

PHYC40970- Advanced Laboratory II



Differential Scanning Calorimeter

Nathan Power

19311361

Abstract

In this experiment, a simple and inexpensive calorimetric system was used. It was shown that, after proper calibration, the system was utilised to measure thermodynamic quantities of a first-order phase transition. The main aim for this experiment was to obtain values for the transition temperature, latent heat and change in specific heat for the first-order phase transition that is the freezing of water. It was shown that the values obtained coincided with accepted values within literature.[1]

Introduction

Phases are states of matter characterised by distinct macroscopic properties. Typical phases include liquid, solid and gas. States of matter come with their stability regions, as seen below in the phase diagram for water, Fig.1. The properties of the microscopic state change by definition at these phase boundaries.

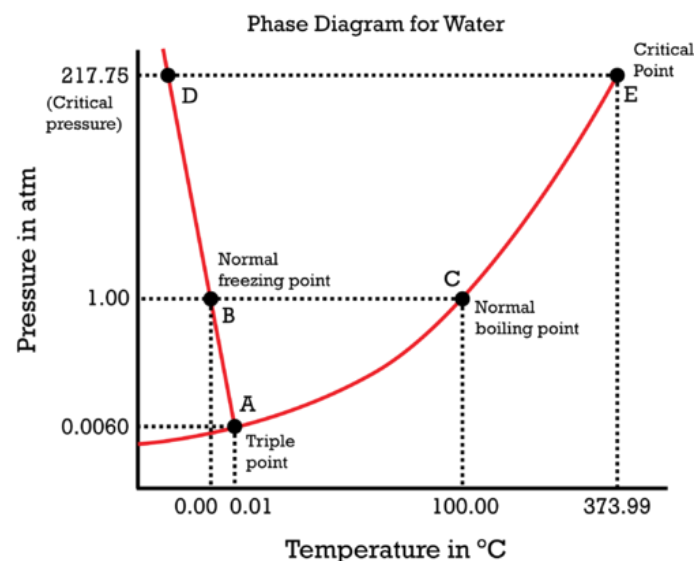


Fig.1 shows the phase diagram for water [2]

When water starts boiling, it undergoes a phase transition from a liquid to a gas phase. For both phases independently, the equation of state is a well-defined regular function, continuous, with continuous derivatives. However, while going from liquid to gas one function sharply changes to the other function. Such a transition is known as a first-order transition.[3]

First-order phase transitions involve a latent heat. During a first-order transition, the system either absorbs or releases a fixed amount of energy per volume. During this process, the temperature of the system will stay constant as heat is added. We can see this during the

boiling of water as the water does not instantly turn into vapour, but forms a turbulent mixture of liquid water and vapour bubbles. [4]

Theory

Energy is required to melt a solid because the cohesive bonds between the molecules in the solid must be broken apart such that, in the liquid, the molecules can move around at comparable kinetic energies; thus, there is no rise in temperature. There is no temperature change until a phase change is complete. The temperature of a cup of water initially at 0°C stays at 0°C until all the ice has melted. Conversely, energy is released during freezing and condensation, usually in the form of thermal energy. We can see this energy at work in the transitional peaks in the thermograms later in this report

Phase changes can have a tremendous stabilising effect even on temperatures that are not near the melting and boiling points, because evaporation and condensation occur even at temperatures below the boiling point. For example, the fact that air temperatures in humid areas stay relatively stable is because most heat transfer goes into evaporating water into the air. Similarly, temperatures in humid weather rarely fall below the dew point because enormous heat is released when water vapour condenses.[5]

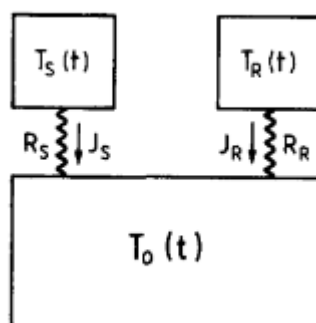


Fig.2 shows the a typical DSC [2]

Differential scanning calorimetry (DSC) is a thermoanalytical technique in which the difference in the amount of heat required to change the temperature of a sample and reference is measured as a function of temperature. A relatively simple description of the operation of a DSC can be performed under the assumption that the heat flow rate between two points is proportional to the temperature difference between those points which is thermal Ohm's law.[1]

Heating or cooling the calorimeter at a rate dT/dt causes a time evolution of the temperatures of the calorimetric block T_0 , the sample T_S and the reference T_R . If a reaction is taking place in the sample, causing its enthalpy to vary at a rate dh/dt , the equations of thermal balance for sample and reference are:

$$\frac{dh}{dt} = C_S \frac{dT_S}{dt} + \frac{T_S - T_0}{R_S},$$

$$0 = C_R \frac{dT_R}{dt} + \frac{T_R - T_0}{R_R},$$

Eq.1 + Eq.2

where C_S and C_R are the heat capacities of sample and reference respectively. R_S and R_R are the thermal resistances between sample and block and between reference and block respectively.[1]

The electric output of the calorimeter, Y , is proportional to the temperature difference between sample and reference. Since it is common to have a negative voltage for an exothermic reaction ($T_S > T_R$), then $Y = B(-\Delta T)$, resulting in the following expression for the calorimetric output;

$$Y(t) = S \left[-\frac{dh}{dt} + (C_S - C_R)\dot{T}_0 \right],$$

Eq.3

Where S is the sensitivity of the calorimeter.

The advantages of the differential strategy clearly show up from this equation. The drift in the calorimetric output associated with T_0 is minimised by the fact that the heat flow of the reference is subtracted from that of the sample, and therefore the desired quantity dh/dt is the major contribution to the calorimetric output.[1]

In the absence of any reaction ($dh/dt=0$), the calorimetric output is proportional to the product between the heating/cooling rate, and the difference in specific heat of sample and reference. Hence, for a sample undergoing a phase transition it is possible to obtain the difference in the specific heat of the two phases by a simple comparison of the baseline before (Y') and after (Y'') the phase transition:

$$\frac{Y'' - Y'}{S\dot{T}_0} = C_S^I - C_S^{II} \equiv \Delta C_p$$

Eq.4

The enthalpy change at a phase transition ΔH is obtained by integration of dh/dt over a time interval large enough to enable the calorimeter to return to its steady state after completion of the phase transition:[1]

$$\Delta H = \int_{t_1}^{t_2} \left[-\frac{Y}{S} + (C_S - C_R)\dot{T}_0 \right] dt.$$

Eq.5

Experimental Procedure

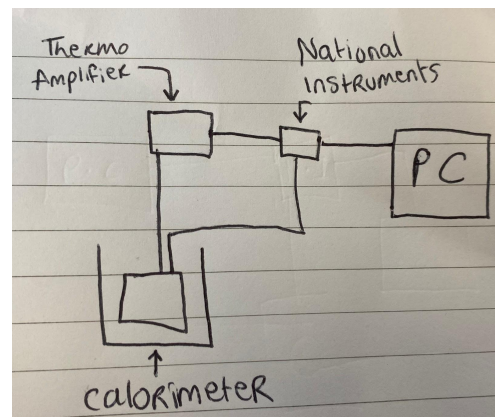
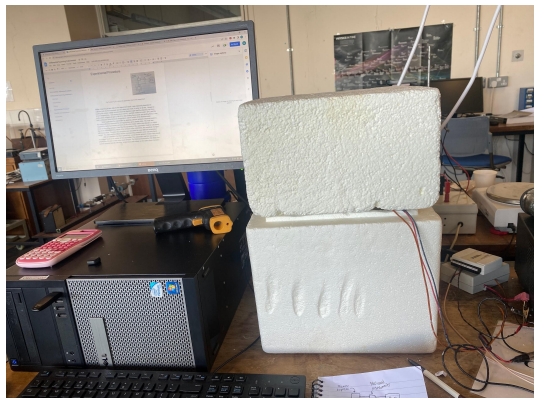


Fig.3a and Fig.3b shows the apparatus used in this experiment

Before beginning to take data for the experiment itself, it was extremely important to calibrate both the thermocouple and the calorimeter itself. In order to do this, I began by tracking the output voltage recorded from the thermocouple itself by exposing it to three known temperatures, room temperature (18.4°C), partially melted ice water (0°C) and the boiling point of liquid nitrogen (-195.8°C). After finding these points and plotting them, I now had a calibration curve between temperature and the output of the thermocouple.

