# Laboratory 8:
# DAQ 3 - Calibration of a Pressure Sensor

## 8.1 Introduction

Many times in engineering, you will want to make measurements of quantities - such as height, pressure, or volume - that are not electrical. To take these measurements with the data acquisition system, you must use a *sensor* or *transducer* to convert one kind of signal to another.

For this lab specifically, you will be developing a system whereby the computer can measure the volume of fluid in a graduated cylinder. To do this, you will be using a pressure sensor - a device which converts a differential pressure across a membrane into an electrical measurement. The pressure can be related to the height of fluid in the cylinder, and the height can then be related to the volume. Through this process, you will be engineering one of the tools of scientific discovery - an electronic measurement system of a physical quantity.

## 8.2 Safety

Please note that as always when working with the DAQ cards, your safety is paramount, followed by the safety of the equipment. The safety notice for all DAQ labs still applies. Please revisit the rules that come into play when doing one of the DAQ labs in the handout for Lab 4. In addition - *Respect The Water.* Needless to say[1], tipping the cylinder or bucket (or both) is to be avoided if at all possible. There are wires connecting the pressure sensor to the DAQ card - make sure that you do not pull on these wires to the point of tipping the cylinder over or sending it off the table. Furthermore, there is a tube clipped to the edges of the reservoir bucket and the graduated cylinder - do not pull on this or allow it to pull the cylinder over. Also, because of the configuration of the room, remember that there may be a cylinder and bucket *behind you.* Control whatever urges you have to stretch your arms behind you and make sure you are cognizant of the amount of clearance your bookbag may require if walking down the aisles or otherwise moving about the room. If you are lifting up the bucket, make sure that you have a firm grasp on it and that you are paying attention to the level of water both in the bucket and in the cylinder. If you are pinching the fill tube closed, make sure not to pull the cylinder over.

[1] But I will say it anyway...

## 8.3 Resources

The additional resources required for this assignment include:

- Books: None
- Pratt Pundit Pages: `MATLAB:CB-68LP Pinout`, `http://pundit.pratt.duke.edu/wiki/User:DukeEgr93/DAQ3`
- Lab Manual Appendices: None

Note - the second Pundit page listed contains pictures of the system as well as a couple graphs MATLAB should show you during the lab. You may therefore want to have Pundit open on the PC *not* being used for data acquisition.

## 8.4 Getting Started

1. Each lab group should have one person log into the PC that has the patch of green tape on it. Log in using the NET ID `mrglocal` and the password `p1p2dell` (that is p-ONE-p-TWO-d-e-l-l). Be sure it is set to log on to "this computer." The other lab partner can log into the other computer either using the `mrglocal` account or any other valid NET ID (as log as the "Log on to" is set to `Kerberos`). You do *not* need to run X-Win, PuTTY, or MATLAB on that second computer.
2. Start MATLAB on the PC with the green tape on it. The first time MATLAB is run, it may take some time to start up.

## 8.5 Equipment

Be sure that you have the following inventory at your lab station:

- *Graduated Cylinder:* We will use a 1000 mL standard graduated "cylinder" with a pressure transducer attached to a short piece of tubing at approximately the 100 mL point. The word "cylinder" is in quotes, because it turns out these are not truly cylindrical but rather frusta of cones.
- *Fluid Reservoir and Tube:* To keep this lab as safe as possible, you will be changing the fluid level in the cylinder by using a closed system. On the lab bench, there should be a bucket of water with a tube attached to it; the tube also goes into the graduated cylinder. This system will use the principles of a siphon to maintain equal heights of fluid in the bucket and the cylinder. If you *carefully* lift the bucket off the bench, you should notice water flowing out of the bucket and into the cylinder, raising the surface of the water in the cylinder until it matches that in the bucket. Similarly, when you put the bucket back on the bench, the fluid will flow back out of the cylinder until the surfaces are again level.
- *Pressure Transducer:* This pressure sensor produces a signal whose voltage is proportional to the pressure acting across it. The pressure sensors for this lab have two ports and can thus measure the difference in pressure between fluids. For this experiment, one of the ports is exposed to air such that the pressure sensor will be measuring the pressure difference between the fluid in the cylinder and the air and the same height. We will assume that the pressure difference is due only to the fluid level and disregard the pressure difference caused by the column of air between the surface of the fluid and the pressure sensor.
- *DAQ Board:* As in previous labs, we will use the data acquisition board to convert analog voltages to data MATLAB can use. In this lab, however, we will use the board to read a voltage from the pressure sensor rather than across a resistor or LED. We will also use one of the DAQ board's analog output channels to power the sensor.

