg+= analogRead(mpxv_pin)-offset;
  }
  adc_avg/=mean_size; // average ADC value for MPXV7002DP

  // make sure if the ADC reads outside of 512+-zero_span (this increases
  // stability around 0 pressure)
  if (adc_avg>512-zero_span and adc_avg<512+zero_span){
  } else{
    delta_P = 5000.0*((adc_avg/1023.0)-0.5); // actual differential pressure
  }
  Serial.print(pwm_val); // print PWM value
  Serial.print(","); // comma separate

  Serial.print(delta_P); // print delta P
  Serial.print(","); // comma separate

  Serial.print(bme.readPressure()); // print barometric pressure
  Serial.print(","); // comma separate

  Serial.println(bme.readTemperature()); // print temperature

  delay(100); // delay for stability

  // iterating the PWM values
  if (iter_change>20) {
    pwm_val+=10; // this can be changed to desired increase in PWM
    if (pwm_val>=pwm_max and max_val){
      pwm_val = pwm_min; // if we're at max, go to pwm_min
      max_val = false;
    }
    if (pwm_val>pwm_max and pwm_val!=pwm_max){
      pwm_val = pwm_max; // if we overshoot, go to pwm_max
      max_val = true;
    }
    iter_change = 0; // reset to get multiple points at PWM value
    analogWrite(fan_pin,pwm_val); // write the actual PWM value
    delay(1000); // allow fan to stabilize
  }
  iter_change+=1; // increase iter every loop
}
```

Appendix II - Python Data Acquisition Code

Code 2: Python code used to acquire data in real-time from the Arduino board and saves the data into a comma-separated value (CSV) file. The differential pressure and barometric pressure are subsequently plotted as a function of PWM.

```
##################################################
# Python + Arduino Datalogger
##################################################
#
```

```python
# This code reads from the serial port and keeps
# the data in memory until the user presses
# CTRL+C, wherein the data is then saved locally to
# a .csv file with the current timestamp. Lastly,
# the data is read from the file and plotted
#
##################################################
#
import numpy as np
import matplotlib.pyplot as plt
import serial,datetime,csv,os
import serial.tools.list_ports as COMs
#
#
############################################
# Find Arudino ports, select one,
# start communication with it
############################################
#
arduino_ports = [ii.device for ii in COMs.comports() if\
                 len((ii.device).split('ttyACM'))>1 or\
                 len((ii.device.split('ttyUSB')))>1]
ser = serial.Serial(arduino_ports[0],baudrate=9600) # match baud on Arduino
ser.flush() # clear the port
#
############################################
# Grabbing real-time data from Arduino
############################################
#
t_now = datetime.datetime.strftime(datetime.datetime.now(),
                                   '%Y_%m_%d_%H_%M_%S')
datafile = 'arduino_data_'+'{0}'.format(t_now)+'.csv' # date
data_array = []; start_bool = False
while True:
    try:
        ser_bytes = ser.readline() # read Arduino serial data
        decoded_bytes = ser_bytes.decode('utf-8') # decode data to utf-8
        data = (decoded_bytes.replace('\r','')).replace('\n','')
        if start_bool==False and data=='Acquisition Start':
            # read the first line after acquisition start as header
            header = ((ser.readline()).decode('utf-8').replace('\r','')).\
                     replace('\n','')
            start_bool = True
            print('Data Acquisition Starting...')
            ser.flush()
            continue
        if start_bool:
            # saving data to variable to incrase speed
            # (saving to file happens after keyboard interrupt)
            data_array.append([float(ii) for ii in data.split(',')])
    except KeyboardInterrupt:
        print('Exiting Loop and Saving Data')
        if data_array == []:
            print('No Valid Data') # if no data, print out and exit
            break
        # save valid data in array to .csv file
        data_array = np.array(data_array) # convert to numpy array
        with open(datafile,'w') as csvfile:
            csv_writer = csv.writer(csvfile,delimiter=',')
            csv_writer.writerow([header])
            for row in data_array:
                csv_writer.writerow(row)
        break # finally, exit loop after save
#
############################################
```