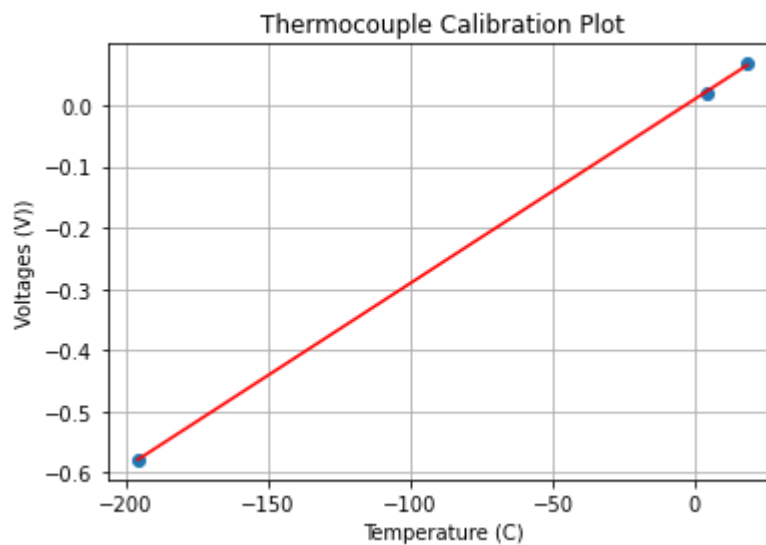
Next, I calibrated the calorimeter by applying a known amount of thermal power to the calorimeter and measured the electrical output at the steady state. The heat was supplied by applying a known current to a 100 ohm resistor which was placed in a copper block on one of the sensors. From this, the sensitivity of the calorimeter could be calculated. Due to the semiconducting nature of the sensors of the calorimeter, the sensitivity changes with temperature. Therefore the sensitivity was calculated at many different temperature ranges. This data was then fitted and the corresponding parabolic curve was then found.

After all calibration had been carried out, a small known amount of water was placed in a copper block on the sensor of the calorimeter. The calorimeter was then placed in a polystyrene box and cooled using liquid nitrogen. The National Instruments USB 6008 was used to record a data file with time, temperature and voltage as the water phase transition took place. This data was then analysed and plotted in order to calculate the transition temperature, latent heat and change in specific heat for each experimental run.

Results and Discussion

Calibration of Thermocouple

Before beginning to take data for the experiment itself, it was extremely important to calibrate both the thermocouple and the calorimeter itself. This was done as described above in the previous section.



*Fig.4 shows the thermocouple calibration plot with slope;
 $m = 0.003 \pm 5.8 \times 10^{-5}$*

It was found that the thermocouple produced $0.003 \pm 5.8 \times 10^{-5}$ V per Kelvin.

Calibration of the Calorimeter

Next, I calibrated the calorimeter using the above method. A voltage of 5.86 V and current of 24.3mA was applied to a resistor of 100 ohms. Using the equation, $P = I^2 R$, the thermal power produced was found to be 0.059 Watts.

I then found thermograms for the calorimeter at different temperatures within the experimental range to find the steady state voltage at certain temperatures when the thermal power was not applied. The V_{ss} was found by observing the depth of the well created on the thermograms when the power was disconnected.

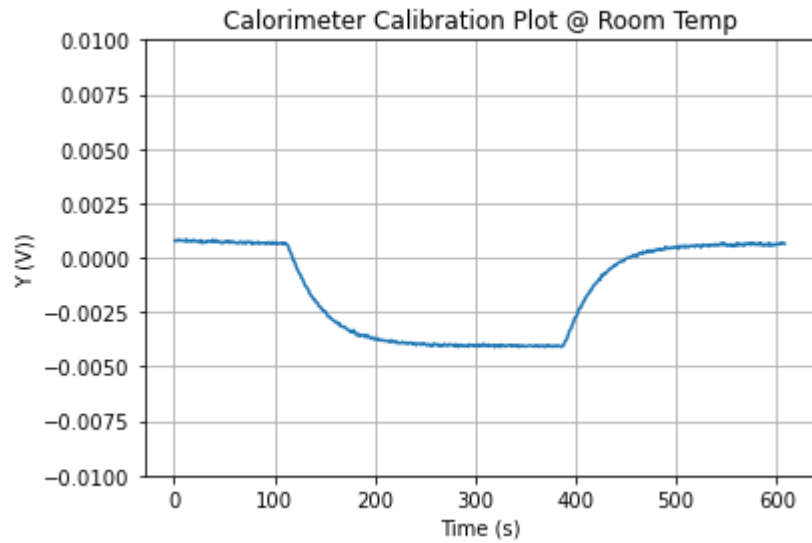


Fig.5 shows the calibration thermogram showing the calorimetric output at Room Temperature (18.4°C) as a function of time.

Using the equation for the sensitivity of the sensors, $S = \frac{q}{V_{ss}}$ (where q is the thermal power), we plotted our values for sensitivity at different temperatures and fitted the data with a parabolic curve.

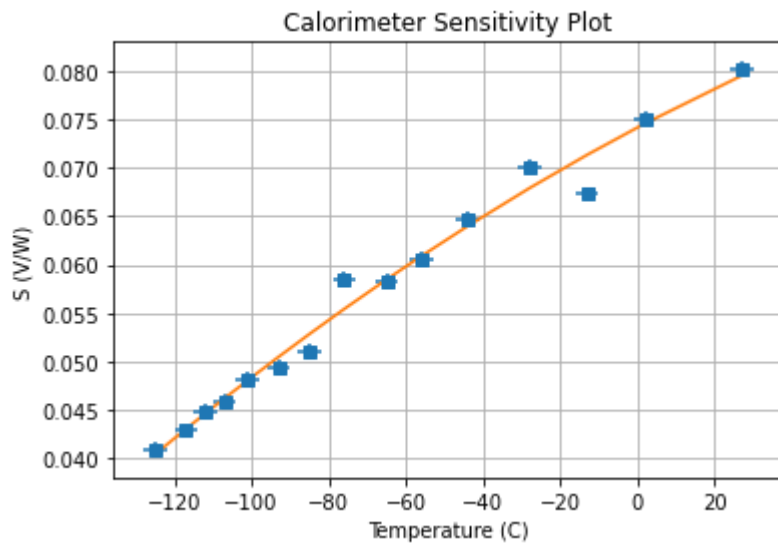


Fig.6 shows the sensitivity as a function of temperature. The solid line is a parabolic fit to the data.
 $y = (-4.72e-07)*T^2 + (2.1e-04)*T + 0.074$

Now that the equipment was calibrated, it was possible to use it accurately at extreme temperatures within our experimental range in order to obtain the results needed.

Investigating the Freezing of Water

The aim for this part of the experiment is to obtain values for the transition temperature, latent heat and change in specific heat for each experimental run. To achieve this, I recorded data lists of time, temperature and voltage while the calorimeter was being cooled by liquid nitrogen.

From this I was able to calculate the change in temperature over the time the calorimeter cooled. This was important because, by taking the slope of Temperature vs Time as dT/dt , I was able to eliminate some noise from the plots, caused by errors on the temperatures.

