UM-SJTU Joint Institute

Vm495 Laboratory II

Lab1: Strain Gauge and Thermocouple Measurements

Group 2E

Jiang Rong 5133709250

Mao Tianyu 5133709015

Zhang Congfei 5133709241

Zhang Xiu 5133709041

2017/06/26

Table of Content

Abstract

Soft drinks are usually injected by carbon dioxide to make their special taste. This report researches on the Coke Cola cans to find a way discovering the internal pressure change with the temperature without destroy the can itself while using Wong Lo Kat cans with no carbon dioxide as a compare experiment. This experiment uses two strain gauges to detect the strains of the cans caused by internal pressure change, Wheatstone bridge circuits to amplify the strain gauge output voltage transferring to DAQ and thermocouple to measure the wall temperature of cans. By processing these experimental data, we calculate the internal pressures under different temperature. The result shows that different cans have different inner pressures given same temperature, and the increase of temperature causes the increase of internal pressure.

Nomenclature

A = *gain of the amplifier*

D = *diameter of the can*

E = *Young's modulus*

L = *length of the can*

P = *force inside the can*

p = *pressure*

R = *resistance*

S = *strain gauge factor*

t = *thickness of the can's wall*

V = *voltage*

ε = *strain on the surface of a can*

σ = *stress on the surface of a can*

ν = *Poisson's ratio*

I. Objectives

THE objective of this experiment is to measure the change of the internal pressure of beverage cans with respect to the change of environment temperature. We design a mechanical system via strain gauge, thermocouple, DAQ and bridge circuit to indirectly measure the internal pressure without opening the cans.

II. Introduction

Soft drinks are made with carbon dioxide injected at a low temperature while kept at room temperature. Manufacturers usually want to know the exact internal pressure after soft drinks injecting by carbon dioxide to ensure the storage safety of the products. The amount of the liquid and carbon dioxide and the initial pressure setting would refer to the change of internal pressure. [1]

Each type of soft drinks has different volume and shapes with a different amount of carbon dioxide and different internal pressure. We use the cans of famous soft drink Coke Cola as the target.

We design a mechanical system to measure the internal pressure of soft drink cans. We fix the strain gauge on the thin surface of the can to measure the micro-strain change with respect to internal pressure. Theoretical knowledge indicates that as temperature rises, pressure in the can should also rise. We use thermocouple to measure can

temperature, DAQ module and bridge circuit to collect the digital signal from strain gauge. Then we can calculate the internal pressure of the target cans. [1]

III. Background

It is hard to directly measure the pressure inside a can, so we transform the pressure inside the can to the stress on the surface of the can. To do so, we can construct a thin-walled cylindrical vessel module because the thickness of the can is much smaller than the diameter of the can. In addition, we assume that the material of the can, aluminum alloy, is a kind of isotropic material. Using Hooke's law, we can find out the relationship between stress and strain and then use strain gauge to measure the change of strain. Next, using Wheatstone bridge, we can get the voltage change in the circuit based on change of strain. Finally, with the help of amplification circuit, the voltage signal is amplified and transmitted to DAQ. For the temperature on the surface of the can, we directly use thermocouple to measure it. The methodology of the whole process is shown in Figure 3.1.

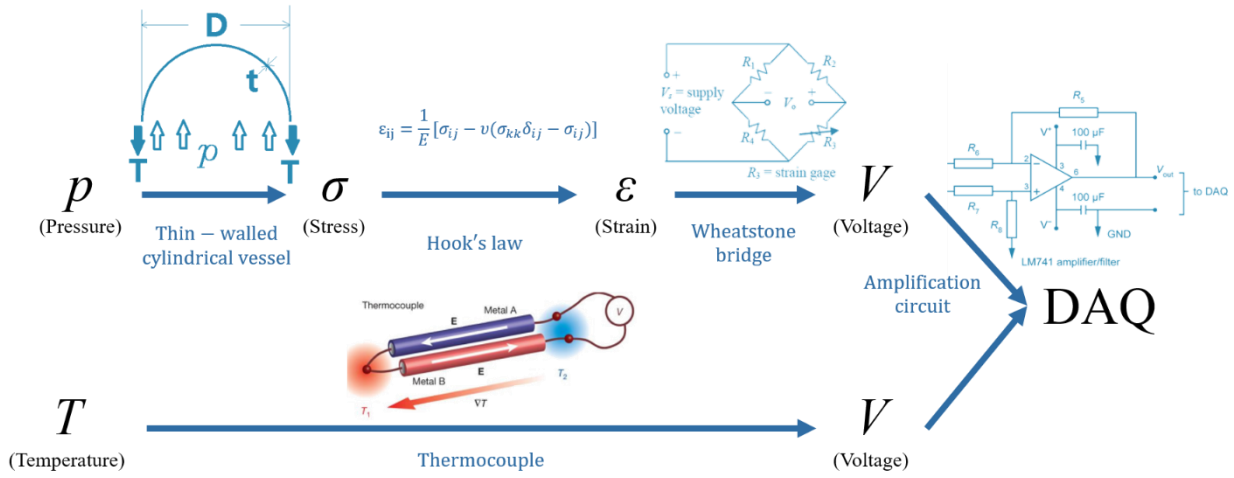


Figure 3.1. The methodology of the experimental process. [1]

A. Pressure Measurement

Figure 3.1 shows the free body diagram of the half-cross-section of a thin-walled cylindrical vessel, which can be considered as half-cross-section of the can to be tested.

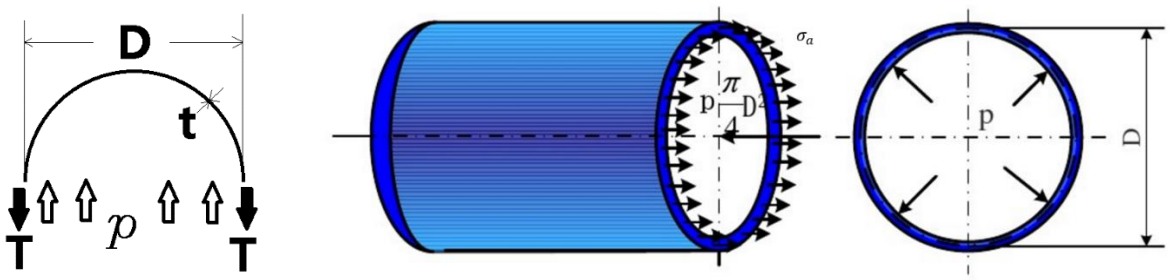


Figure 3.2. The free body diagram of half-cross-section of the thin-walled cylindrical vessel. [1-2]

As shown in Figure 3.2, the pressure inside of the can, the tension and the thickness of the wall and the diameter of the can are *p*, *T*, *t* and *D* respectively. Assume the length of the can is *L*, we can get $P = DLp$, where *P* is the transvestite external force on the half-cross-section. The transverse force inside $P' = 2Lt\sigma_t$, where σ_t is the transverse stress across the section. Considering the force balance in transverse direction on the surface of the can, we can get that $DLp = 2Lt\sigma_t$ or $\sigma_t = \frac{pD}{2t}$. Similarly, the force balance in axial direction on the surface of the can is $\frac{\pi}{4}D^2p = \pi Dt\sigma_a$ which means $\sigma_a = \frac{pD}{4t}$, so

$$p = \frac{4\sigma_a t}{D} = \frac{2\sigma_t t}{D} \quad . \tag{1}$$

We assume the can is isotropic, because it is made of aluminum or steel. According to Hooke's Law, we can get $\varepsilon_{ii} = \frac{1}{E}[\sigma_{ii} - \upsilon(\sigma_{jj} + \sigma_{kk})]$ where *E* is the Young's modulus of the material and *i*, *j* and *k* represent three dimensions. Considering the assumption that the can is a thin-walled cylindrical vessel, the radial strain of the wall in the last equation can be neglected and we can get the axial and transverse strain from Equation (2)

$$\begin{cases} \varepsilon_{\mathrm{a}} = \frac{1}{E}(\sigma_a - \upsilon\sigma_t) \\ \varepsilon_{\mathrm{t}} = \frac{1}{E}(\sigma_t - \upsilon\sigma_a) \end{cases} \tag{2}$$

where ε_a is the axial strain, ε_t is the transverse strain and υ is Poisson's ratio.

B. Analog Signal Amplification

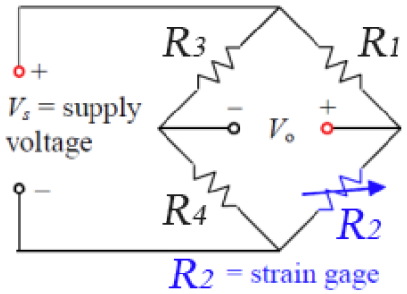


Figure 3.3. Quarter bridge circuit. [1]

A strain gauge is a small piece collects small wires. We stick it to the surface of the can to measure the strain of the can. When the wires are stretched or compressed, the resistance of it will change, based on the change of resistance, we can calculate the strain. With Wheatstone bridge circuit (quarter bridge circuit), we can get the change of the resistance of the strain gauge. Figure 3.3 shows a quarter bridge circuit where R_3 is a strain gauge, from which we can get $V_o = V_s \frac{R_2R_3 - R_1R_4}{(R_1+R_2)(R_3+R_4)}$ where V_o is the output voltage and V_s is the supply voltage. Assume that $R_{2,initial}R_3 - R_4R_1 = 0$ .When the resistance of the strain gauge changes δR_3 which is small, then $V_o = V_s \frac{(R_{2,initial}+\delta R_2)R_3 - R_1R_4}{(R_1+R_{2,initial}+\delta R_2)(R_3+R_4)}$. From the previous two equations, we can get that $V_o = V_s \frac{\delta R_2 R_3}{(R_1+R_{2,initial}+\delta R_2)(R_3+R_4)}$. Considering that $\delta R_2/R_{2,initial}$ is much less than 1 and $\delta R_2 = R_{2,initial} S\varepsilon$ we can get $V_o = V_s \frac{\delta R_2 R_3}{(R_1+R_{2,initial})(R_3+R_4)}$. So

$$\varepsilon \approx \frac{V_o}{V_s}\frac{1}{S}\frac{(R_1+R_{2,initial})^2}{R_1R_{2,initial}} \tag{3}$$

where S is strain gauge factor. If R_1 equals to $R_{2,initial}$, we can get that $\varepsilon_a \approx 4\frac{V_o}{V_s}\frac{1}{S}$. However, in real experiment, R_1 did not equal to $R_{2,\,initial}$, so we prefer to use Equation (3) than this equation.