## 8.6 Description of Experiment

The pressure in a stationary fluid at a particular depth below the surface is given by adding the atmospheric pressure at the top surface to the hydrostatic pressure generated by the fluid itself. For an incompressible fluid (or one assumed to be incompressible), this latter portion may be calculated using the density of the fluid, the acceleration constant of gravity, and the depth below the surface. This means the total, or *absolute pressure*, is:

$$P(z) = P(0) + \rho g z$$

where $z$ is the depth below the surface, $P(0)$ is the atmospheric pressure at the top surface, $\rho$ is the density of the fluid, and $g$ is the local acceleration constant (the positive direction is down in this case). This equation indicates that the pressure increases linearly as the depth below the surface increases. The assumptions made for this equation (incompressible fluid, constant gravity, no motion) are *generally* acceptable for this lab.

The pressure sensor is designed to produce an output signal whose voltage is proportional to the pressure acting across the sensor. For this experiment, the pressure *difference* will be measured between the fluid at approximately the 100 mL mark in the cylinder and the air on the other side of the sensor. Because the density of air is so much smaller than the density of water, the atmospheric pressure at the surface of the fluid is very nearly the same as the atmospheric pressure at the open end of the pressure sensor. Given that, the pressure sensor measures predominantly the pressure difference caused by the fluid column $\rho g z$ where $z$ is the distance between the surface of the fluid and the center of the pressure sensor. This is known as the *Gage Pressure*, named for James Tiberius Gage.[2]

For these graduated cylinders, there is a geometric relationship between the volume in the cylinder and the height of fluid in the cylinder, meaning there is also a relationship between the voltage and the volume. In a "perfect" cylinder, the height would be linearly proportional to the volume and therefore the pressure would also be linearly proportional to the volume. Assuming the relationship between the pressure across the sensor and the voltage measured across the sensor is linear, this would yield a linear relationship between the height and the voltage as well. For these cylinders, however, the sides flare out as the volume increases, yielding a more complicated relationship. For this lab, then, a third-order fit will be used between the volume of the fluid and the voltage measured by the sensor.

In this lab, we will determine the characteristics that optimize measuring the pressure in the cylinder given the geometry of the experiment. We will then use MATLAB to find the mathematical relationship between the volume of the water in the cylinder and the voltage produced by the sensor. We will also find a relationship between the calculated pressure and the measured voltage to check the accuracy of this relationship against the published relationship given for the sensor. In each case, we will compare the results of two different calibrations - both having the same number of basis points but taken over different ranges - to determine if the range over which the sensor is calibrated leads to differing results.

## 8.7 Skeleton Files and Scripts

For this week, some of the scripts you need have already been written. You will be making modifications to them in order to acquire data from the pressure sensor, find a calibration curve, verify the calibration visually, and validate the calibration numerically. The files are at:

`http://www.duke.edu/~mrg/DAQS/DAQ3/`

and are called `AcquireXYY.m`, `CalibrateXYY.m`, `ReadVol000.m`, and `ValidateXYY.m`. Open a web browser on the PC (specifically, Internet Explorer), point to the URL above, and then right-click on the file you want. Choose `Save Target As...` and go to

`C:\Documents and Settings\mrglocal\My Documents\MATLAB`

[2] No, not really...Really it is named for the fact that the pressure is measured across a gage.

Do this for all the files. Recall that when MATLAB is running on a PC, it assumes that the working directory is a directory called `MATLAB` that was installed within the `My Documents` directory. Copies of these files are at the end of this handout so you can take notes regarding how they work.

Note that when you start to work on the files ending in `XYY`, you will change those characters to numbers based on your lab section (`X`) and workstation (`YY`). This way, every group will end up with a different file name. As an example, if you are working on station 22 during lab period 4, you will end all of your files with 422. Also, in the skeleton codes there will be several comments. Comments beginning with two % symbols mark the beginning of major sections of code, but you will not need to add any code directly below them. Comments beginning with a single % denote code that must be added (potentially more than one line of code).