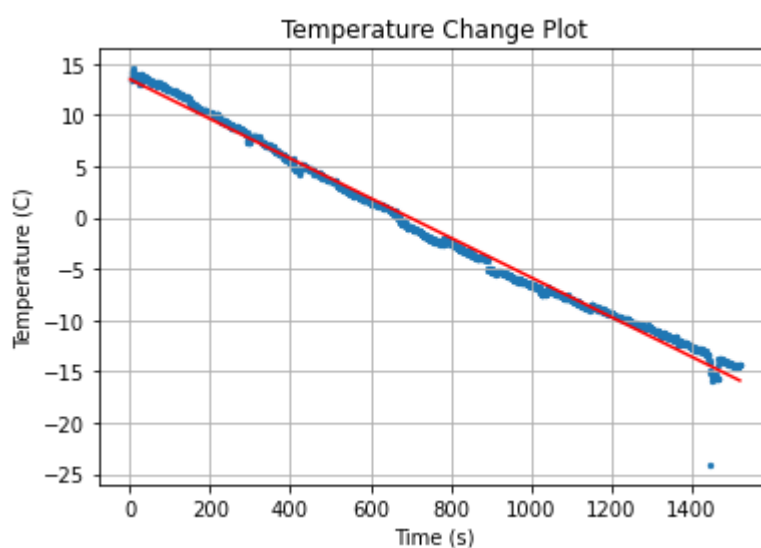


Fig.7 shows the decrease in temperature over time during a run of the experiment.

Next, I found the heat capacity, dQ/dT , of the system throughout the run. This was found by dividing the electric output at each temperature by the cooling rate and the sensitivity of the calorimeter at this temperature. By plotting this thermogram we can find the change in heat capacity. The change in heat capacity can be seen by the change in baseline after the transition

Table.1 shows the results found during this experiment

Run Number	Mass (g)	Transition Temperature (°C)	Latent Heat ΔH (J/g)	ΔC_p (J/g K)
1	0.017	-7.25	-327.06	-2.72
2	0.017	-5.4	-338.82	-2.04
3	0.017	-5.8	-345.88	-1.87

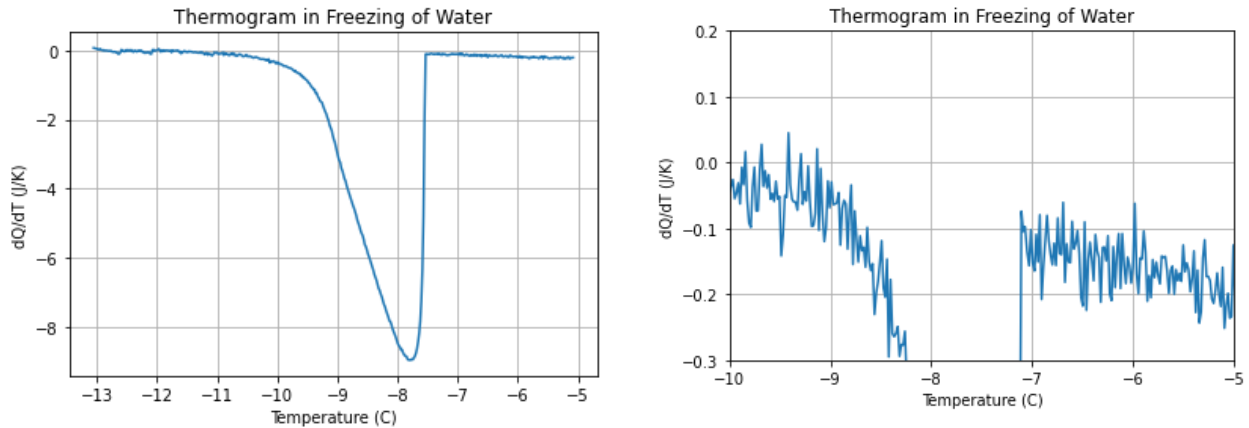


Fig.8 shows a typical thermogram found for the freezing of water. We can clearly observe from the enlarged view that there is a discontinuity in the baseline associated with a change in specific heat.

Numerical Integration of the thermogram renders the value for the latent heat, ΔH , at the transition. While taking the integration it was important that the baseline was at zero to ensure an accurate enthalpy change result. It is worth noting that the transition peak did not occur at zero. This may be due to the slow cooling of the calorimeter or the significant surface tension of the small container.

Using our results from Table.1, and neglecting the temperature dependence of specific heat, we can compare our values to theoretically accepted values. Using 4.18 J/g K for the specific heat capacity of water and 2.01 J/g K for ice, we find a change of -2.17 J/g K which agrees with our experimental value of -2.21 ± 0.12 J/g K

When comparing ΔH we can take into account that the transition does not take place at zero. The dependence of latent heat on temperature is given by;

$$\Delta H(T) = \Delta H(T_e) + \Delta C_p(T - T_e)$$

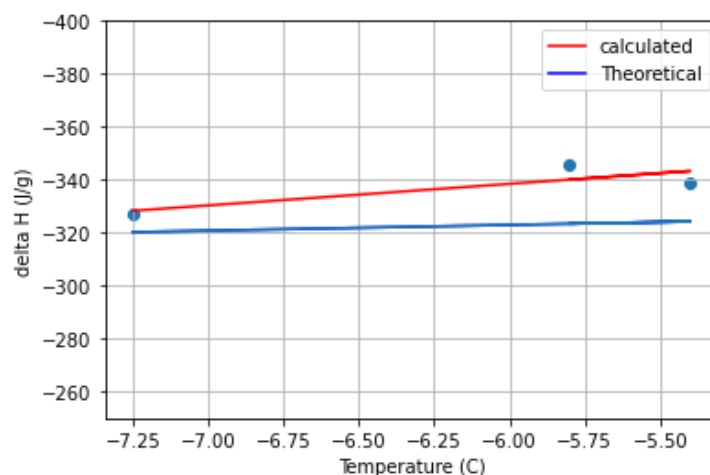


Fig.9 shows measured latent heat as a function of transition temperature. The theoretical line shows $\Delta H(T) = \Delta H(T_e) + \Delta C_p(T - T_e)$ where $\Delta H(T_e) = -336$ J/g and $\Delta C_p = -2.17$ J/g K.

$$\text{Expected value for } \Delta H(T_e) = -336 \pm 0.57 \text{ J/g}$$

$$\text{Calculated value for } \Delta H(T_e) = -387.26 \pm 2.58 \text{ J/g}$$

In Fig.9 , we can see there is an agreement between the theoretical and calculated values for the temperature dependence of the latent heat. However, if I had time and had taken more successful runs of the experiment then I believe the discrepancy between them would be less than the significant 13% it was calculated at.

Conclusion

From the results, it can be clearly seen that most of the data analysis agreed with the expected results.[1] We can clearly see that we were able to calculate the values for the transition temperature, latent heat and change in specific heat. It was found that the values for the change in specific heat during the freezing of water agreed with the values we expected from theory. There was a significant difference in the values found for the latent heat for the transition, however I believe had I had more time to conduct more successful runs of the experiment, I would have minimised this error significantly. There may have also been errors which arose due to the baseline not being completely at zero while integrating to find the latent heat.