For a strain gauge, the strain gauge factor S is defined as the constant ratio of the percentage changes in resistance to strain, of which the relationship is shown in Equation (4).

$$S = \frac{1}{\varepsilon}\frac{\delta R}{R}. \tag{4}$$

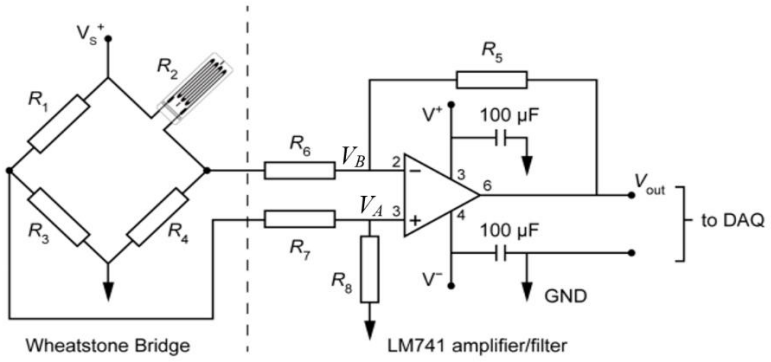


Figure 3.4. Quarter bridge circuit with a LM741 amplifier.[1]

Figure 3.4 shows a quarter bridge circuit with a LM741 amplifier. LM741 amplifier is kind of common amplifier which can be used to increase the amplitude of an input signal. In this lab, to amplify the signal, a LM741 amplifier with a difference amplifier circuit is added to the circuit. Applying KCL we can get that

$$V_{out} = V_A \frac{R_8(R_5+R_6)}{R_6(R_7+R_8)} - V_B \frac{R_5}{R_6} \text{ where } V_o = V_A - V_B. \quad (5)$$

Combining Equation (1), (2), (3), (4) and (5) we can get the internal pressure p of a beverage can

$$p = \frac{4tE}{DSV_S(1-\upsilon^2)} \left[\frac{(R_{1,axial}+R_{2,initial,axial})^2}{R_{1,axial}R_{2,initial,axial}} \frac{\delta V_1}{A_{axial}} + \frac{(R_{1,transverse}+R_{2,initial,transverse})^2}{R_{1,transverse}R_{2,initial,transverse}} \frac{\upsilon \delta V_2}{A_{transverse}} \right) + p_{atm} \quad (6)$$

where t is the wall thickness of can, E is Young's modulus, A_{axial} and $A_{transverse}$ are gain of amplifier, D the is diameter of can, S is strain gauge factor, υ is Poisson's ratio, V_s is input voltage, δV_1 is the change of axial output signal, δV_2 is the change of axial output signal and p_{atm} is the atmospheric pressure. The physics meaning of R_1 and $R_{2,initial}$ in both axial and transverse directions can be found in Figure 4.

C. Temperature Measurement

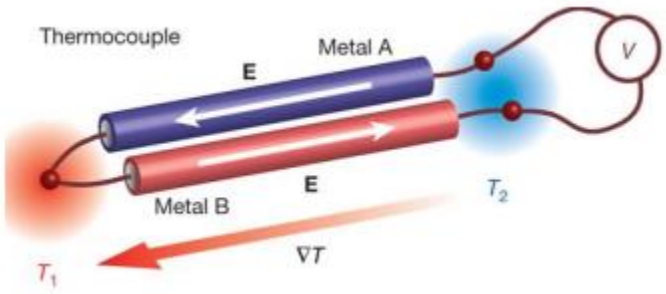


Figure 3.5. The concept diagram of a thermocouple. [1]

Finally, we use a K-Type thermocouple to measure the surface temperature change of the can during the heating process. As shown in Figure 3.5, the theory of the thermocouple is connecting two different metal wires at different temperatures forming a thermojunction. Then heat and electrical current transferred through the wires and the thermojunction voltage is generated. According to Thompson effect, the voltage changes with the temperature of the thermojunction. Also according to Seebeck effect, when a conductor is subjected to a temperature gradient, it generates a voltage. So if we know the materials of the metal wires and a reference temperature, then we can use the output voltage to measure the temperature of the can.[1]

IV. Experimental Method

A. Equipment & Sensor

Table 4.1. The table of equipment and sensors used in the experiment.[1]

Name	Quantity	Usage
Aluminum alloy Coca-Cola can	4	Carbonated can
Aluminum alloy Wong Lo Kat can	1	Non-carbonated can
K-Type thermocouple	1 per can	Measuring temperature
Strain gauge	2 per can	Measuring strain
DAQ	1	Acquiring data
Heater	1	Heating and stirring
Circuit board	1	Measuring voltage and amplifying signal
Holder	2	Acrylic holder to hold can
Beaker	1	Containing can and water
Computer	1	LabVIEW
Power supply	1	Supplying power
Thermometer	1	measure the temperature of water

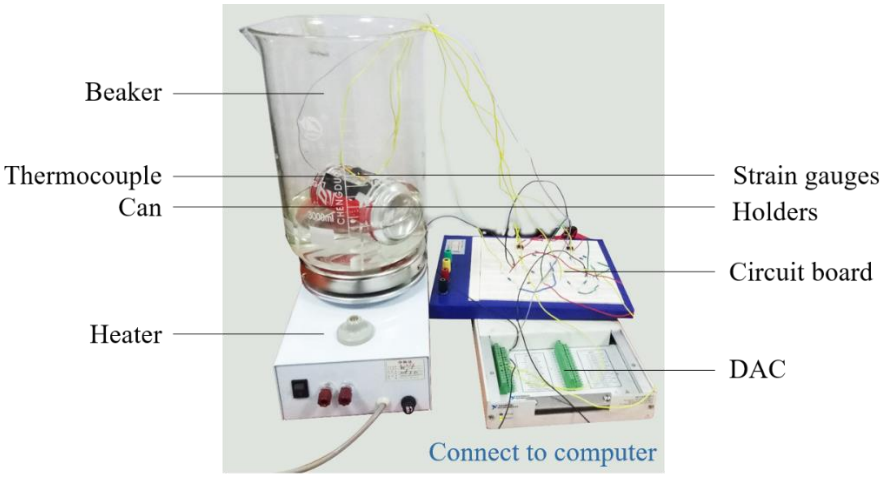


Figure 4.1. Experimental setup.

B. Test Procedure

1. Weld thermocouple

Cut a 50-60cm thermocouple and remove the insulating layers of the ends of the two wires. Next, weld two wires together at one end to form a thermojunction.

2. Polish cans

Use abrasive paper to polish the surface of the cans and then clean the polished surface.

3. Stick and weld strain gauges

As shown in Figure 4.2, axially and transversely stick two strain gauges on the polished surface of the can respectively. Then, weld strain gauges to wires.

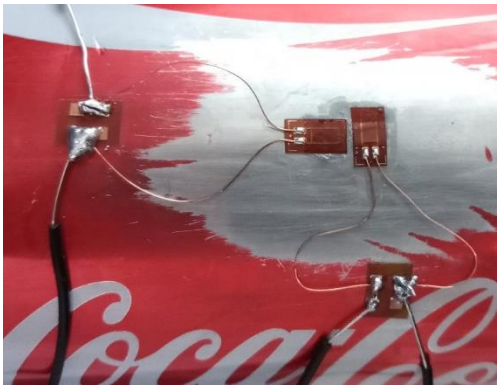


Figure 4.2. Stick and weld strain gauges on the polished surface of the can.

4. Stick thermocouple

Stick a layer of tape on the polished surface to prevent contact of the thermocouple and the can. Then put the thermocouple on the tape. Next, stick another layer of tape on the thermocouple to fix it.

5. Connect circuit

As shown in Figure 4.3, build two quarter bridge circuits with amplification circuits to transfer and amplify voltage signal.

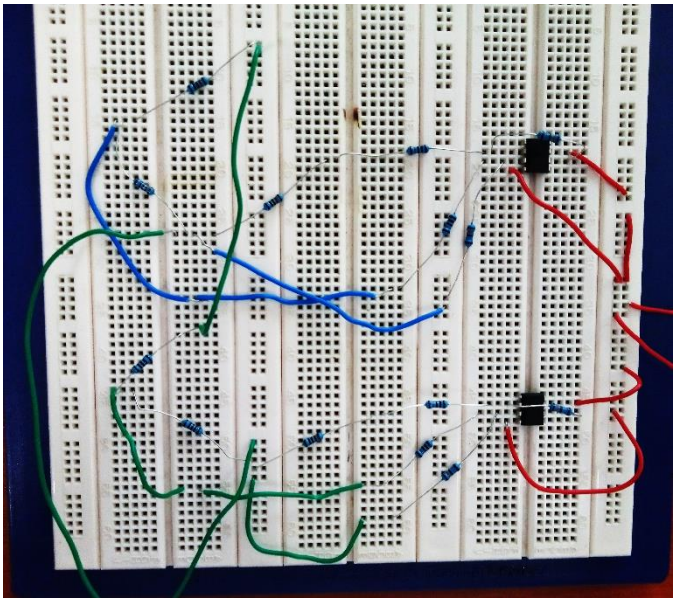


Figure 4.3. Two quarter bridge circuits with amplification circuits.

6. DAQ connection and LabVIEW programing

Create a new program in LabVIEW and then add three data acquiring outputs for the two strain gauges and the thermocouple. Next, connect the circuit output to DAQ and adjust LabVIEW program so that we can collect 50 set of data in one second.

7. Build holders

Use laser cutting machine to cut two holders whose inner diameter is 66.4mm which is a little greater than the diameter of cans. The CAD diagram of the holders is shown in Figure 4.4.

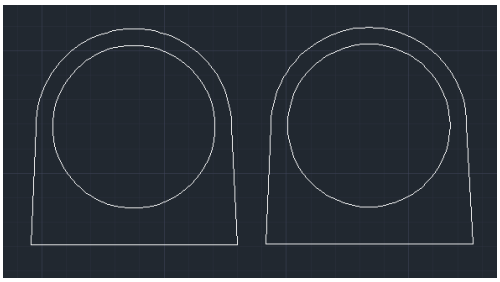


Figure 4.4. CAD diagram of two holders.

8. Construct experimental setup

As shown in Figure 6, put equipment together and fill about 1000ml water so that it can submerge about half of the can.

9. Do experiment

Heat and stir the water by heater and record data by LabVIEW. Heat up the water to 55℃ and then close the heater for a while. Then, take out and open the can. Pour the liquid inside the can. Measure the signal when the temperature of the can reaches to room temperature. Repeat this step for all the cans.

10. Measurement

Measure and record all the resistances used in the circuit. Also, measure the diameter and wall thickness of the cans by Vernier caliper and micrometer caliper. The result is shown in Appendix.

C. Calibration Method

Thanks to the inbuilt calibration function inside the LabVIEW software, the thermocouple can be calibrated automatically. As a result, we do not need to calibrate the result again.