```
# Read back the data file and plot the data
############################################
#
if os.path.isfile(datafile):
    data_check = []
    with open(datafile,'r') as csvfile:
        csvreader = csv.reader(csvfile) # read the saved file
        header = next(csvreader)[0].split(',')
        for row in csvreader:
            data_check.append([float(ii) for ii in row])

    data_check = np.array(data_check) # convert to numpy array

    # Plot the data from the saved file to verify
    plt.style.use('ggplot')
    fig,axs = plt.subplots(2,1,figsize=(12,9))
    ax1 = axs[0]
    ax1.scatter(data_check[:,0],data_check[:,1],label=header[1])
    ax1.legend()
    ax1.set_xlabel(header[0]) # use the header for x-label
    ax1.set_ylabel(header[1]) # use the header for y-label
    ax2 = axs[1]
    ax2.scatter(data_check[:,0],data_check[:,2],label=header[2])
    ax2.legend()
    ax2.set_xlabel(header[0]) # use the header for x-label
    ax2.set_ylabel(header[2]) # use the header for y-label
    plt.show() # show the plot
```

References

[1] American Society of Mechanical Engineers. Flow Measurement - PTC 19.5 - 2004. *American Society of Mechanical Engineers*, 2004.

[2] Zachary B Sharp, Michael C Johnson, and Steven L Barfuss. Optimizing the ASME Venturi recovery cone angle to minimize head loss. *Journal of Hydraulic Engineering*, 144(1):04017057, 2018.

[3] Ben Wittbrodt and Joshua M Pearce. The effects of PLA color on material properties of 3-D printed components. *Additive Manufacturing*, 8:110–116, 2015.

[4] Vladimir E Kuznetsov, Alexey N Solonin, Oleg D Urzhumtsev, Richard Schilling, and Azamat G Tavitov. Strength of PLA components fabricated with fused deposition technology using a desktop 3D printer as a function of geometrical parameters of the process. *Polymers*, 10(3):313, 2018.

[5] Robert P Benedict. *Fundamentals of pipe flow*, chapter 1. Wiley,, 1980.

[6] John David Anderson. *Modern compressible flow: with historical perspective*, volume 12, chapter 1. McGraw-Hill New York, 1990.

[7] Robert T Balmer. *Modern engineering thermodynamics*, chapter 7. Academic Press, 2011.

[8] Mounika Sakhineti and Sudharsan Jayabalan. Design and fabrication of SHRALA: Social humanoid robot based on autonomous learning algorithm. *Procedia Computer Science*, 171:2050–2056, 2020.

[9] LI Culda, ES Muncut, and A Komjaty. Rapid filament prototyping of a bionic hand. In *IOP Conference Series: Materials Science and Engineering*, volume 568, page 012105. IOP Publishing, 2019.

[10] Mateusz Maciej Gaździński. *3D Printing Using Fused Deposition Modeling in the Injection Moulding Technology*. PhD thesis, Instytut Technik Wytwarzania, 2019.

[11] Poposky Cherichel and Rodolfo de Curtis. EMG-controlled mechanical hand. *Journal of Innovative Ideas in Engineering and Technology*, 1(1):1–38, 2020.

[12] SA Peleshok, DA Volov, MV Titova, MI Eliseeva, VN Adamenko, and Ya I Nebylitsa. Development of 3D printing techniques individual splints for brush and forearm immobilization with injury. *Russian Military Medical Academy Reports*, 38(2):34–40, 2019.

[13] R Srinivasan, W Ruban, A Deepanraj, R Bhuvanesh, and T Bhuvanesh. Effect on infill density on mechanical properties of PETG part fabricated by fused deposition modelling. *Materials Today: Proceedings*, 2020.

[14] Leonard Bunting, Steven Roy, Hannah Pinson, Tobin Greensweig, International Differential Multi-Ventilation Working Group, et al. A novel inline PEEP valve design for differential multi-ventilation. *The American Journal of Emergency Medicine*, 2020.

[15] Walt Boyes. *Instrumentation reference book*, chapter 6, pages 37–38. Butterworth-Heinemann, 2009.