References

- [1] Lluís Mañosa, Marc Bou, Carme Calles, and Albert Cirera , "Low-cost differential scanning calorimeter", American Journal of Physics 64, 283-287 (1996) <https://doi.org/10.1119/1.18216>
- [2] Coursehero.com. (2022). [online] Available at: <https://www.coursehero.com/study-guides/cheminter/phase-diagram-for-water/>.
- [3] Chapter 6 Phase transitions 6.1 Concept of phase. (n.d.). [online] Available at: https://itp.uni-frankfurt.de/~gros/Vorlesungen/TD/6_Phase_transitions.pdf.
- [4] Wikipedia. (2022). Phase transition. [online] Available at: https://en.wikipedia.org/wiki/Phase_transition#Types_of_phase_transition [Accessed 1 Nov. 2022].
- [5] Phase Change and Latent Heat | Physics. [online] Available at: <https://courses.lumenlearning.com/atd-austincc-physics1/chapter/14-3-phase-change-and-latent-heat/>

```
In [1]: from pydaqmx_helper.adc import ADC
import time
import numpy as np
import matplotlib.pyplot as plt
from scipy.optimize import curve_fit

myADC = ADC()
myADC.addChannels([1,2], minRange=-1,maxRange=1)

Activated Channel 1
Activated Channel 2

In [2]: data = myADC.sampleVoltages(100,10)

samples1 = data[1] #voltage
samples2 = data[2] #temperature

In [22]: print(np.mean(samples1))
print(np.mean(samples2)/0.003006396798762452)

0.0006444848157306349
31.236207378571727

In [4]: def thermo_cal(start_time,time_step,end_time):
    voltage = np.array([])
    temperature = np.array([])
    while start_time < end_time:
        data = myADC.sampleVoltages(1000,1000)
        voltage = np.append(voltage,np.mean(data[1]))
        temperature = np.append(temperature,np.mean(data[2]))
        start_time += time_step #starts loop again for allotted time
    return temperature, voltage
```

Thermocouple Calibration

```
Room Temp

In [7]: room_temp = thermo_cal(0,1,301) # 18.4 degrees
roomtemp1=room_temp[0]
np.mean(roomtemp1)

Out[7]: 0.06946288744444888

In [24]: room_temp2 = thermo_cal(0,1,301) # 18.4 degrees
roomtemp2=room_temp2[0]
room_run2=np.mean(roomtemp2)

Out[24]: 0.0657890740014618

In [25]: room_temp3 = thermo_cal(0,1,301) # 18.4 degrees
roomtemp3=room_temp3[0]
np.mean(roomtemp3)

Out[25]: 0.0671501501539575

In [26]: room_temp4 = thermo_cal(0,1,301) # 18.4 degrees
roomtemp4=room_temp4[0]
np.mean(roomtemp4)

Out[26]: 0.06817583916234796

Ice Water

In [10]: ice_temp=thermo_cal(0,1,301)
icetemp1=ice_temp[0]
np.mean(icetemp1) #mean voltage at ice water temp - 4.2 degrees

Out[10]: 0.021029739622043204

In [14]: ice_temp2=thermo_cal(0,1,301)
icetemp2=ice_temp2[0]
np.mean(icetemp2)

Out[14]: 0.01936067473150964

In [20]: ice_temp3=thermo_cal(0,1,301)
icetemp3=ice_temp3[0]
np.mean(icetemp3)

Out[20]: 0.019457306134720955

In [22]: ice_temp4=thermo_cal(0,1,301)
icetemp4=ice_temp4[0]
np.mean(icetemp4)

Out[22]: 0.01943109771247294

Liquid Nitrogen

In [28]: nitro_temp=thermo_cal(0,1,301)
nitrotemp1=nitro_temp[0]
np.mean(nitrotemp1) #mean voltage at room temp - 4.2 degrees

Out[28]: -0.5787154311114199

In [29]: nitro_temp2=thermo_cal(0,1,301)
nitrotemp2=nitro_temp2[0]
np.mean(nitrotemp2)

Out[29]: -0.5788149297820874

In [30]: nitro_temp3=thermo_cal(0,1,301)
nitrotemp3=nitro_temp3[0]
np.mean(nitrotemp3)

Out[30]: -0.5789306225613999

In [31]: nitro_temp4=thermo_cal(0,1,301)
nitrotemp4=nitro_temp4[0]
np.mean(nitrotemp4)

Out[31]: -0.5780536626528351

Plotting Calibration

In [41]: total_room=roomtemp1+roomtemp2+roomtemp3+roomtemp4
std_room=np.std(total_room)
mean_room=np.mean(total_room)/4

total_ice=icetemp1+icetemp2+icetemp3+icetemp4
std_ice=np.std(total_ice)
mean_ice=np.mean(total_ice)/4

total_nitro=nitrotemp1+nitrotemp2+nitrotemp3+nitrotemp4
std_nitro=np.std(total_nitro)
mean_nitro=np.mean(total_nitro)/4

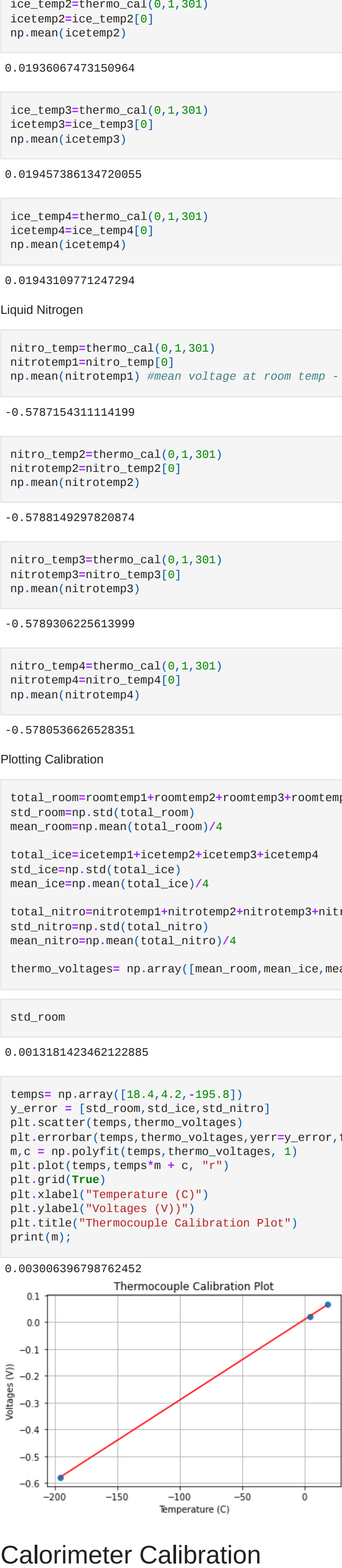
thermo_voltages= np.array([mean_room,mean_ice,mean_nitro])

In [65]: std_room

Out[65]: 0.0013181423462122885

In [77]: temps=np.array([18.4,4.2,-195.0])
y_error = [std_room,std_ice,std_nitro]
plt.scatter(temps,thermo_voltages)
plt.errorbar(temps,thermo_voltages,yerr=y_error,fmt="+")
m,c = np.polyfit(temps,thermo_voltages,1)
plt.plot(temps,temps*m + c, "r")
plt.grid(True)
plt.xlabel("Temperature (C)")
plt.ylabel("Voltages (V)")
plt.title("Thermocouple Calibration Plot")
print(m);

0.003006396798762452
```