For strain gauges, we cannot directly use a load to calibrate them because they are stick to the surface of the cans. However, after we open the can and pour out the liquid inside the can, we can regard the pressure of the can to be the same with atmospheric pressure (about 101kPa). As such, we are able to calibrate the voltage output of the strain gauge.

V. Results and Data Analysis

In this part, we first show you the results of the measurements and then the analysis process of raw data to final results, and discussion about these results. In order to start analyzing the inner pressure of target can, we first make an assumption:

For identical cans, they have the same inner pressure under the same temperature. (5.1)

A. Result

Our objective is to measure the inner pressure of cans. In this experiments, we totally did 5 trials on measuring the inner pressure of cans. Among the trials, 1,2,3,5 trials are done based on identical coke-cola cans(carbonated), and trial 4 is done based on Wang loo can(non-carbonated).

1. Results of Carbonated Cans

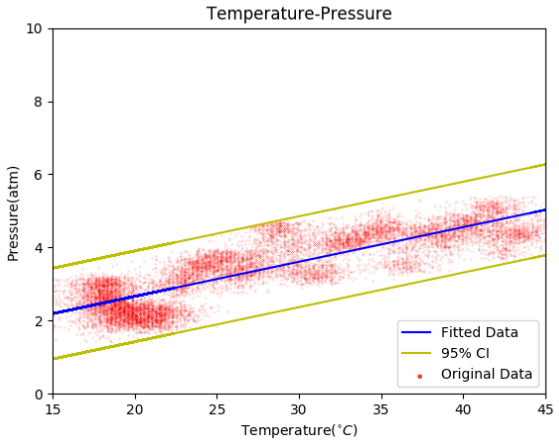


Figure 5.1 Trial 1

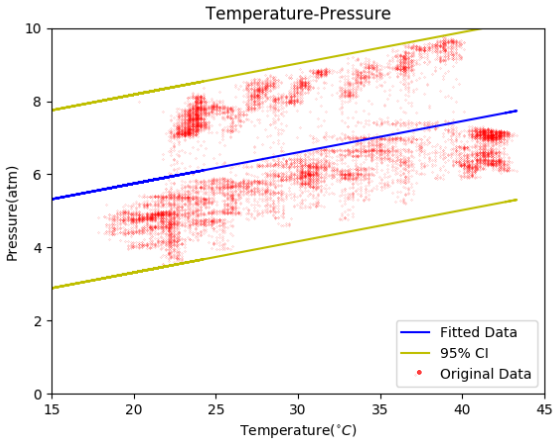


Figure 5.2 Trial 2

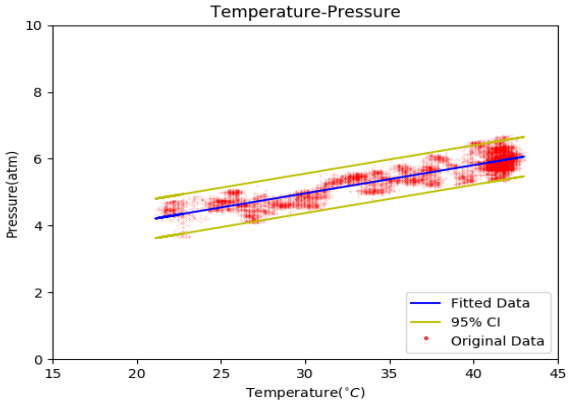


Figure 5.3 Trial 3

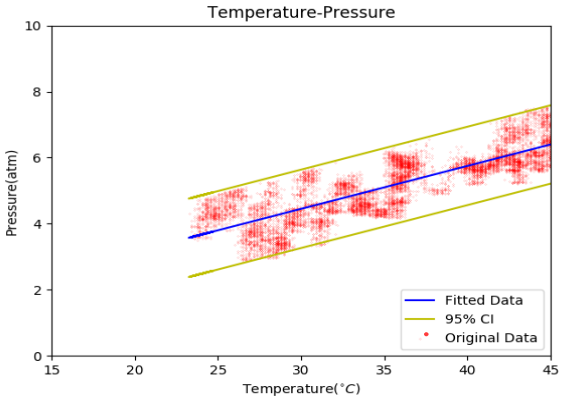
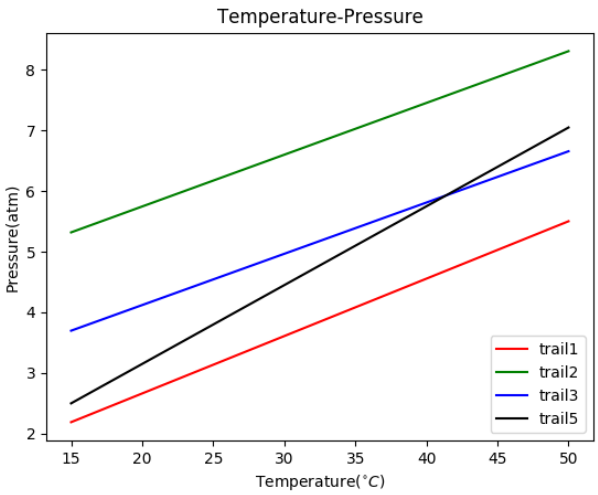


Figure 5.4 Trial 5

In the graphs shown, the red scatter points are the processed original data, the blue line is the linear fitted data based on original data. The area between two brown line indicates the *95% Confidence Interval* with 2 *standard error of estimate*. This *95% CI* is calculated as: *95%CI=fitted data*±1.96* *standard error of estimate*.

Then we summarize the 4 trials into one graph to compare them with each other:



Parameters of each trail:

1st trial: Regression: 0.094 *x+ 0.770

Linear correlation coefficient:0.780

2nd trial: Regression: 0.085 *x+ 4.038

Linear correlation coefficient:0.392

3rd trial: Regression: 0.084 *x+ 2.426

Linear correlation coefficient:0.875

5th trial: Regression: 0.129 *x+ 0.550

Linear correlation coefficient:0.789

Figure 5.5 Summary of carbonated cans

Each line in the summary has 2 parameters, one is the equation of linear regression, denoted as Regression,

The second parameter is the Linear Correlation Coefficient of the line. This parameter reflects the linearity of the original data, the bigger the coefficient is, the better linearity the original data has.

For simple linear regression, coefficient of determinant is simply the square of linear correlation coefficient. Thus we have table 5.1. We also show the standard error of estimate of each trial in this table.

Table 5.1 Parameters of trial 1,2,3,5

Trial number	Coefficient of Determinant	Standard Error of Estimate(atm)
1	0.608	0.633
2	0.154	1.241
3	0.766	0.301
5	0.623	0.605

As mentioned in the lectures, "Typical good fit for engineering data: 0.8 – 0.9"[3]. Thus none of the trials can be said with a good linearity.

From the summary we can see there exist large uncertainty between each analysis. Thus we can't get an explicit result for the experiment based on our initial assumption (5.1). We will further talk about the uncertainty in discussion part.

2. Results of Non-Carbonated Can

We did trial 4 based on non-carbonated can to compare the results of non-carbonated can with the carbonated can. As shown before, there exists large uncertainty in different trials, thus we take trail5 to compare with trial4. The reason is that we did trial 4,5 in the same lab section. With little knowledge of what caused such big uncertainties, we believe this would be the method to minimize the noise of experiment between trial 4, and 5.

Trial 4 also has a large in group *standard error of estimate* of 1.967 *atm*, which is an extremely high value of error. The assumption, pressure change with temperature of non-carbonated can should be less than the pressure change with temperature of carbonated cans. However, since this is a trial with such large uncertainty, it may fail to reflect the truth correctly.

Table 5.2 Parameters of trial 4

Trial number	Coefficient of Determinant	Linear Regression Equation	Standard Error of Estimate(atm)
4	0.608	0.175 *x+ -1.977	1.967
5	0.623	0.129 *x+ 0.550	0.605

B. Data Analysis Procedure

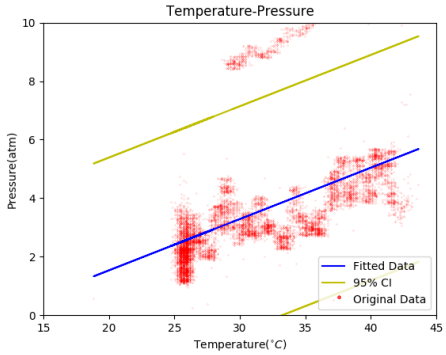


Figure 5.6 Trial 4

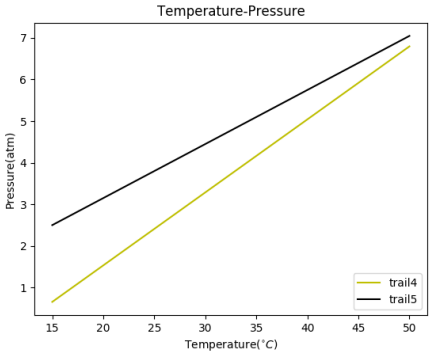


Figure 5.7 Comparison of Carbonated and non-carbonated can

In this part, we show how we analysis the raw data and how we convert raw data to final results.

As derived before, equation (6) indicates how raw data is calculated into pressure.

1. Based the equation, plot the raw data as pressure with respect to temperature on the graph.
2. Do the first try of linear regression.
3. Plot the standard residual of the pressure based on the equation:

$$\frac{e_i}{S_{y,x}} = (P_{predict} - P_{raw})/(S_{y,x})$$

Note: $S_{y,x}$ is the standard error of estimation

4. Based on the standard residual, exclude the outliers with standard residual smaller than -2 or greater than +2.
5. After outliers are excluded, do the linear regression again.

Take trial 3 as an example (data of other trials will be attached in appendix).

Figure 5.8 shows the original fit, figure 5.9 shows the fit after outliers are excluded. Figure 5.10 shows the Standard residual plot of original fit. Based on this process, we can improve the precision of the linear regression and exclude some of the noise.