## 8.8 Connecting the Equipment

The pressure sensor has 4 metallic pins which fit into the small white or clear connector. The connector is in turn attached to a bundle of 4 wires that run to the DAQ board. The wires should be connected to the following four pins on the pressure sensor:

- positive sensor power (white)
- positive sensor measurement (red)
- negative sensor power (black)
- negative sensor measurement (green)

You should connect the sensor power leads (white and black) to DACOUT0 (line 22) and AOGND (line 55) and the sensing leads (red and green) to ACH1 (line 33) and ACH9 (line 66), respectively. The sensor we are using requires a 10 V power signal to take the most precise pressure measurements. According to the published specifications of the sensor, every change in pressure of 1 pound per square inch (psi) will cause a voltage change of 0.01 V across the sensor when the power leads maintain a 10 V difference. The maximum pressure these particular sensors can read is 5 psi, so the maximum voltage they should produce is 0.05 V.

## 8.9 Optimizing the Experiment

An important consideration for this experiment is whether or not the 5 psi sensors you have been given are appropriate for measuring the pressure in these cylinders. That is to say, you must determine if the hydrostatic pressure in a full cylinder is such that a 5 psi gage would be able to measure it precisely without being saturated.

Assuming the specific weight of water is 62.4 lb/ft$^3$ (specific weight is $\rho$ times $g$), determine the height of fluid required to create a pressure change of 5 psi. You can use MATLAB as a calculator since you should not have a calculator on your desk. Write that value here:

$$z_{5\psi} =$$

Given that value, is this an appropriate pressure sensor? Would you have purchased/invented a different sensor for this particular application? The pressure sensors from this "family" are at:

`http://www.omega.com/pptst/PX26.html`

## 8.10 Acquiring Data

Since your calculations above should have at least determined that the 5 psi pressure sensors will be able to measure the full range of possible hydrostatic pressures in these cylinders, we can therefore take data to calibrate the sensors for these cylinders. Open the `AcquireXYY.m`, and save it as your own file, replacing X with your lab section number and YY with your station number as given by the sticker on the computer.

Stations 1-9 may put a 0 before their section number; for example, Station 1 for lab section 3 may write `Acquire301.m`. You will need to fill in the skeleton to perform the following steps. Many of these procedures are similar to the previous two labs, so you should use those handouts for reference.

- Initialize variables and objects: you should make sure that the `Volume` and `Voltage` variables are cleared and that any DAQ objects are deleted.
- Prepare output: the DAQ card itself will act as the power source for the pressure sensor. The sensor requires 10 V to operate, but does not require a very high current. Given that, you can simply send a 10 V signal to one of the output channels.

  Create a handle for the output object (call it `AO`). Add channel 0 to it, and use the `putsample` command to put 10 V to DAC0OUT.
- Prepare input: the DAQ card will be taking differential voltage readings from the pressure sensor. Technically, these DAQ boards could measure up to eight cylinders at once using differential voltages or sixteen at once using single-ended measurements. For this experiment, however, you will only be measuring one.

  Create a handle for the input object (call it `AI`). Add channel 1 to it.
- Prepare DAQ card: this step involves setting the input voltage range. In previous labs, we used the default values which took measurements from -5 V to +5 V. In this experiment, we *could* use that range, but to get as much precision as possible, it is better to set the card to measure voltages between -0.05 V and +0.05 V. There are three different card characteristics you need to set to change the range. The first is the `input` range, which determines the range of voltage values you expect to read at the card. The second and third are the `sensor` range and the `units` range. These two settings together allow you to teach MATLAB how to convert the input voltage into some other measurement. For this experiment, the range for input, sensor, and units will ALL be $\pm 0.05$ V since we do not yet have a calibration curve. Once calibrated, we could use the `SensorRange` and `UnitsRange` to have MATLAB report the measurements directly in pounds per square inch, but for now we want the voltages themselves. You should therefore add the following code:

```
set(AI.Channel(1), 'InputRange',  [-0.05 0.05]);
set(AI.Channel(1), 'SensorRange', [-0.05 0.05]);
set(AI.Channel(1), 'UnitsRange',  [-0.05 0.05]);
```

Once the preliminaries are done, you can begin writing the acquisition loop. This will be where DAQ actually retrieves the data and determines the voltages and volumes to be used later.