Calorimeter Calibration

```
In [23]: def cal_cal(start_time,time_step,end_time):
    printcounter=0
    voltage = np.array([])
    temperature = np.array([])
    seconds=np.array([])
    starttime=time.time()
    while start_time < end_time:
        data = myADC.sampleVoltages(1000,1000)
        voltage = np.append(voltage,np.mean(data[1]))
        temperature = np.append(temperature,np.mean(data[2]))
        timetaken=time.time()
        timewanted=timetaken-starttime
        seconds=np.append(seconds, timewanted)
        if (printcounter == 15):
            print(np.mean(data[1]))
            print(np.mean(data[2])/0.003006396798762452))
            print(timewanted) #prints a voltage value every 15 loops
            printcounter = 0
        printcounter += 1
        start_time += time_step #starts loop again for allotted time
    return temperature/0.003006396798762452, voltage, seconds

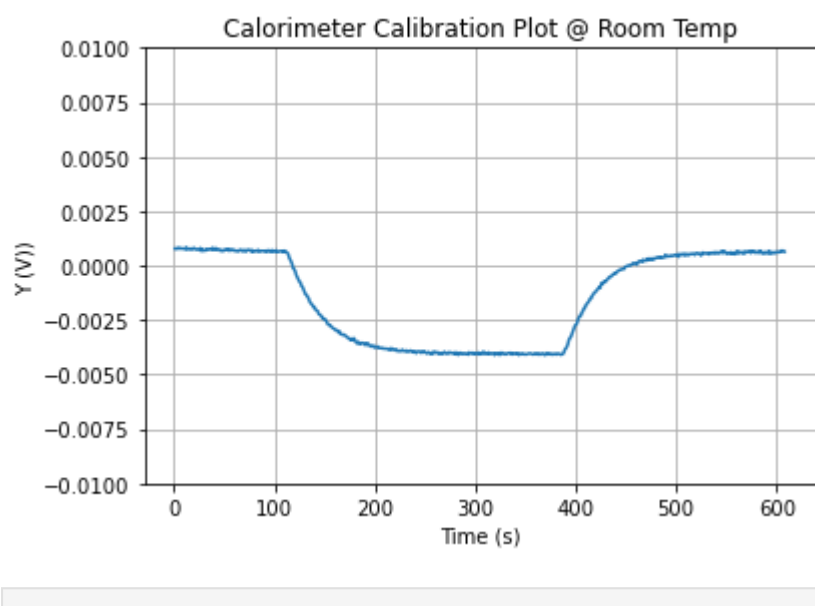
Thermal Power

In [7]: V = 5.86 #V
I = 0.0243 #A
R = 100 #ohms
P = (I**2) * R
P

Out[7]: 0.05904899999999999

Room Temperature Calibration

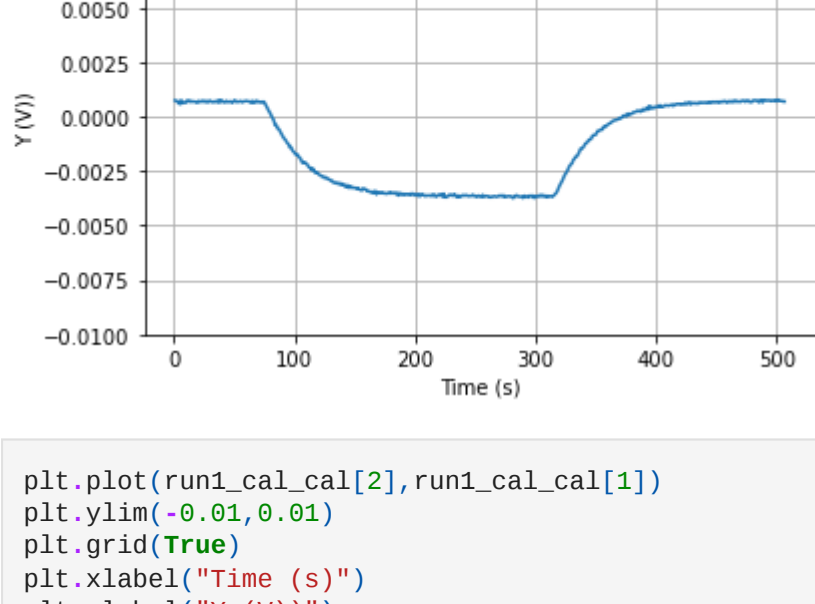
In [17]: np.savetxt("room_cal_cal3.txt",room_cal_cal3)
plt.plot(room_cal_cal3[2],room_cal_cal3[1])
plt.ylim(-0.01,0.01)
plt.grid(True)
plt.xlabel("Time (s)")
plt.ylabel("Y (V)")
plt.title("Calorimeter Calibration Plot @ Room Temp");
```



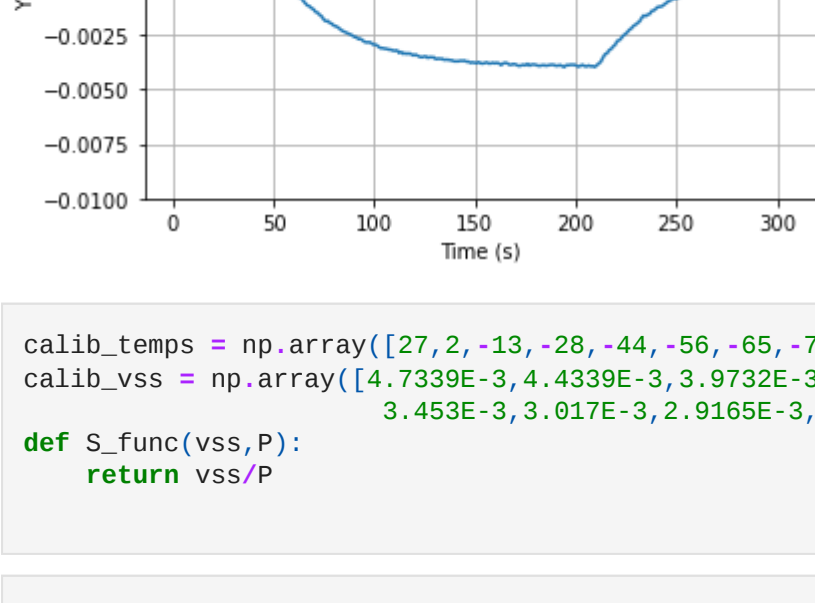
```
In [50]: #temp check
data = myADC.sampleVoltages(1000,1000)
print(np.mean(data[2])/0.003006396798762452)

23.72054327519946

In [51]: plt.plot(run1_cal_cal[2],run1_cal_cal[1])
plt.ylim(-0.01,0.01)
plt.grid(True)
plt.xlabel("Time (s)")
plt.ylabel("Y (V)")
plt.title("Calorimeter Calibration Plot ");
```