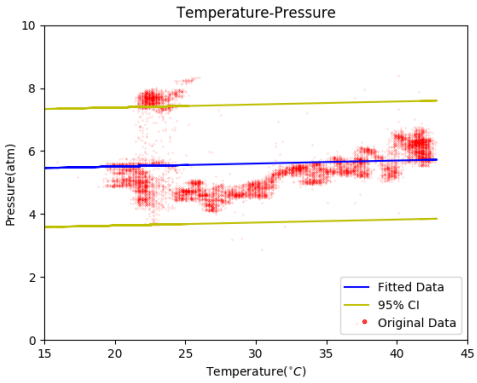


Figure 5.8 Fit of raw data

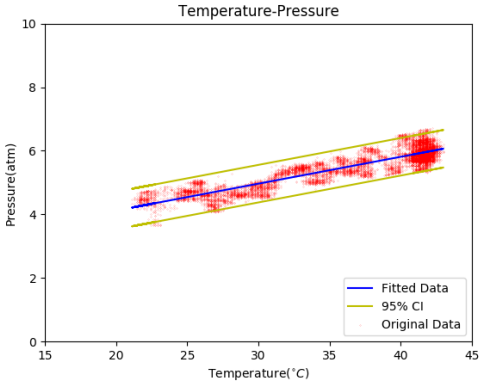


Figure 5.9 Fit of data excluded outliers

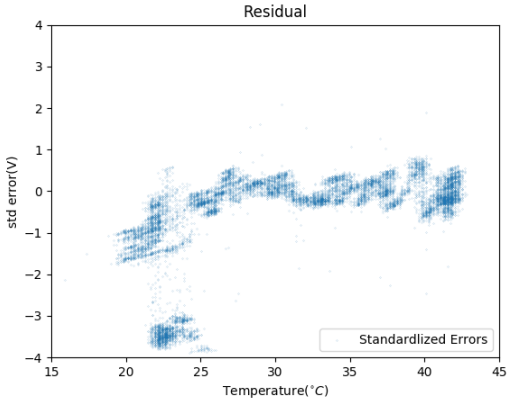


Figure 5.10 Standard residual of original fit of trial 3

C. Uncertainty Analysis and Discussion

Total absolute uncertainty u is composed of systematic error b and random error p.

For each value of temperature (we take 0.1 as resolution of temperature),

By equation (6), we can derive the random error with respect to temperature in the measurement.

$$b^2 = \sqrt{\sum \frac{\partial p}{\partial x_i} |u_{x_i}}$$

Assume here we take p=0, since we have no method to measure system error, then we have the equation

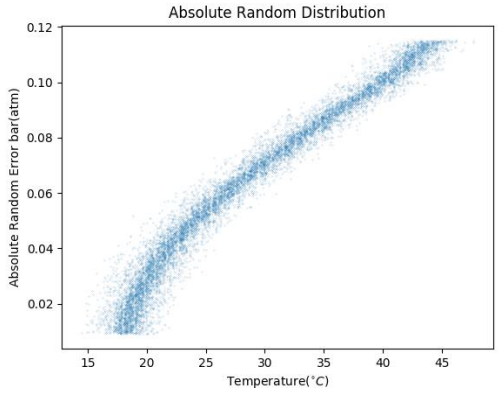
$$u = \sqrt{p^2 + b^2}$$

to calculate total uncertainty with the two errors.

Relative uncertainty $u\% = \frac{u}{p}$

By these equations we can plot the total uncertainty and relative uncertainty for each trial with respect to temperature, the uncertainty of each variable is shown in appendix table A1~A4

Take trial 1 as an example:



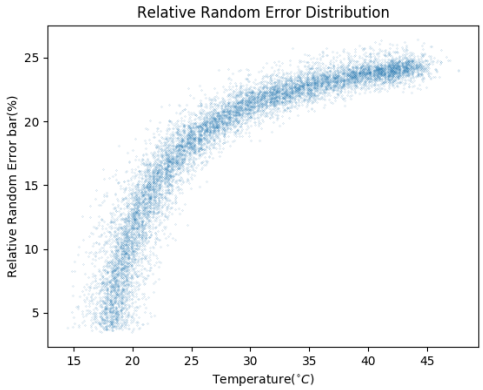
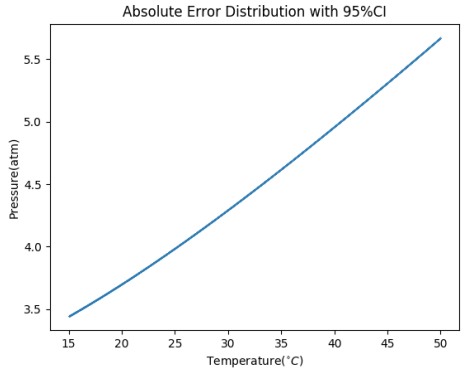


Figure 5.11 Theoretical Absolute Random Error of Trial 1 **Figure 5.12 Theoretical Relative Random Error of Trial 1**

This is theoretical uncertainty calculated.

Since we have done 4 trials, we can get the real uncertainty with 95% CI, take trial 1 as example:

The error distribution in figure 5.13 indicates the radius of 95% CI. Denote the radius as *r*, we obtain *r* by the following



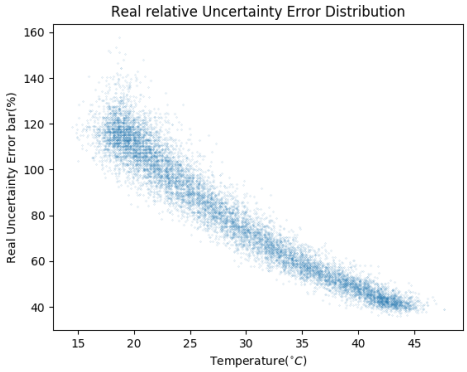


Figure 5.13 Real Absolute Random Error of Trial 1 **Figure 5.14 Real Relative Random Error of Trial 1**

equation:

$$r^2 = \sigma^2(P_1, P_2, P_3, P_5)$$

For each temperature. It is obvious that the real uncertainty is far larger the theoretical.

D. Why is that?

In order to find out the reason, we look further into the raw data to find the possible factors of uncertainty we have not counted for in the previous analysis.

First of all, we assume that the thermocouple is calibrated by LabVIEW. But when we plot out the Temperature-Voltage graph, we can see that although we start and end the experiment at the same temperature every trial (with

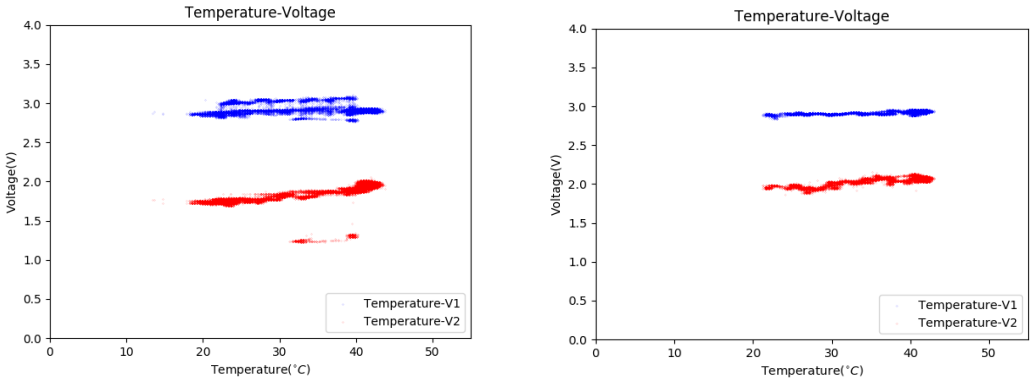


Figure 5.14 Comparison of Temperature – Voltage graph of trial 2&3

reference to thermal meter), temperature of different trials starts and ends at different value in the recorded data. Thus the assumption of thermocouple calibration failed.

Secondly, one of our team mates reported that he observed some wave noise during the data collection process.

In order to justify this observation, we did Fast Fourier Transform to find the unusually frequencies. If there is no such frequency, we can expect most of the frequency is in the low frequency domain.

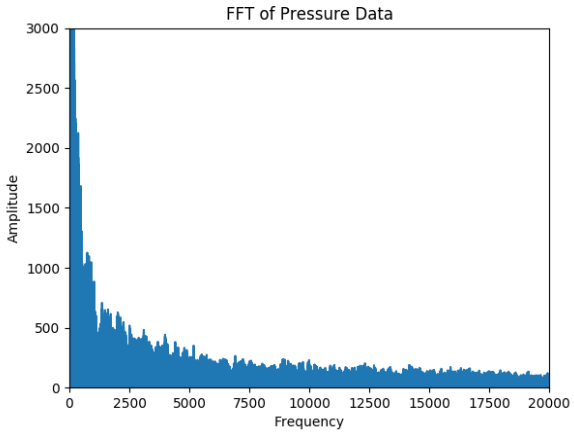


Figure 5.15 FFT of trial 1 Pressure-Temperature analysis

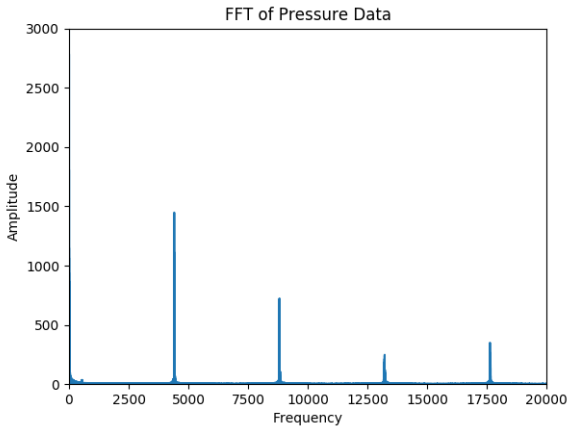


Figure 5.16 FFT of trial 1 Pressure-Temperature analysis

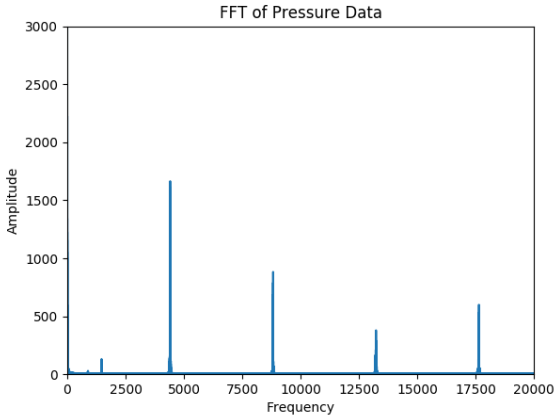


Figure 5.17 FFT of trial 2 Pressure-Temperature analysis

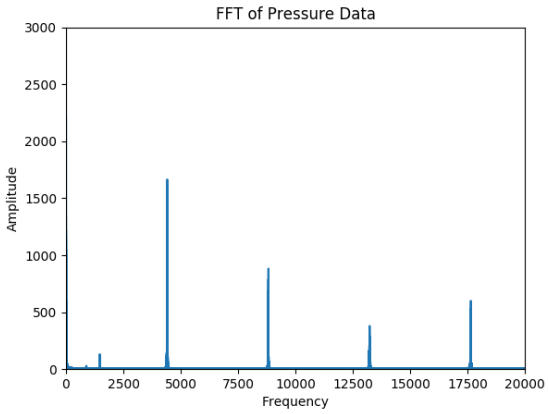


Figure 5.18 FFT of trial 3 Pressure-Temperature analysis

From figure 5.15,16,17,18. Frequency in the units of *Hz*. Surprisingly, we can observe obvious frequency interferes in the raw data of trial 1,2,3,5.