- Write a `while` loop that executes as follows:
  - Initialize Variables: for the loop, there will be a text variable that determines if the loop should run again. As long as the variable is not 'n' the loop will run. Therefore, setting the variable to 'y' will start the loop running.

    Given that this is the way the loop runs, there should also be a variable to keep track of how many data points have been acquired. For this program, the `Count` variable is being used. Note that this code is already in the skeleton - always check to see if the code is already written.
  - Next, increment the counter, prompt the user to input a volume and then store that volume in an array called `Volume`. Specifically, you should store it in the `Count` element of `Volume`. **It is a good idea to leave the semi-colon off this line, so you can see all the Volumes you have taken data for each step along the way.**
  - Prompt the user to hit return to take data by using the `pause` command. This part is already written.

  - Acquire twenty data points from the pressure transducer using the `getsample` command in a loop. You will want to pause 0.05 seconds between taking points, and you will want to store each of the measurements in an array. The reason for taking multiple points is so that you can average them to hopefully reduce any errors that may have crept into one of the data points. You will want to store each data point into a temporary array (say, `VoltTemp(k)` if your controlling variable is `k`).
  - After getting the information from the card, average the data from the transducer and store this average value in an array called `Voltage`, specifically the `Count` element of `Voltage`. The command for taking an average value of a vector is `mean`.
  - Ask the user if another measurement is required. For this loop, the program will run as long as the input is not 'n' so making a typographical error will cause the program to run an extra time. This is a much easier problem to correct than ending a run early.

- Saving data: This part of the code is written for you. It will ask the user for a filename (which should *not* include any kind of extension or dots) and will store your `Volume` and `Voltage` vectors into that file in MATLAB format. Be sure to give a name that makes it obvious what the data is for. For example, for your first run, you may want to save data to a file with a base name of `VoltVolData1`. MATLAB will automatically add the `.mat` to the end so long as you do not put a dot in your filename. Do not put a dot in your filename. I say again: *Do not put a dot in your filename.*[3]

- Cleaning up the DAQ: for this part, you need to make sure that the outputs are all set to zero and then that the DAQ objects are cleared out. This part is done for you.

Once your program is written, you will use it to take data from the graduated cylinder. You will actually be producing two calibration runs - one over a small range of volumes and one over a large range. The specific volumes to use are given in the "Acquired Voltages" table in the sample laboratory assignment on page Lab 8 – 10.

To set the fluid level, **carefully** raise or lower the fill bucket such that the siphon fills or drains fluid from the cylinder. Put a little extra into the cylinder, then set the bucket down. Allow the fluid to drain down until it is at precisely the volume you want. At that point you can crimp the tube. *Be careful crimping the tube* - do not pull the cylinder over! After you have taken your data point, repeat this process for the remaining points. Note that the data can be taken in any order. Be sure to write down the names of the two data files in the space given for future reference.

[3] Note: `filename` + • ⟶ ☹

## 8.11 Calibrating the Sensor

For this part, you are going to add code to an m-file called `CalibrateXYY.m` where again you should save the file under a new name where `X` is your lab section number and `YY` is your station number. This program will load the data file you created in the first part of the lab and use those values to calibrate the pressure sensor so that MATLAB will have an equation from which it can calculate the volume as a function of the voltage. The program will accomplish the following tasks:

- Load the data: this part is already written for you. The file name you give here should contain the `Voltage` and `Volume` vectors obtained using the `Acquire` program (for example:`VoltVolData1`).
- Enter your `XYY` values as a string. This will eventually find its way into the title of your figure.
- Create a third order fit: use the `polyfit`, `polyval`, and `linspace` commands however you need to in order to acquire the coefficients for a fit. Note that you should call the coefficients `P` since that is the variable name used further down in the program for plotting purposes. The lines of code labeling the plot are already done for you.

  Ideally, the graduated "cylinder" would be a cylinder and would have perfectly straight sides, leading to a linear relationship between the volume of fluid and the height of the fluid. Given that, there would be a linear relationship between the volume of fluid and the pressure difference between the sensor and the surface of the fluid. Given *that* and assuming the manufacturer is correct in saying there is a linear relationship between the pressure and the voltage, there would be a linear fit overall between the volume of fluid and the voltage across the sensor.