[16] E Shashi Menon. *Transmission pipeline calculations and simulations manual*, chapter 12, pages 467–469. Gulf Professional Publishing, 2014.

[17] Jaroslav Sobota, Roman PiŜl, Pavel Balda, and MiloŜ Schlegel. Raspberry Pi and Arduino boards in control education. *IFAC Proceedings Volumes*, 46(17):7–12, 2013.

[18] Radhika Grover, Shoba Krishnan, Terry Shoup, and Maryam Khanbaghi. A competition-based approach for undergraduate mechatronics education using the Arduino platform. In *Fourth Interdisciplinary Engineering Design Education Conference*, pages 78–83. IEEE, 2014.

[19] Peter Jamieson and Jeff Herdtner. More missing the boat—Arduino, Raspberry Pi, and small prototyping boards and engineering education needs them. In *2015 IEEE Frontiers in Education Conference (FIE)*, pages 1–6. IEEE, 2015.

[20] Anees Bashir, Mariam Alhammadi, Moath Awawdeh, and Tarig Faisal. Effectiveness of using Arduino platform for the hybrid engineering education learning model. In *2019 Advances in Science and Engineering Technology International Conferences (ASET)*, pages 1–6. IEEE, 2019.

[21] Bosch Sensortec. BME280 Combined humidity and pressure sensor, 2018.

[22] NXP Semiconductors. MPXV7002 Integrated Silicon Pressure Sensor, 2017.

[23] William Sutherland. LII. the viscosity of gases and molecular force. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, 36(223):507–531, 1893.

[24] Mechatronics. MQ8038 series, 2014. https://www.mechatronics.com/pdf/MQ8038.pdf.

[25] Orion Fans. OD8038-IP68 Series. https://orionfans.com/productFiles/datasheet/OD8038-IP68.pdf.

[26] Sanyo Denki. San Ace 80. https://www.farnell.com/datasheets/1878245.pdf.

[27] RODRICKV Chima. A k-omega turbulence model for quasi-three-dimensional turbomachinery flows. In *34th Aerospace Sciences Meeting and Exhibit*, page 248, 1995.

[28] J Kolář and V Dvořák. Verification of K-ω SST turbulence model for supersonic internal flows. *World Academy of Science, Engineering and Technology*, 81:262–266, 2011.

[29] A Diring, L Fromme, M Petry, and E Weizel. Comparison between COMSOL Multiphysics and STAR-CCM+ simulation results and experimentally determined measured data for a venturi tube, 2017.

[30] Hongbo Shi, Mingda Li, Petr Nikrityuk, and Qingxia Liu. Experimental and numerical study of cavitation flows in venturi tubes: From CFD to an empirical model. *Chemical Engineering Science*, 207:672–687, 2019.

[31] Hongbo Shi, Mingda Li, Qingxia Liu, and Petr Nikrityuk. Experimental and numerical study of cavitating particulate flows in a venturi tube. *Chemical Engineering Science*, page 115598, 2020.

[32] Joanna Karcz and Lukasz Kacperski. An effect of grid quality on the results of numerical simulations of the fluid flow field in an agitated vessel. In *Proceedings of 14th European conference on mixing*, pages 205–210, 2012.

[33] Mohammad S Alsoufi and Abdulrhman E Elsayed. How surface roughness performance of printed parts manufactured by desktop FDM 3D printer with PLA+ is influenced by measuring direction. *American Journal of Mechanical Engineering*, 5(5):211–222, 2017.

[34] Akshay Parab, Ammar Sakarwala, Vaibhav Patil, and Amol Mangrulkar. Aerodynamic analysis of a car model using Fluent-Ansys 14.5. *International Journal on Recent Technologies in Mechanical and Electrical Engineering*, 1(4):07–13, 2014.

[35] G Satyanarayana, Ch Varun, and SS Naidu. CFD analysis of convergent-divergent nozzle. *Acta Technica Corviniensis-Bulletin of Engineering*, 6(3):139, 2013.

[36] R Lanzafame, S Mauro, and M Messina. Wind turbine CFD modeling using a correlation-based transitional model. *Renewable Energy*, 52:31–39, 2013.