Also, for trials 2,3,5, the interferes happened at the same 5 frequencies, we have reasons to believe that this is a kind of system error that we do not detect before.

Till now, we have analyzed possible resources of noise. We think although the uncertainty is large, but not that large to provide a relative error rate of over 100%.

E. What are the other possible sources of errors?

One is the uncontrollable human factor, the strain gauge can be easily deformed by human force, and the scale of the deformation may be much higher than what can be caused by temperature change.

F. What if the initial assumption is not correct?

Another thought is that, what if, our initial assumption, all the cans have the same inner pressure at the initial temperature, is correct. This is worth thinking, because we cannot tell whether all the cans have the same property inside it, even if they are identical cans.

To test the assumption, we make another assumption:

Random error is exactly equal to the Theoretical error we have calculated before (Figure 5.11, 5.12). (5.2)

Then, at t=20° C, the pressure of can will from a Gaussian Distribution $X \sim N(3.92\ \text{atm}, 9\text{e}^{-4}\ \text{atm}^2)$. The possibility of obtaining pressure of each trial that is extremer than the trials we have (P=[2.3,2.6,3.8,5.3]) is almost p=[0,0,0,0]. Thus we can reject the assumption that the inner pressure of can has a mean of 3.92 atm.

Or alternatively, we can make another assumption with respect to assumption (5.1):

Inner pressure of cans can be different given same temperature. (5.3)

It is reasonable to make this assumption because when we are preparing the samples, we cannot guarantee it has never experienced any collision or other accidents that may cause state of gas dissolved in liquid to change.

Results of the experiment seems to fit this assumption (5.3) better. However, in this case, we still have a large in group measurement uncertainty (table 5.1,5.2, *Standard Error of Estimate*).

Then we have reasons to believe that the initial pressure inside the cans are different, and their pressure behaves similar with temperature change.

VI. Conclusion

This is an experiment with large uncertainty. Our original objective of this experiment is to measure the inner pressure of a can. In order to measure the pressure without breaking the can, we change the temperature of the can and measure the change of strain on the surface of the can with respect to temperature, and by analyzing the strain, we can reach to the inner pressure of the can. We finally reached to the result that or the pressure in the cans are not the

same, but follow similar behavior of pressure-temperature change. If we still have chance to do further experiments on this topic, we should design further experiments to exam the effect of human factors, calibration of thermocouple, inner frequency disturbance of the system, and prepare cans with exactly the same inner conditions to minimize the possible interference.

References

[1]Chen, C.P. “Lecture 2”, Vm495 Mechanical Engineering Laboratory II Lecture Slides. Mechanical Engineering Apartment, SJTU-Joint Institute, Shanghai, 2017.

[2]Li, X. "Stress Analysis of a Thin-Walled Cylindrical Vessel." Lecture Slides. Beijing Institute of Technology, 2014.

[3]Chen, C.P. “Lecture 6”, Vm495 Mechanical Engineering Laboratory II Lecture Slides. Mechanical Engineering Apartment, SJTU-Joint Institute, Shanghai, 2017.

Appendix

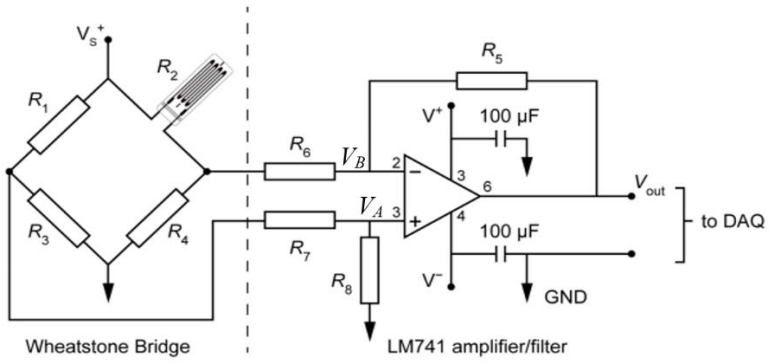


Figure A1. Quarter bridge circuit with a LM741 amplifier.[1]

Table A1. Values and systematic uncertainties of the resistances and amplifiers.

Category	Name	Value	Systematic uncertainty
Circuit for axial strain gauge	R_1 *	119.0Ω	0.05Ω
	$R_{2,\ initial}$ (strain gauge)	**	0.05Ω
	R_3	119.9Ω	0.05Ω
	R_4	119.1Ω	0.05Ω
	R_5	177.3kΩ	50Ω
	R_6	1.776kΩ	0.5Ω
	R_7	1.765kΩ	0.5Ω
	R_8	175.4kΩ	50Ω
	A (gain of amplifier)	98.7	
Circuit for transverse strain gauge	R_1*	119.9Ω	0.05Ω
	$R_{2,\ initial}$ (strain gauge)	**	0.05Ω
	R_3	119.4Ω	0.05Ω
	R_4	118.8Ω	0.05Ω
	R_5	177.2kΩ	50Ω
	R_6	1.778kΩ	0.5Ω
	R_7	1.788kΩ	0.5Ω
	R_8	176.6kΩ	50Ω
	A (gain of amplifier)	98.9	
Input voltage	V_s	5V	27.5mV

* R_1 to R_8 is shown in Figure A1.

** The results of R_2 for each trail are shown in Table A2.

Table A2. Values and systematic uncertainties of the initial resistances ($R_{2,\ initial}$) for the strain gauges in each trail.

Trail	$R_{2,initial}$ for Axial strain gauge	$R_{2,initial}$ for transverse strain gauge	Systematic uncertainty
1	120.9Ω	120.4Ω	0.05Ω
2	120.5Ω	120.2Ω	0.05Ω
3	120.7Ω	121.0Ω	0.05Ω
4	120.8Ω	120.7Ω	0.05Ω
5	121.1Ω	121.1Ω	0.05Ω

Table A3. Values and systematic uncertainties of the diameter and wall thickness of the can for each trail.

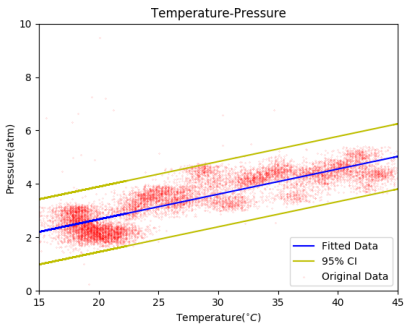
Trail	*Name*	Value	Systematic uncertainty
1	Diameter	66.1 mm	0.05mm
	Wall thickness	0.10 mm	0.01mm
2	Diameter	66.2 mm	0.05mm
	Wall thickness	0.10 mm	0.01mm
3	Diameter	66.1 mm	0.05mm
	Wall thickness	0.11 mm	0.01mm
4 (Wong Lo Kat can)	Diameter	66.1 mm	0.05mm
	Wall thickness	0.10 mm	0.01mm
5	Diameter	66.3 mm	0.05mm
	Wall thickness	0.10 mm	0.01mm

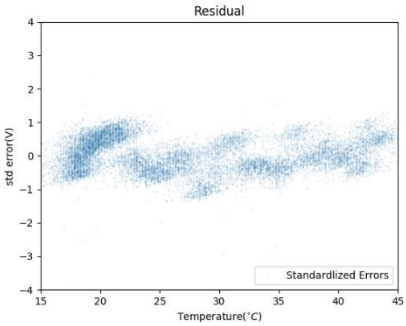
Table A4. Values of constants used in calculation.

Name	Value
Young's modulus (E)	69 kN/mm^2
Poisson's ratio (υ)	0.35
Strain gauge factor (S)	2 ($\pm$ 1.0%)
Atmospheric pressure	101.2kPa

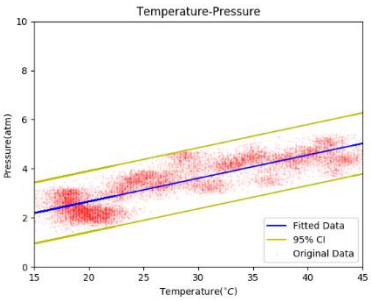
Figures A5. Data for trial 1

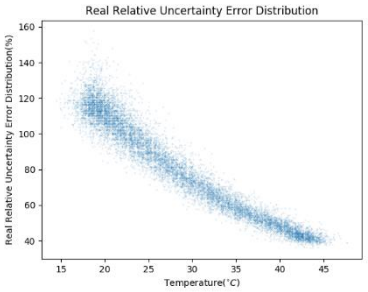
With outliers:

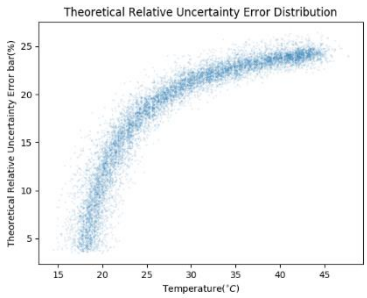


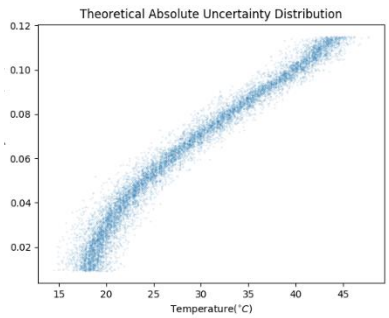


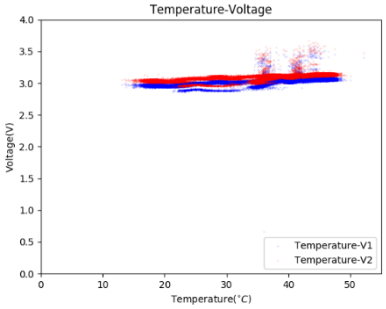
Without outliers:

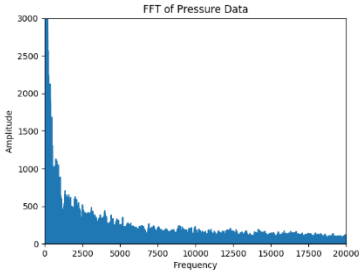






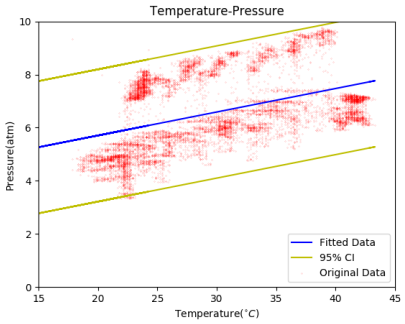


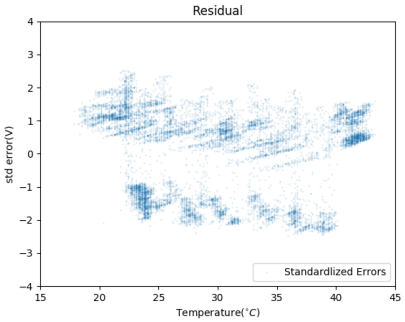




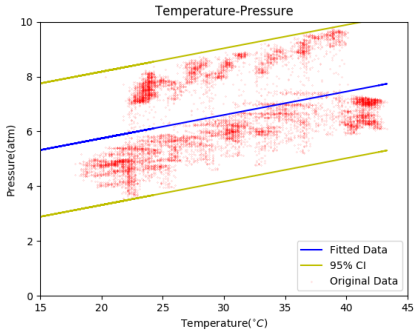
Figures A6. Data for trial 2

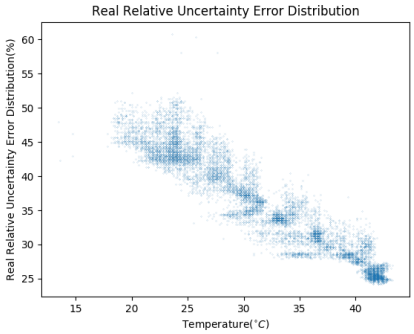
With outliers:

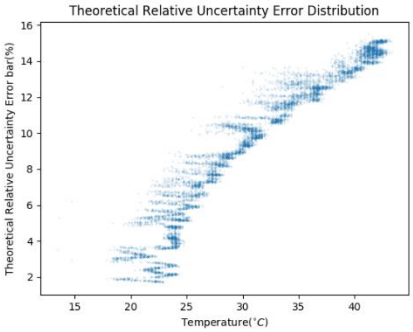


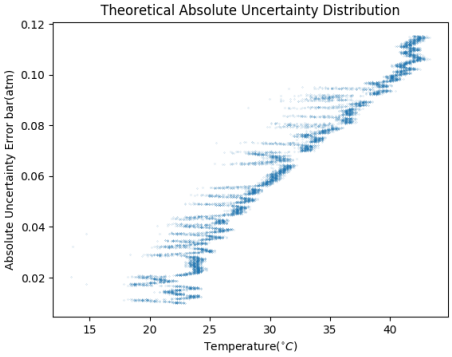


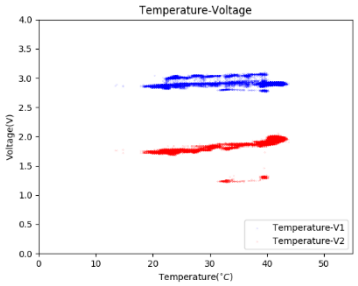
Without outliers:

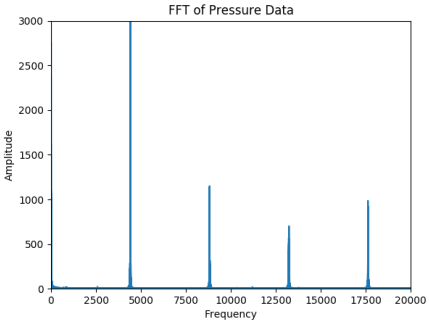






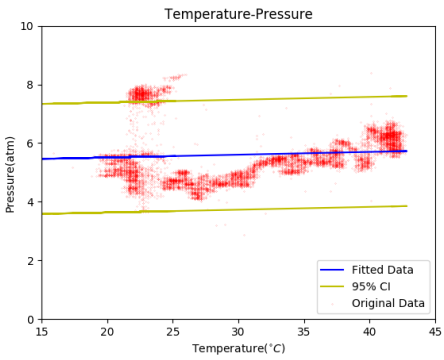


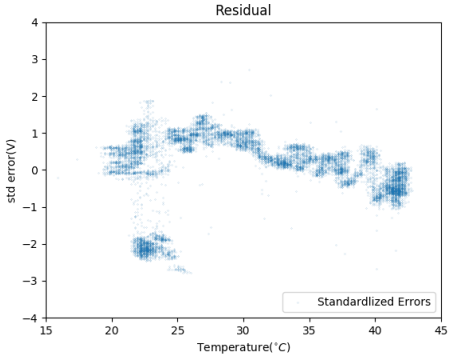




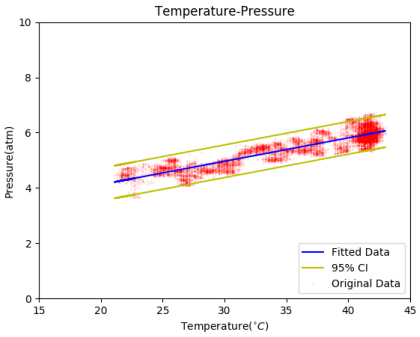
Figures A7. Data for trial 3

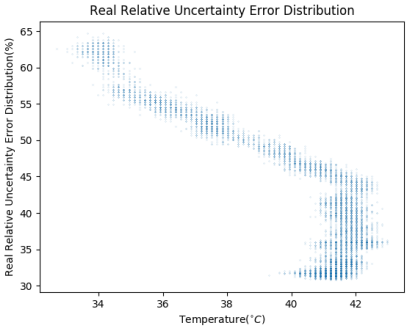
With outliers:

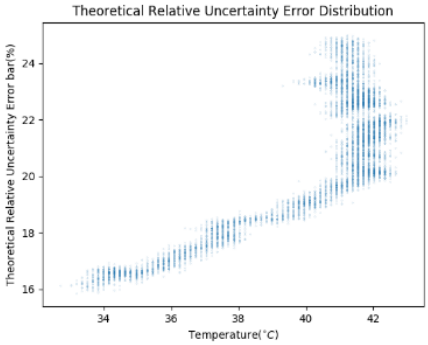


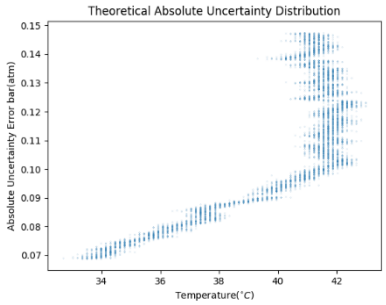


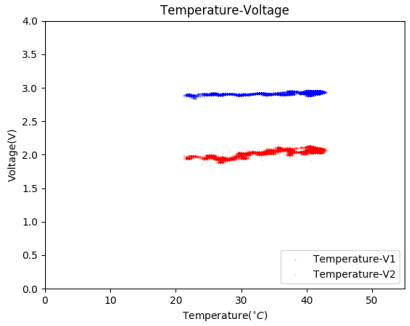
Without outliers:

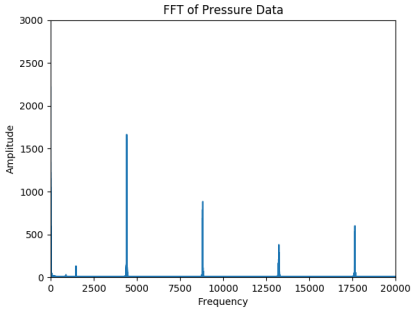






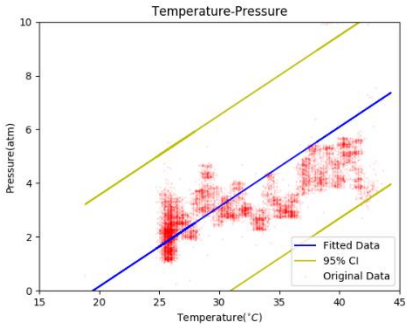


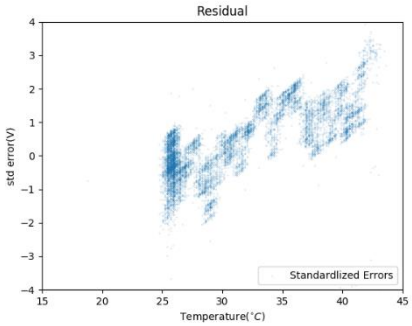




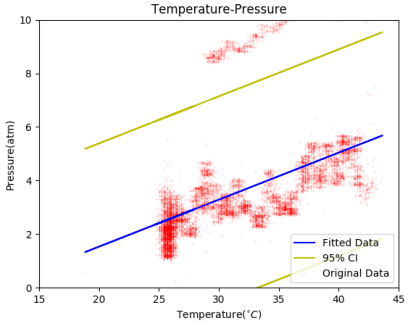
Figures A8. Data for trial 4

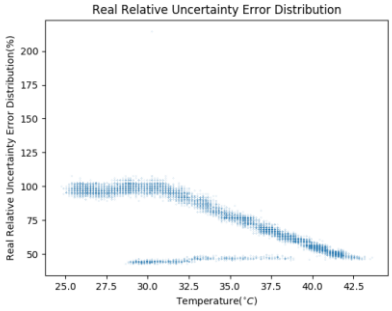
With outliers:

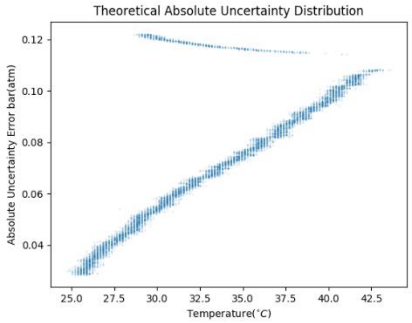


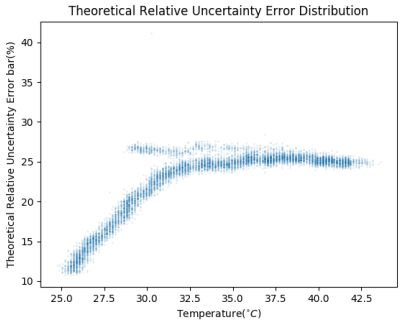


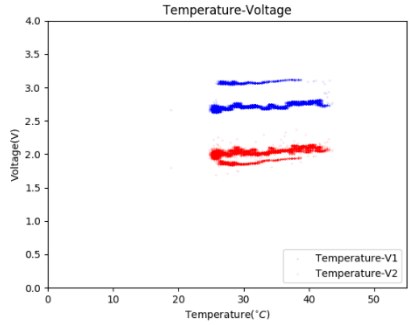
Without outliers:

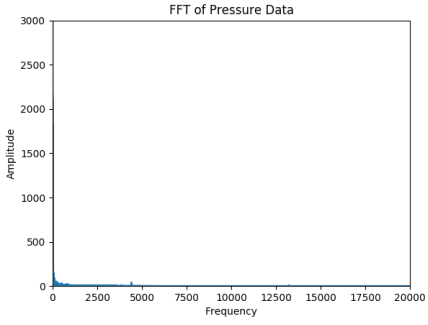






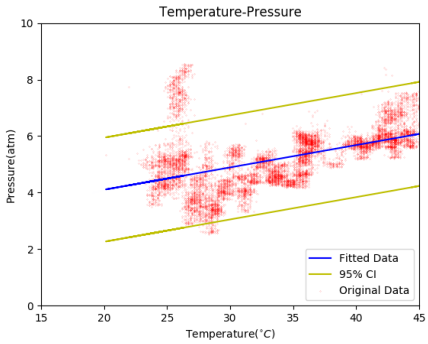


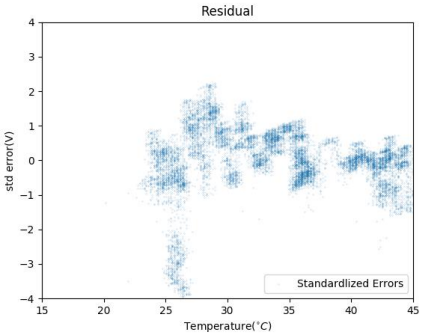




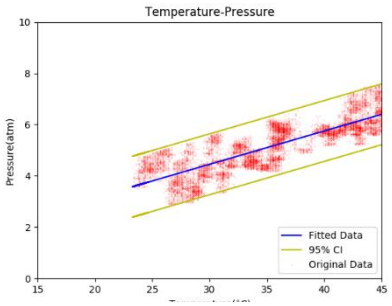
Figures A9. Data for trial 5

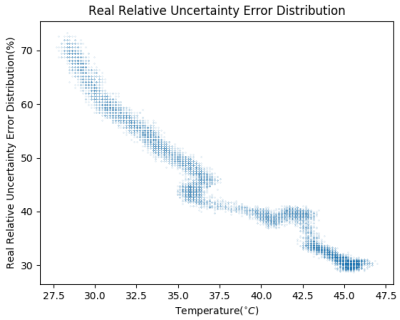
With outliers:

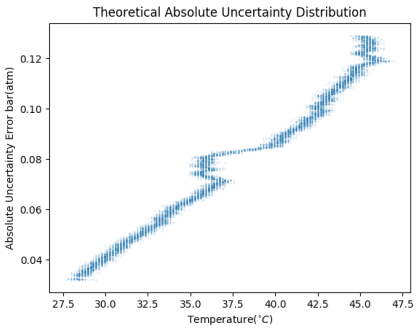


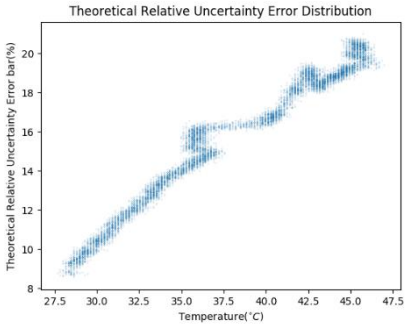


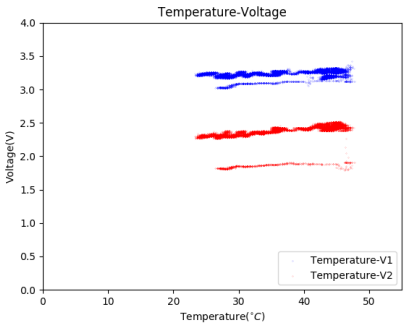
Without outliers:

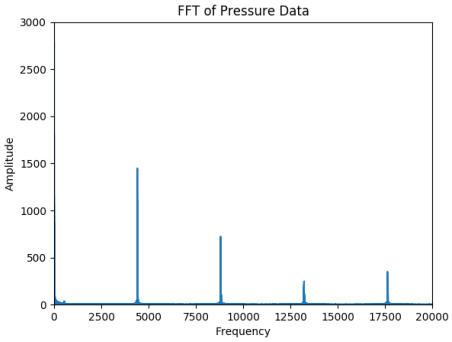












Appendix A9 Python Codes for Data Processing

```
import numpy as np
import matplotlib.pyplot as plt
from scipy.stats.stats import pearsonr
from sklearn.metrics import mean_squared_error
from sympy import *
import scipy.stats
def find_std(x):
    #x = np.linspace(15, 50, 500)
    #E=np.empty((5,x.shape[0]))
    #ERR=np.empty(x.shape[0])
    E=np.empty(5)# pressure std
    E[0] = 0.0946351550329 * x + 0.770132957906
    E[1] = 0.0853694646877 * x + 4.03822156582
    E[2] = 0.0845234089432 * x + 2.42978154566
    E[3] = 0.175470966994 * x + -1.97749171132
    E[4]= 0.129940121323 * x + 0.550779010841
    return np.std(E)
class data_plot:
    def __init__(self,a):
        '''
        t = 1e-4
        r = 33.2e-3
        S = 2
        E = 69e10
        v = 0.35
        R2 = 119
        R4 = 119
        Vs = 5
        A = 100
        '''
        self.a=a
        self.const = 1.93e1
        if a == 1:
            self.d = np.load('a.npy')
            self.refV1 = self.d[-500, 2]
            self.refV2 = self.d[-500, 3]
        elif a == 2:
            self.d = np.load('a2.npy')
            self.refV1 = self.d[-500, 2]
            self.refV2 = self.d[-500, 3]
        elif a == 3:
            self.d = np.load('a3.npy')
            self.refV1 = self.d[-500, 2]
            self.refV2 = self.d[-500, 3]
        elif a == 4:
            self.d = np.load('a4.npy')
            self.refV1 = self.d[-500, 2]
            self.refV2 = self.d[-500, 3]
        elif a == 5:
            self.d = np.load('a5.npy')
            self.refV1 = self.d[-500, 2]
            self.refV2 = self.d[-500, 3]
        self.e = np.empty([int(self.d.shape[0] / 50), 5], dtype="float64")
        self.totallen=int(self.d.shape[0] / 50)-3
        for i in range(self.totallen ):
            self.e[i, 0] = self.d[i * 50, 1]  # temperature
            self.e[i, 1] = self.const * ((self.d[i * 50, 2] - self.refV1) + 0.35 * (self.d[i * 50, 3] - self.refV2))  # pressure
            self.e[i, 2] = self.d[i * 50, 0]  # time
            self.e[i, 3] = self.d[i * 50, 2]  # v1
            self.e[i, 4] = self.d[i * 50, 3]  # v2
    def find_frequency(self):
        self.freq=np.fft.fft(self.d[:,2])
        self.freq_axis = np.linspace(0, 44100, len(self.d))
plt.plot(self.freq_axis[0:int(len(self.d)/2)],self.freq[0:int(len(self.d)/2)])
        plt.xlabel('Frequency')
        plt.ylabel("Amplitude")
        plt.title("FFT of Pressure Data")
        plt.axis([0, 20000, 0, 3000])
        plt.savefig('FFT of Pressure Data'+str(self.a))
        #plt.show()
        plt.close()
    def linear_fit(self,startpoint,lenofplot):
        self.coeff = np.polyfit(self.e[startpoint:lenofplot, 0], self.e[startpoint:lenofplot, 1], 1)
        self.predict=self.coeff[0] * self.e[0:lenofplot, 0] + self.coeff[1]
        print('regression:', self.coeff[0], '*x+', self.coeff[1])
    def plot_pressure_temperature(self,lenofplot,startpoint=0):
        plt.figure('pressure_temperature'+str(self.a))
        plt.scatter(self.e[startpoint:lenofplot, 0], self.e[startpoint:lenofplot, 1], s=0.01,c='r',label='Original Data')
        plt.axis([15, 45, 0, 10])
        plt.plot(self.e[startpoint:lenofplot, 0],self.predict,c='b',label='Fitted Data')
        self.confidence_interval()
        plt.plot(self.e[startpoint:lenofplot, 0],self.upbb,c='y',label='95% CI')
        plt.plot(self.e[startpoint:lenofplot, 0], self.lobb, c='y')
        aa=r'^{\circ}C'
        plt.xlabel("Temperature("+'$%s$' %aa+')')
        plt.ylabel("Pressure(atm)")
        plt.title("Temperature-Pressure")
        plt.legend(scatterpoints=1,loc=4)
```

```
            plt.savefig('pressure_temperature'+str(self.a))
            plt.close()
            #plt.show()
        def plot_time_v(self,startpoint=0):
            plt.figure('v-time'+str(self.a))
            lenofplot = self.totallen
            plt.scatter(self.e[startpoint:lenofplot, 2]*500,
self.e[startpoint:lenofplot, 3],c='b', s=0.01,label='Time-V1')
            plt.scatter(self.e[startpoint:lenofplot, 2]*500,
self.e[startpoint:lenofplot, 4],c='r', s=0.01,label='Time-V2')
            plt.axis([0,4000*10,0,4])
            plt.xlabel("Time(s)")
            plt.ylabel("Voltage(V)")
            plt.title("Time-Voltage")
            plt.legend(scatterpoints=1,loc=4)
            plt.savefig('Time-Voltage' + str(self.a))
            plt.close()
            #plt.show()
            return None
        def plot_temperture_v(self,startpoint=0):
            plt.figure('Temperature-Voltage'+str(self.a))
            lenofplot = self.totallen
            plt.scatter(self.e[startpoint:lenofplot, 0],
self.e[startpoint:lenofplot, 3],c='b', s=0.01,label='Temperature-V1')
            plt.scatter(self.e[startpoint:lenofplot, 0],
self.e[startpoint:lenofplot, 4],c='r', s=0.01,label='Temperature-V2')
            plt.axis([0, 55,0,4])
            aa=r'^{\circ}C'
            plt.xlabel("Temperature("+'$%s$' %aa+')')
            plt.ylabel("Voltage(V)")
            plt.title("Temperature-Voltage")
            plt.legend(scatterpoints=1,loc=4)
            plt.savefig('Temperature-Voltage' + str(self.a))
            plt.close()
            #plt.show()
            return None
        def PCC(self,lenofplot,startpoint):
            with open("pcc%d.txt"%self.a, "w") as f:
                f.write('Peasonor Coefficient is:
'+str(pearsonr(self.e[startpoint:lenofplot, 0], self.e[startpoint:lenofplot,
1]))+'\n'
                        '95%confidence interval=')
            return None
        def error_analysis(self,lenofplot,startpoint):
            self.mse=mean_squared_error(self.e[startpoint:lenofplot,
1], self.coeff[0] * self.e[startpoint:lenofplot, 0] + self.coeff[1])
            self.se=np.sqrt(self.mse)
            print('self.se is ',self.se,'self.mse is',self.mse)
        def plot_std_residual(self,a,b):
            plt.scatter(a,b,s=0.01,label='Standardlized Errors')
            aa=r'^{\circ}C'
            plt.xlabel("Temperature("+'$%s$' %aa+')')
            plt.ylabel("std error(V)")
            plt.title("Residual")
            plt.axis([15,45,-4,4])
            plt.legend(scatterpoints=1,loc=4)
            plt.savefig('Standardized Errors' + str(self.a))
            plt.close()
            return None
        def confidence_interval(self):
            #95% CI
            z=1.96
            self.upbb=self.predict+z*self.se
            self.lobb=self.predict-z*self.se
        def exclude_outliers(self,startpoint,lenofplot,a):
            self.residual=self.predict-self.e[0:lenofplot,1]
            self.plot_std_residual(self.e[0:lenofplot,0],self.residual)
            out_index=[]
            for idx,val in enumerate(self.residual):
                if abs(val)>2*self.se:
                    out_index.append(idx)
            self.e=np.delete(self.e,out_index,0)
            return None
        def make_error_bar(self):
            ut=5e-5
            uD=5e-4
            uV=1e-6
            uS=2*0.01
            uVs=27.5e-3
            t=Symbol('t')
            E=69e9
            D=Symbol('D')
            v1=Symbol('v1')
            v2=Symbol('v2')
            refv1=Symbol('refv1')
            refv2=Symbol('refv2')
            vs=Symbol('vs')
            A=98.8
            S=2
            eq=(16*E*t*(v1-refv1+0.35*v2-
refv2))/(A*D*S*vs*0.8775)

self.uncertainty=(diff(eq,t)*ut)**2+(diff(eq,D)*uD)**2+(diff(eq,v1)*
uV)**2+(diff(eq,v2)*uV)**2+(diff(eq,refv1)*uV)**2+(diff(eq,refv2)
*uV)**2+(diff(eq,vs)*uVs)**2
            print(self.uncertainty)
        def plot_error_bar(self,startpoint,lenofplot):
            t=1e-4
            D=66.4e-3
            vs=5
            refv1=self.refV1
            refv2=self.refV2
            v1=self.e[startpoint:lenofplot,2]
            v2=self.e[startpoint:lenofplot,3]
```

```
            error_bar=np.empty(v1.shape[0])

            error_random=np.empty(v1.shape[0])

            relative_error=np.empty(v1.shape[0])

            relative_random=np.empty(v1.shape[0])

            real_realtive=np.empty(v1.shape[0])

            ERR=np.load('ERR.npy')

            aa = r'^{\circ}C'

            for i in range(v1.shape[0]):

                error_bar[i]=1e-6*sqrt(12658214858.7309 * t ** 2 / (D
** 2 * vs ** 2) + 3.0657405882835e+18 * t ** 2 * (

                                                    -refv1
- refv2 + v1[i] + 0.35 * v2[i]) ** 2 / (

                                                D ** 2 * vs ** 4) +
10134679630689.3 * (

                                                    -
refv1 - refv2 + v1[i] + 0.35 * v2[i]) ** 2 / (

                                                    D ** 2 * vs
** 2) + 1.01346796306893e+15 * t ** 2 * (


-refv1 - refv2 + v1[i] + 0.35 * v2[i]) ** 2 / (


D ** 4 * vs ** 2))+self.predict[i]*0.3*100000

                error_random[i]=error_bar[i]-
self.predict[i]*0.3*100000# random error, theroy

                relative_error[i]=(error_bar[i]/self.predict[i])/10000# not
used


relative_random[i]=(error_random[i]/self.predict[i])/10000# random
error, relative, theory

                real_realtive[i]=1.96*find_std(self.e[i,0])/self.predict[i]#
real error based on 4 trials, 95%CI

            "plt.figure('Absolute Uncertainty Distribution' + str(self.a))


plt.scatter(self.e[startpoint:lenofplot,0],error_bar/100000,s=0.01)

            plt.xlabel("Temperature(" + '$%s$' % aa + ')')

            plt.ylabel("Absolute Uncertainty (atm)")

            plt.title("Absolute Uncertainty Distribution")

            plt.savefig('absolute error distribution' + str(self.a))

            plt.close()

            #-----------------------------------

            plt.figure('Relative Uncertainty Distribution'+str(self.a))


plt.scatter(self.e[startpoint:lenofplot,0],relative_error*100,s=0.01)

            aa = r'^{\circ}C'

            plt.xlabel("Temperature(" + '$%s$' % aa + ')')

            plt.ylabel("Relative Uncertainty bar(%)")

            plt.title("Relative Uncertainty Distribution")

            plt.savefig('relative error distribution' + str(self.a))

            plt.close()

            '''

            #--------------------------

            plt.figure('Theoretical Absolute Uncertainty Distribution')


plt.scatter(self.e[startpoint:lenofplot,0],(error_random)/100000,s=0.01
)

            plt.xlabel("Temperature(" + '$%s$' % aa + ')')

            plt.ylabel("Absolute Uncertainty Error bar(atm)")

            plt.title("Theoretical Absolute Uncertainty Distribution")

            plt.savefig('Theoretical Absolute Uncertainty Distribution'
+ str(self.a))

            plt.close()

            #----------------------

            plt.figure('Theoretical Relative Uncertainty Error
Distribution')


plt.scatter(self.e[startpoint:lenofplot,0],(relative_random)*100,s=0.01)

            plt.xlabel("Temperature(" + '$%s$' % aa + ')')

            plt.ylabel("Theoretical Relative Uncertainty Error bar(%)")

            plt.title("Theoretical Relative Uncertainty Error
Distribution")

            plt.savefig('Theoretical Relative Uncertainty Error
Distribution' + str(self.a))

            plt.close()

            #----------------

            '''

            plt.figure('random only relative')


plt.scatter(self.e[startpoint:lenofplot,0],(relative_random)*100,s=0.01)

            plt.xlabel("Temperature(" + '$%s$' % aa + ')')

            plt.ylabel("Relative Uncertainty Error bar(%)")

            plt.title("Relative Uncertainty Error Distribution")

            plt.savefig('Relative Uncertainty error distribution' +
str(self.a))

            plt.close()

            '''

            #---------------------

            plt.figure('Real Relative Uncertainty Error Distribution')


plt.scatter(self.e[startpoint:lenofplot,0],real_realtive*100,s=0.01)

            plt.xlabel("Temperature(" + '$%s$' % aa + ')')

            plt.ylabel("Real Relative Uncertainty Error
Distribution(%)")

            plt.title("Real Relative Uncertainty Error Distribution")

            plt.savefig('Real Relative Uncertainty Error Distribution' +
str(self.a))

            plt.close()

        def linear_regression(x, y, prob):

            """

            Return the linear regression parameters and their <prob>
confidence intervals.

            ex:

            #>>> linear_regression([.1,.2,.3],[10,11,11.5],0.95)

            """

            x = np.array(x)

            y = np.array(y)

            n = len(x)

            xy = x * y

            xx = x * x


            # estimates
```

```
        b1 = (xy.mean() - x.mean() * y.mean()) / (xx.mean() -
x.mean() ** 2)
        b0 = y.mean() - b1 * x.mean()
        s2 = 1. / n * sum([(y[i] - b0 - b1 * x[i]) ** 2 for i in
xrange(n)])
        print(    'b0 = ', b0)
        print('b1 = ', b1)
        print('s2 = ', s2)

        # confidence intervals

        alpha = 1 - prob
        c1 = scipy.stats.chi2.ppf(alpha / 2., n - 2)
        c2 = scipy.stats.chi2.ppf(1 - alpha / 2., n - 2)
        print(
        'the confidence interval of s2 is: ', [n * s2 / c2, n * s2 / c1])

        c = -1 * scipy.stats.t.ppf(alpha / 2., n - 2)
        bb1 = c * (s2 / ((n - 2) * (xx.mean() - (x.mean()) ** 2))) ** .5
        print(
        'the confidence interval of b1 is: ', [b1 - bb1, b1 + bb1])

        bb0 = c * ((s2 / (n - 2)) * (1 + (x.mean()) ** 2 / (xx.mean() -
(x.mean()) ** 2))) ** .5
        print    ('the confidence interval of b0 is: ', [b0 - bb0, b0 +
bb0])
        return None
    def analysis(a,lenofplot):
        print('This is the ',a,'th trial')
        if a==3:
            startpoint=4000
        elif a==4:
            startpoint=3000
        elif a==5:
            startpoint=2000
        else:
            startpoint=1000
        trial=data_plot(a)
        #-----initial fit
        trial.linear_fit(startpoint,lenofplot)
        trial.PCC(lenofplot,startpoint)
        trial.error_analysis(lenofplot,startpoint)
        trial.exclude_outliers(startpoint,lenofplot,a)
        if a==2:
            trial.linear_fit(startpoint, lenofplot)
            trial.PCC(lenofplot, startpoint)
            trial.error_analysis(lenofplot, startpoint)
            trial.exclude_outliers(startpoint, lenofplot, a)
        #-------------------------
        trial.linear_fit(0, lenofplot)
        trial.plot_pressure_temperature(lenofplot)
        trial.plot_temperture_v()
        trial.plot_time_v()
        trial.plot_error_bar(startpoint,lenofplot)
        return None
    def analysis_not_killed(a,lenofplot):
        print('Outliers not killed')
        print('This is the ', a, 'th trial')
        startpoint=0
        trial = data_plot(a)
        trial.linear_fit(startpoint, lenofplot)
        trial.PCC(lenofplot, startpoint)
        trial.error_analysis(lenofplot, startpoint)
        #trial.plot_pressure_temperature(lenofplot)
        #trial.plot_temperture_v()
        #trial.plot_time_v()
        #trial.exclude_outliers(startpoint, lenofplot, a)
    def combine(a):
        x=np.linspace(15,50,500)
        y1=0.0946351550329 *x+ 0.770132957906
        y2=0.0853694646877 *x+ 4.03822156582
        y3=0.0845234089432 *x+ 2.42978154566
        y4=0.175470966994 *x+ -1.97749171132
        y5=0.129940121323 *x+ 0.550779010841
        plt.figure('combine'+str(a))
        if a==1:
            plt.plot(x,y1,c='r',label='trail1')
            plt.plot(x, y2, c='g', label='trail2')
            plt.plot(x, y3, c='b', label='trail3')
            plt.plot(x, y5, c='k', label='trail5')
        if a==2:
            plt.plot(x, y4, c='y', label='trail4')
            plt.plot(x, y5, c='k', label='trail5')
        aa = r'^{\circ}C'
        plt.xlabel("Temperature(" + '$%s$' % aa + ')')
        plt.ylabel("Pressure(atm)")
        plt.title("Temperature-Pressure")
        plt.legend(scatterpoints=1, loc=4)
        plt.savefig('combined'+str(a))
        plt.close()

for i in range(1,6):
    analysis(i,12000)

for i in range(1,6):
    trial = data_plot(i)
    trial.find_frequency()
```