  Unfortunately, the graduated cylinders we are using are flared - they are larger at the top than the bottom so that they can more easily be taken off the mold on which they were formed. This leads to a more complicated relationship between the height (and thus the pressure, and thus the voltage) and the volume. To account for this, use a cubic fit between volume and voltage:

$$\text{Volume} = P(1)\ \text{Voltage}^3 + P(2)\ \text{Voltage}^2 + P(3)\ \text{Voltage} + P(4)$$

  Note in this equation that *Voltage* is the independent variable and *Volume* is the dependent variable - make sure the arguments of `polyfit` and `polyval` correct reflect that.

- Save your model coefficients and graph: this code is already included. Be sure to use a sensible name for each file - for example, for the coefficients calculated using the data from your first run, you may want to save data to a file with a base name of `VolumeModelCoefs1` and the graph to a file with a base name of `VolumeModelPlot1`. MATLAB will automatically add the `.mat` to the end of the data file and the `.eps` to the end of the plot file so long as you do not put a dot in your filenames. Do not put a dot in your filenames. Ahem. **Do not put a dot in your filenames.**[4] Write these filenames here for future reference.

You should make sure to have two different model coefficient files - one for the narrow-range data and one for the full-range data. Note that the program will automatically put the title of your volume / voltage file as well as your section / station number in the title of the graph.

As part of the lab assignment, you will compare the values of `P` for these two curves. In a "perfect" world, the coefficients would be exactly the same. Identify some possible sources of differences (if indeed there are differences) and what could be done to correct these.

[4] Also, be sure not to put a dot in your filenames.

## 8.12 Using Calibration Curves

Run the `ReadVol000.m` file in MATLAB by typing `ReadVol000` and then enter the base filename containing the appropriate model coefficients (for example, for the first set of data with volumes between 300-700 mL, you would enter `VolumeModelCoefs1`. This program will read the coefficients (the `P` values) from your file and will produce a continuously updated bar graph of the calculated volume in the cylinder. Note that this program expects the coefficients in `P` from your calibration file to have units of $mL/V^3$, $mL/V^2$, mL/V, and mL, in that order. First use `VolumeModelCoefs1` to see if volume values seem to be coming out correctly. Then use your second set of coefficients for the same purpose. By hand, fill in the "Volume Estimates" table given in the sample laboratory assignment on page Lab 8 – 10. You will get these values by changing the level of fluid to the appropriate point and reading the calculated volume off the screen. Do not worry if the volume values on the screen are changing - just write down an approximate average. Note that in some cases, the predicted volume may be *significantly* different from the set volume - you will need to explain this in your lab assignment. To make `ReadVol000` stop, hit CTRL-C.

## 8.13 Validating Results

As mentioned, the sensor is supposed to change 0.01 V for every extra 1 pound per square inch of pressure across the sensor. In order to see if the sensor is working properly, you need to determine whether this is true. You already have voltage measurements and volume measurements, but are lacking height measurements from which you can get the pressure values. The cylinders you are using have a larger diameter at the top (2.3125 in) than at the bottom (2.1680 in). This gives an equation for height in inches as a function of volume in milliliters of:

$$h(\mathrm{V}ol) = 14.8463 * (5179.79 + \mathrm{V}ol)^{(1/3)} - 256.87499$$

Admittedly, mixing SI and Imperial units in this way is generally frowned upon, but it does allow us to more easily compare the calibration results to published data.

The pressure difference measured at the sensor should be the pressure difference between the ambient air and the pressure on the fluid side of the sensor. This pressure difference can be given by the difference in the height between the top surface and the sensor times the density of the fluid times gravity (assuming we neglect the difference in air pressure from the surface of the fluid to the other end of the pressure sensor).

To get a pressure in psi from a height of fluid in inches, use the formula:

$$P(h) = \rho g(h(\text{fluid}) - h(\text{sensor})) = \frac{0.0366 \text{ lb}}{\text{in}^3}(h(\text{fluid}) - h(\text{sensor}))$$