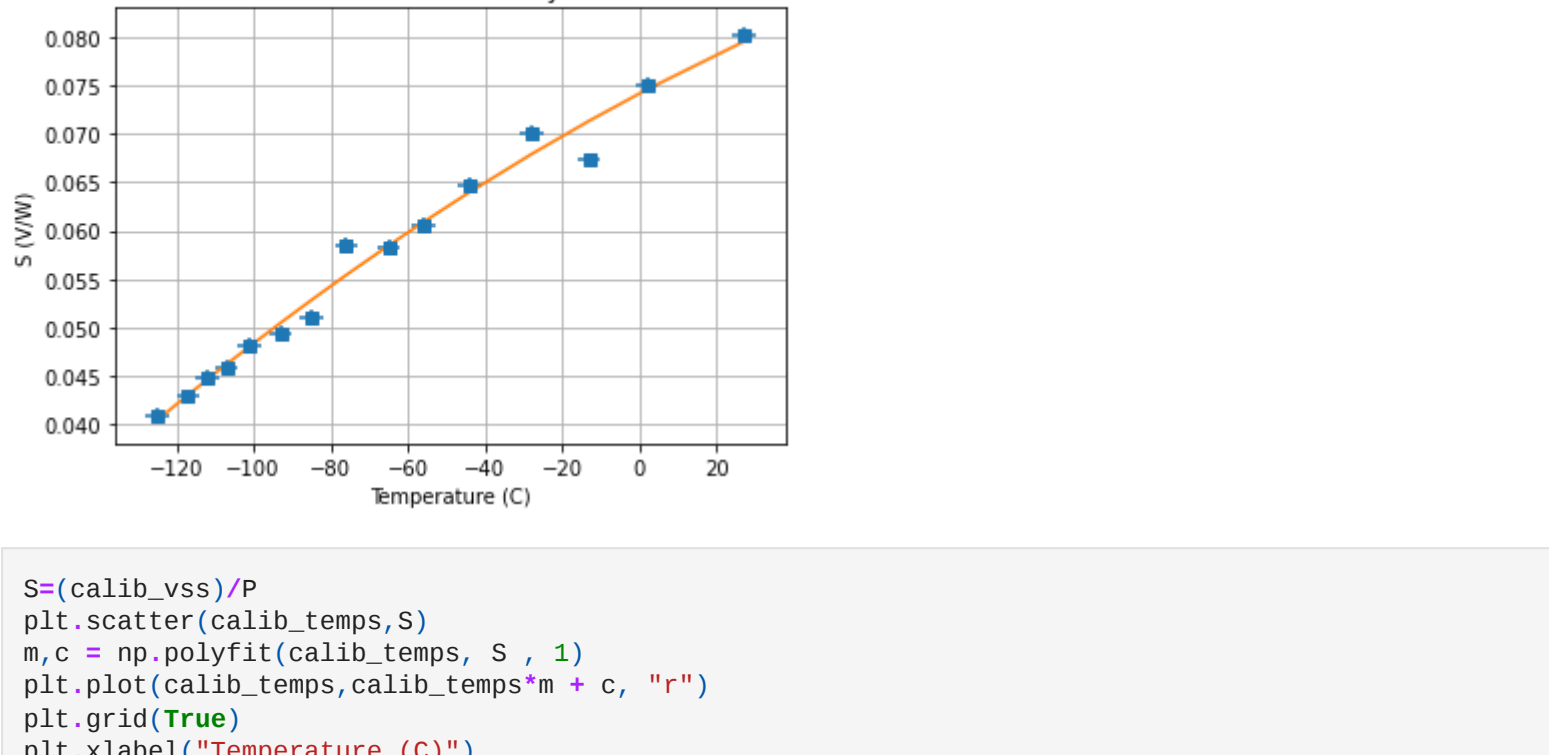
```
In [8]: plt.plot(run1_cal_cal[2],run1_cal_cal[1])
plt.ylim(-0.01,0.01)
plt.grid(True)
plt.xlabel("Time (s)")
plt.ylabel("Y (V)")
plt.title("Calorimeter Calibration Plot ");
```



```
In [5]: calib_temps = np.array([27,-13,-20,-44,-56,-65,-76,-85,-93,-101,-107,-112,-117,-125])
calib_vss = np.array([4.7339E-3,4.4339E-3,3.0732E-3,4.141E-3,3.818E-3,3.58E-3,3.446E-3,3.453E-3,3.017E-3,2.9165E-3,2.845E-3,2.704E-3,2.6445E-3,2.533E-3,2.414E-3])

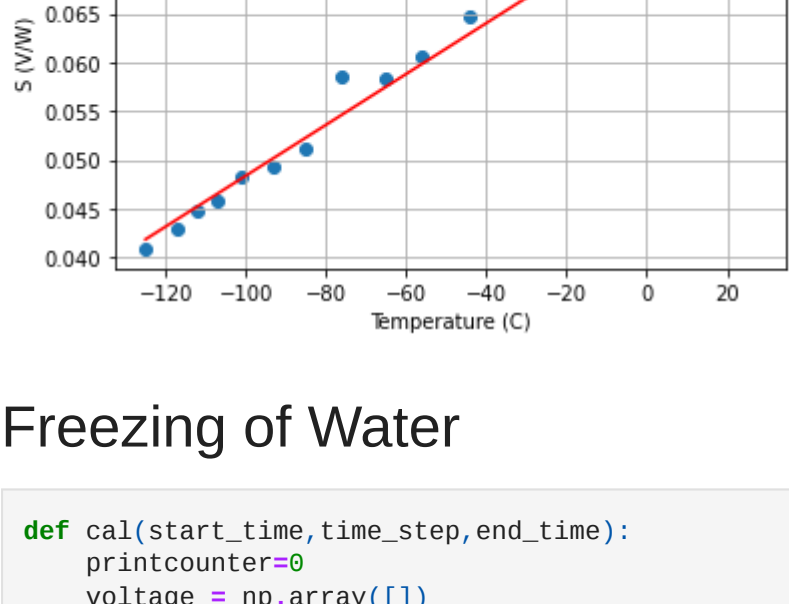
def S_func(vss,P):
    return vss/P

In [9]: def func(x,a,b,c):
    return a*x**2 + b*x + c
S=(calib_vss)/P
x=calib_temps
y=(calib_vss)/P
X_err = np.std(x/np.size(x))
Y_err = np.std(y/np.size(y))
plt.errorbar(x,y,Y_err,X_err, marker= "s", fmt=".")
params, covs = curve_fit(func, x, y)
a, b, c = params[0], params[1], params[2]
yfit1 = a*x**2+b*x+c
plt.plot(x, yfit1)
plt.scatter(calib_temps,S)
plt.grid(True)
plt.xlabel("Temperature (C)")
plt.ylabel("S (V/W)")
plt.title("Calorimeter Sensitivity Plot")
print("y=a*T^2+b*T+c")
print(a,b,c);
```



```
In [109]: S=(calib_vss)/P
plt.scatter(calib_temps,S)
m,c = np.polyfit(calib_temps, S, 1)
plt.plot(calib_temps,calib_temps*m + c, "r")
plt.grid(True)
plt.xlabel("Temperature (C)")
plt.ylabel("S (V/W)")
plt.title("Calorimeter Sensitivity Plot");
m,c
#0.0002611x + 0.07447

Out[109]: (0.00026110493919833267, 0.07447268049229092)
```



Freezing of Water

```
In [3]: def cal(start_time,time_step,end_time):
    printcounter=0
    voltage = np.array([])
    temperature = np.array([])
    std_volt = np.array([])
    seconds=np.array([])
    starttime=time.time()
    while start_time < end_time:
        data = myADC.sampleVoltages(1000,1000)
        std_volt = np.append(std_volt,np.std(data[1]))
        voltage = np.append(voltage,np.mean(data[1]))
        temperature = np.append(temperature,np.mean(data[2]))
        timetaken=time.time()
        timewanted=timetaken-starttime
        seconds=np.append(seconds, timewanted)
        if (printcounter == 15):
            print(np.mean(data[1]))
            print(np.mean(data[2])/0.003006396798762452))
            print(timewanted) #prints a voltage value every 15 loops
            printcounter = 0
        printcounter += 1
        start_time += time_step #starts loop again for allotted time
    return (temperature/0.003006396798762452), voltage, seconds, std_volt/np.sqrt(1000)

In [37]: #temp check
data = myADC.sampleVoltages(1000,1000)
print(np.mean(data[2])/0.003006396798762452)

14.445779340569649

In [76]: def dQ_dT(Y,T_dot,S):
    return Y/(T_dot * S)

In [31]: run4 = np.savetxt("run4.txt",run1)

In [38]: myADC = ADC()
myADC.addChannels([1,2], minRange=-.01,maxRange=.01)

Activated Channel 1
Activated Channel 2

In [40]: run4 = np.savetxt("run5.txt",run1)

In [ ]: def dQ_dT(Y,T_dot,S):
    return Y/(T_dot * S)

In [ ]: def S(T):
    return -4.7267429011132883e-07*(T**2) + 0.00021061357938282626 * T + 0.0741597110675443

In [ ]:
```


Run 1

```
In [561.] run1= np.loadtxt("run3.txt")

In [582.] plt.scatter(run1[2],run1[0])
m1,c1 = np.polyfit(run1[2], run1[0] , 1)
plt.plot(run1[2],run1[2]**m1 + c1, "r")
m1,c1
plt.grid(True)
plt.xlabel("Time (s)")
plt.ylabel("Temperature (C)")
plt.title("Temperature Change Plot")

Out[582.] Text(0.5, 1.0, 'Temperature Change Plot')
```