Load the script called `ValidateXYY.m` then save it as a new file where again `X` and `YY` have been replaced by your lab section number and workstation number, respectively. Change the value of the `XYY` variable in the code to represent your lab section and workstation. You may also need to change the `SensorVol` variable value to reflect where your sensor actually is - look at the graduated cylinder and estimate the volume reading at the center of the fill tube. This is most likely 100 mL, but if not, change the value for this in the code as well. This should be all you need to change. The rest of the program uses the equations above to create and graph a linear fit of voltage as a function of pressure. The script also uses `polyfit` to determine a first order fit between pressure and voltage:

$$\text{Voltage} = a(1)\ \text{Pressure} + a(2)$$

Run this program on both data files (e.g. `VoltVolData1` for the first set) to determine if $a(1)$ is close to the proscribed value of 0.01 V / psi for each calibration and if $a(2)$ is close to 0. Be sure to run the program on both calibration curves and save them to files that make sense to you - for example, for the coefficients calculated using the data from your first run, you may want to save data to a file with a base name of `VoltageModelCoefs1` and the graph to a file with a base name of `VoltageModelPlot1`. MATLAB will automatically add the `.mat` to the end of the data file and the `.eps` to the end of the plot file so long as you do not put a dot in your filenames. Do not put a dot in your filenames. **Do not put a dot in your filenames.** Write those file names in the margin of this page for future reference.

## 8.14 Lab Assignment

For the lab assignment, each person in the group will be uploading all your work to your OIT directory. Also, you and your partner(s) will be turning in a single **group lab document** (blank copies will be available in the lab) consisting of the two tables as well as answers to some questions about your data and lab equipment. The tables to be included are the data tables given in the sample assignment on page Lab 8 – 10. You will be writing down the two sets of two equations each obtained with the `CalibrationXYY` and `ValidationXYY` codes - you must also have a TA verify your four equations at some point during the lab (for the Volume as a function of Voltage equations, only use two significant figures; for Voltage as a function of Pressure, use five). The questions to be answered are written in the sample assignment.

## 8.15 Uploading Files

While grading, the TAs will go into the directories of the people in each group to verify the graph and also that files have been transferred over. Make sure you have copied everything into the correct directory. Go to the Start Button, then All Programs, then SSH Secure Shell, then Secure File Transfer Client. Log into your favorite OIT machine. The left side of the screen will be the file structure for the PC you are on and the right side will be the file structure of your OIT account. Note that on the PC, your files are stored in

```
C:\Documents and Settings\mrglocal\My Documents\MATLAB
```

On the OIT account, go into your EGR53 directory and then create a new folder called `daq3`. You can then drag and drop all your scripts (`.m`), data (`.mat`), and graphics (`.eps`) files into your `daq3` folder. You do *not* need to copy any other files, such as `.asv` files. **Make sure** that *each person in the group* has all the required data and script files for this lab. *Each person* will need to upload:

- Your group's `AcquireXYY.m` and `CalibrateXYY.m` files
- Two data files containing Voltages and Volumes (the ones made in the `AcquireXYY` script during Section 8.10)
- Two data files containing coefficients for Volume as a function of Volume (the ones made in the `CalibrateXYY` script during Section 8.11)
- Two data files containing coefficients for Voltage as a function of Pressure (the ones made in the `ValidateXYY` script during Section 8.13)
- Two plots of Volume as a function of Voltage with both data points and a cubic fit line (should be made in the `CalibrateXYY` script during Section 8.11) - one for each data set.
- Two plots of Voltage as a function of Pressure with both data points and a linear fit line (should be made in the `ValidateXYY` script during Section 8.13) - one for each data set.

## 8.16 Cleaning Up

Once you are sure your files have been copied over, delete all the files in MATLAB's `Current Directory` (either by selecting them all in the `Current Directory` window in MATLAB and deleting them or by going to the directory listed above and deleting the files). Next, empty the recycling bin on the desktop. Leave the wiring connected.

When you think your lab station is cleaned up and the files are cleared off, ask a TA to verify that, the contents of your `MATLAB` directory, and the contents of the recycling bin - this will serve as attendance for the lab and you cannot receive a grade higher than 0 for this work without getting checked off for having properly dumped your files.

For this week, leave MATLAB up and running and leave the Recycling Bin folder open on the screen. Do *not* log off the PC.

## 8.17 Sample Laboratory Assignment

### Names and NET IDs

### Acquired Voltages

| Calibration Data 1 | | | | | |
|---|---|---|---|---|---|
| Volume / Voltage Data File Name: | | | | | |
| Volume (mL) | 300 | 400 | 500 | 600 | 700 |
| Voltage (V) | | | | | |

| Calibration Data 2 | | | | | |
|---|---|---|---|---|---|
| Volume / Voltage Data File Name: | | | | | |
| Volume (mL) | 460 | 480 | 500 | 520 | 540 |
| Voltage (V) | | | | | |

### Volume Estimates

| Coefficient Set 1 | | | | | | | |
|---|---|---|---|---|---|---|---|
| Volume as f(Voltage) Coefficient File Name: | | | | | | | |
| Act. Vol. (mL) | 300 | 450 | 500 | 550 | 700 | 800 | 900 |
| Calc. Vol. (mL) | | | | | | | |

| Coefficient Set 2 | | | | | | | |
|---|---|---|---|---|---|---|---|
| Volume as f(Voltage) Coefficient File Name: | | | | | | | |
| Act. Vol. (mL) | 300 | 450 | 500 | 550 | 700 | 800 | 900 |
| Calc. Vol. (mL) | | | | | | | |

### Equations

| Equation | Calibration 1 | Calibration 2 |
|---|---|---|
| Volume as f(Voltage) | | |
| Voltage as f(Pressure) | | |

## Questions to Answer

**(1)** How tall is the tallest column of fluid that can be measured by a 5 psi pressure sensor? Might there be a better sensor for this experiment?

**(2)** What might be the cause of differences between the two equations for volume as a function of voltage, and what conclusions can you draw from these discrepancies about calibrating a sensor?

**(3)** Given the volume estimates determined above from the `ReadVol000`, what might be the cause of differences the calibration curves have in reading the same volume, and what conclusions can you draw from these discrepancies about calibrating a sensor?

**(4)** For each of your two Voltage vs. Pressure equations, how close is your pressure sensor calibration to the 0.01 V/psi listed in the specifications?

**(5)** Describe when you would want to use the narrow-range calibration and when you would want to use the long-range calibration. Support your conclusion with evidence generated during the experiment.

**(6)** What did you like best/least about this lab? What changes should be made to improve the lab?

## 8.18 Scripts

### 8.18.1 AcquireXYY.m Code

```
%% Initialize variables and objects  (done)
clear; format short e
delete(daqfind);

%% Prepare Output
% Create handle to output (to do)

% Add channel 0 to output (to do)

% Put 10V to sample (to do)

%% Prepare Input
% Create handle to input (to do)

% Add channel 1 to input (to do)

%% Prepare DAQ Card
% set input, sensor, and units range (to do)

%% Acquisition loop
% Initialize Loop and Count Variables (done)
Again = 'y';
Count = 0;
% Start loop (done)
while ~strcmp(Again, 'n')
    % Increment counter (done)
    Count=Count+1;
    % Get volume from user, store in an array, and pause (done)
    Volume(Count)=input('Enter volume: ')
    fprintf('Hit return to take data\n');
    pause;
    fprintf('Taking data\n')
    % Get 20 datapoints from transducer / store in VoltTemp (loop) (to do)

    % Find average value of VoltTemp data and store data in array (to do)
    Voltage(Count) =
    % Ask user if there are more inputs (done)
    Again = input('More inputs? (y/n)', 's')
end

%% Saving data (done)
FileName = input('Enter filename for Volume and Voltage data (eg VoltVol1): ', 's');
FileSave = sprintf('save %s Volume Voltage', FileName);
eval(FileSave)

%% Cleaning up DAQ (done)
putsample(AO, 0)
clear AI AO
delete(daqfind)
```

### 8.18.2 `CalibrateXYY.m` Code

```
% Clear Variables and Initialize Workspace (done)
clear; format short e

% Load data (done)
FileName = input('Enter filename for Volume and Voltage data (eg VoltVol1): ', 's');
FileLoad = sprintf('load %s', FileName);
eval(FileLoad)

% Enter string for section and station. (to do)
% For example, section 3 station 8 is '308'
XYY = ;

% Create third order fit of Volume as a function of Voltage (to do)
P =

% Create model line with 100 points (done)
VoltageModel = linspace(min(Voltage), max(Voltage));
VolumeModel = polyval(P, VoltageModel);

% Plot data and model on the same graph
% Plot Volume as a function of Voltage
% Use blue circles for data points and
% a black line for your model (to do)
plot()

% Change the axes to go from 0 to 110% of maximum values (done)
axis([0 1.1*max(Voltage) 0 1.1*max(Volume)])

% Label and title (done)
xlabel('Voltage (V)');
ylabel('Volume (mL)');
title(sprintf('Volume vs. Voltage for PX26-005DV from %s (%s)', FileName, XYY))

% save data (done)
FileName = input('Enter filename for saving V=f(v) 3rd order fit coefficients (eg Vol1Coefs): ',
FileSave = sprintf('save %s P', FileName);
eval(FileSave)

% save your graph (done)
FileName = input('Enter filename for saving V=f(v) graph (eg Vol1Plot): ', 's');
PlotSave = sprintf('print -deps %s', FileName);
eval(PlotSave)
```