A scatter plot showing Temperature (C) on the y-axis (ranging from -20 to 5) versus Time (s) on the x-axis (ranging from 0 to 1400). The data points are blue dots, and a red line represents a linear fit. The plot is titled "Temperature Change Plot".

```
In [511.] new_t1 = (run1[2]**-0.01751768946537915) + 6.486199902630516
dq1=dQ_dT((run1[1]),-m1,S(new_t1))
plt.plot(new_t1,dq1,)
plt.grid(True)

In [386.] dq1_new= dq1[750:950]

In [563.] plt.plot(new_t1[650:1100],dq1[650:1100]*-7)
plt.grid(True)
plt.xlabel("Temperature (C)")
plt.ylabel("dQ/dT (J/K)")
plt.title("Thermogram in Freezing of Water")

Out[563.] Text(0.5, 1.0, 'Thermogram in Freezing of Water')
```



A plot showing dQ/dT (J/K) on the y-axis (ranging from -5 to 0) versus Temperature (C) on the x-axis (ranging from -13 to -5). The plot is titled "Thermogram in Freezing of Water".

Run 2

```
In [558.] run3= np.loadtxt("run5.txt")

In [224.] plt.scatter(run3[2],run3[0],s=5)
m,c = np.polyfit(run3[2], run3[0] , 1)
plt.plot(run3[2],run3[2]**m + c, "r")
plt.grid(True)
plt.xlabel("Time (s)")
plt.ylabel("Temperature (C)")
plt.title("Temperature Change Plot")
m,c

Out[224.] (-0.019319551830348905, 13.472771805703307)
```

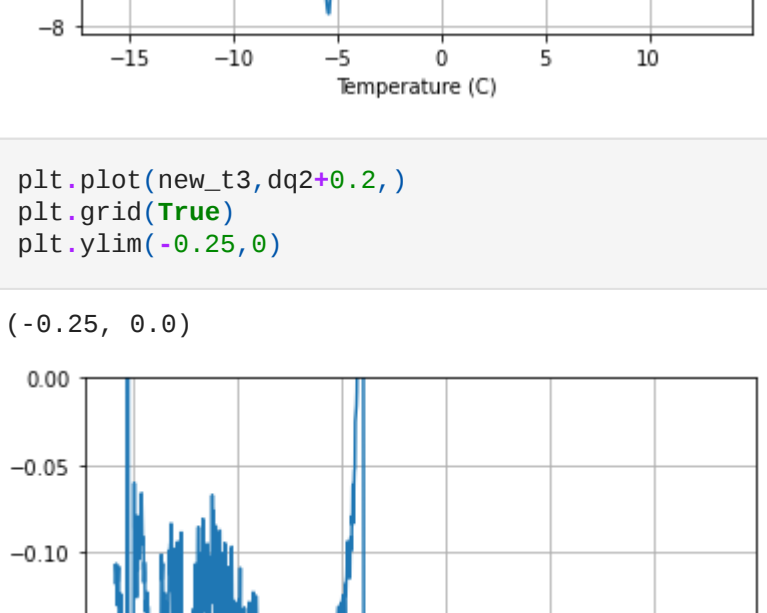


A scatter plot showing Temperature (C) on the y-axis (ranging from -25 to 15) versus Time (s) on the x-axis (ranging from 0 to 1400). The data points are blue dots, and a red line represents a linear fit. The plot is titled "Temperature Change Plot".

```
In [178.] new_t3 = (run3[2]**-0.019319551830348905) + 13.472771805703307
dq2=dQ_dT((run3[1]),-m,S(new_t3))

In [179.] new_t3 = (run3[2]**-0.019319551830348905) + 13.472771805703307
plt.plot(new_t3,dq2*0.45,)
plt.grid(True)
plt.grid(True)
plt.xlabel("Temperature (C)")
plt.ylabel("dQ/dT (J/K)")
plt.title("Thermogram in Freezing of water")

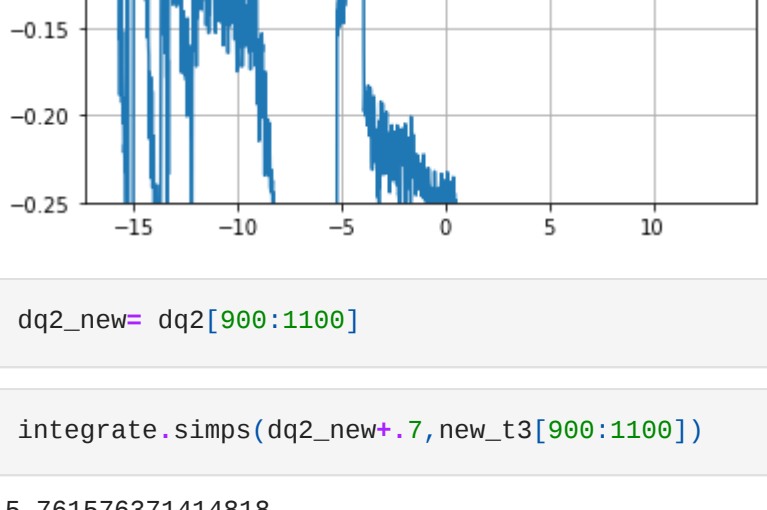
Out[179.] Text(0.5, 1.0, 'Thermogram in freezing of water')
```



A plot showing dQ/dT (J/K) on the y-axis (ranging from -8 to 0) versus Temperature (C) on the x-axis (ranging from -15 to 10). The plot is titled "Thermogram in freezing of water".

```
In [382.] plt.plot(new_t3,dq2*0.2,)
plt.grid(True)
plt.ylim(-0.25,0)

Out[382.] (-0.25, 0.0)
```



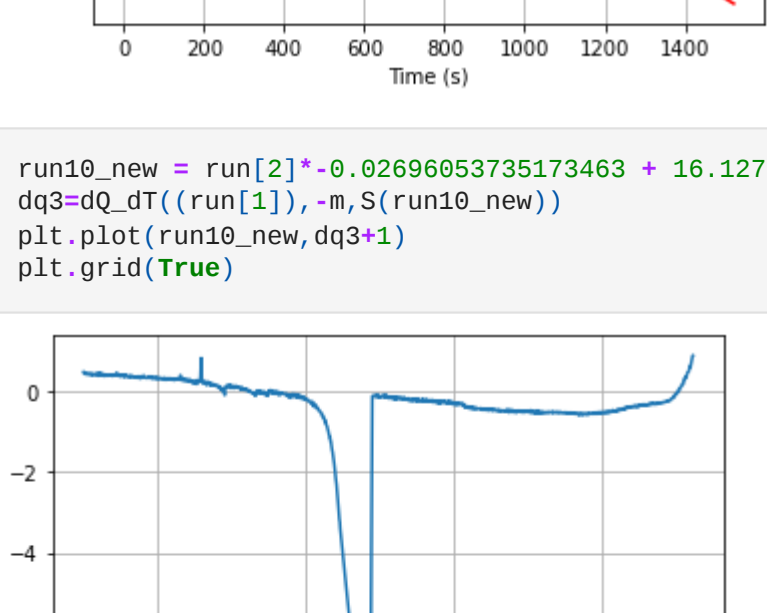
A zoomed-in plot showing dQ/dT (J/K) on the y-axis (ranging from -25 to 0.00) versus Temperature (C) on the x-axis (ranging from -15 to 10). The plot is titled "Thermogram in freezing of water".