### 8.18.3 `ReadVol000.m` Code

```
clear; format short e; figure(1); clf

% load data
FileName = input('Enter filename for 3rd order fit coefficients (eg Vol1Coefs): ', 's');
FileLoad = sprintf('load %s', FileName);
eval(FileLoad)

%% Prepare Output
% Create handle to output
ao = analogoutput('nidaq',1);
% Add channel 0
addchannel(ao, 0);
% Put 10V to sample
putsample(ao, 10);

%% Prepare input
% Create handle to input
AI = analoginput('nidaq',1);
% Add channel 1
addchannel(AI, 1);
% Set Input, Sensor, and Units range to [-0.05, 0.05]
set(AI.Channel(1),'InputRange', [-0.05, 0.05]);
set(AI.Channel(1),'SensorRange',[-0.05, 0.05]);
set(AI.Channel(1),'UnitsRange', [-0.05, 0.05]);

%% Plot
figure(1);
hold off;
QPlot = bar(0)
axis([.5 1.5 0 1000]);
ylabel('Volume (mL)');
fprintf('Time (s) Voltage (mV)  Volume (mL)\n');
tic
while 1,
    % Get data from transducer
    for k=1:1000
        VolArray(k) = getsample(AI);
    end
    % Find average value of data
    AvgVoltage = mean(VolArray);
    % Use coefficients to determine volume
    Volume = polyval(P, AvgVoltage);
    % Update bar plot
    QPlot = bar(Volume);
    axis([.5 1.5 0 1000]);
    fprintf('%8.2f %12.4f %12.2f\n', toc, AvgVoltage*1000, Volume);
    drawnow
end
```

### 8.18.4 `ValidateXYY.m` Code

```
% Clear Variables and Initialize Workspace (done)
clear; format short e; figure(1); clf

% load data (done)
FileName = input('Enter filename for loading volume/voltage data: ', 's');
FileLoad = sprintf('load %s', FileName);
eval(FileLoad)

% Enter string for section and station. (to do)
% For example, section 3 station 8 is '308'
XYY = '308';

% Set sensor "volume" in mL (to do)
% only line you need to change - most are correctly at 100 mL
SensorVol = 100;

%% NO MORE CHANGES NEEDED %%
% Determine height as a function of volume
% height in inches from volume in mL
h=14.8463*(5179.79+Volume).^(1/3)-256.87499

% Determine height of sensor from volume reading
hs = 14.8463*(5179.79+SensorVol).^(1/3)-256.87499

% Determine pressure difference measured
Pressure = 0.0366 * (h-hs);

% Create first order fit of Voltage as a function of Pressure
a = polyfit(Pressure, Voltage, 1)

% Create model line with 100 points
PressureModel = linspace(min(Pressure), max(Pressure), 100);
VoltageModel = polyval(a, PressureModel);

% Plot data and model on the same graph
% Use blue circles for data points and a black line for your model
plot(Pressure, Voltage, 'bo', PressureModel, VoltageModel, 'k-');
% Change the axes to go from 0 to 110% of maximum values
axis([0 1.1*max(Pressure) 0 1.1*max(Voltage)])

% Label and title your graph
xlabel('Pressure (psi)');
ylabel('Voltage (V)');
title(sprintf('Voltage vs. Pressure for PX26-005DV from %s (%s)', FileName, XYY))

% Save Pressure-Voltage model coefficients and graph
FileName = input('Enter filename for saving v=f(P) coefficients for fit: ', 's');
FileSave = sprintf('save %s a', FileName);
eval(FileSave)
FileName = input('Enter filename for saving v=f(P) calibration graph: ', 's');
PlotSave = sprintf('print -deps %s', FileName);
eval(PlotSave)
```