Run 3

```
In [417.] run= np.loadtxt("run7.txt")

In [583.] plt.scatter(run[2],run[0])
m,c = np.polyfit(run[2], run[0] , 1)
plt.plot(run[2],run[2]**m + c, "r")
m,c
plt.grid(True)
plt.xlabel("Time (s)")
plt.ylabel("Temperature (C)")
plt.title("Temperature Change Plot")

Out[583.] Text(0.5, 1.0, 'Temperature Change Plot')
```

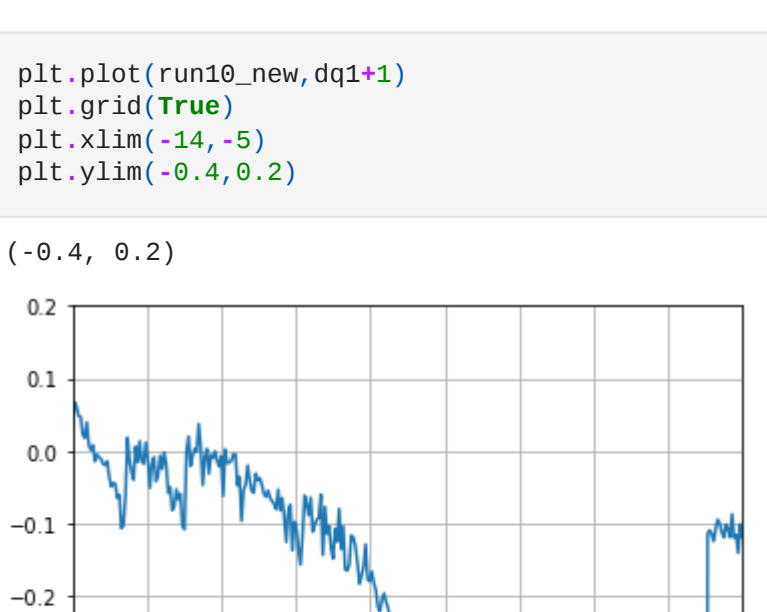


A scatter plot showing Temperature (C) on the y-axis (ranging from -20 to 20) versus Time (s) on the x-axis (ranging from 0 to 1400). The data points are blue dots, and a red line represents a linear fit. The plot is titled "Temperature Change Plot".

```
In [422.] run10_new = run[2]**-0.02690653735173463 + 16.12795492360233
dq3=dQ_dT((run[1]),-m,S(run10_new))
plt.plot(run10_new,dq3*1)
plt.grid(True)

In [426.] plt.plot(run10_new,dq3*1)
plt.grid(True)
plt.ylim(-14, 5)
plt.ylim(-0.4,0.2)

Out[426.] (-0.4, 0.2)
```



A plot showing dQ/dT (J/K) on the y-axis (ranging from -4 to 0.2) versus Temperature (C) on the x-axis (ranging from -14 to -5). The plot is titled "Thermogram in freezing of water".

```
In [454.] plt.plot(run10_new[550:1000],dq3[550:1000]*1.2)
plt.grid(True)

In [459.] integrate.simps(dq3[550:1000]*2,run10_new[550:1000])

Out[459.] 5.881373967586423
```

Run 4

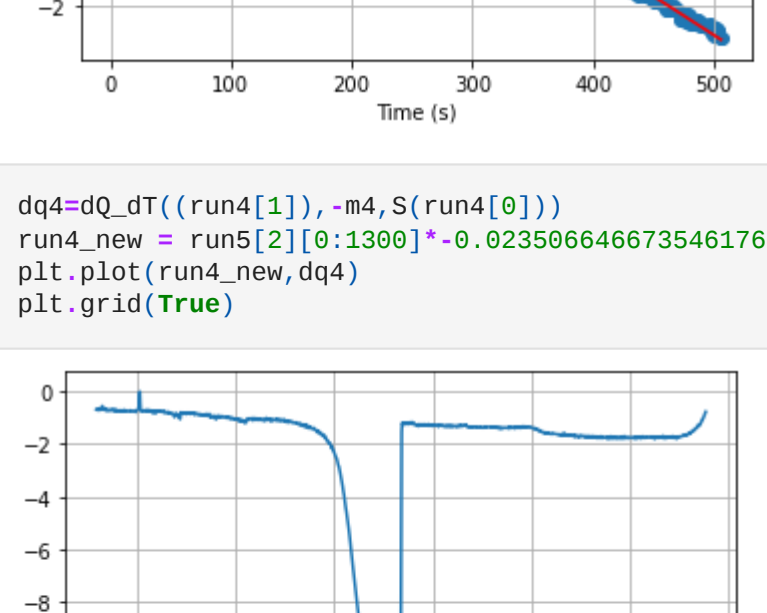
```
In [238.] run4= np.loadtxt("run8.txt")

In [488.] np.min(run4[1])

Out[488.] -0.02582960446349655

In [584.] plt.scatter(run4[2],run4[0])
m4,c4 = np.polyfit(run4[2], run4[0] [0:499] , 1)
plt.plot(run4[2],run4[2]**m4 + c4, "r")
m4,c4
plt.grid(True)
plt.xlabel("Time (s)")
plt.ylabel("Temperature (C)")
plt.title("Temperature Change Plot")

Out[584.] Text(0.5, 1.0, 'Temperature Change Plot')
```

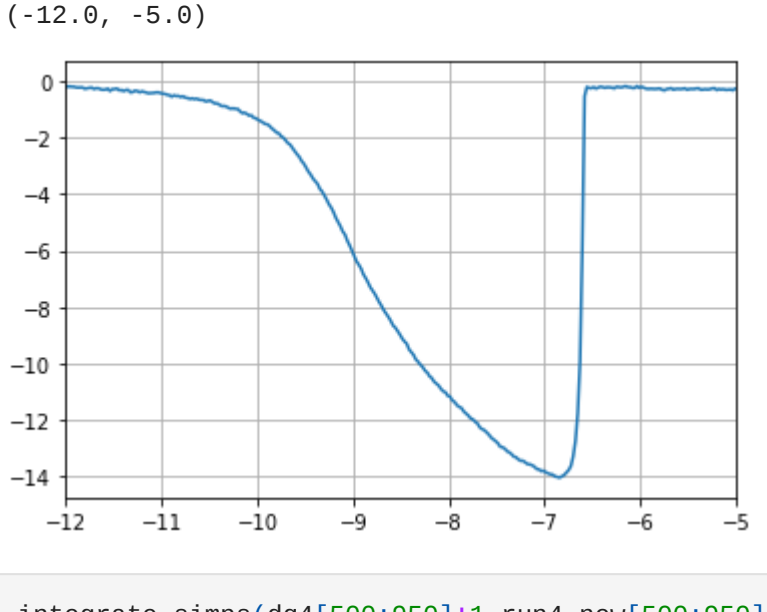


A scatter plot showing Temperature (C) on the y-axis (ranging from -2 to 8) versus Time (s) on the x-axis (ranging from 0 to 500). The data points are blue dots, and a red line represents a linear fit. The plot is titled "Temperature Change Plot".

```
In [483.] dq4=dQ_dT((run4[1]),-m4,S(run4[0]))
run4_new = run5[2][0:1300]**-0.023506646073546176 + 8.887879458599164
plt.plot(run4_new,dq4)
plt.grid(True)

In [280.] plt.plot(run4_new,dq4)
plt.grid(True)
plt.ylim(-14,0)
plt.ylim(-1.5,-.5)


Out[280.] (-1.5, -0.5)
```



A plot showing dQ/dT (J/K) on the y-axis (ranging from -14 to 0) versus Temperature (C) on the x-axis (ranging from -14 to -5). The plot is titled "Thermogram in freezing of water".

```
In [484.] plt.plot(run4_new[500:950],dq4[500:950]*1)
plt.grid(True)
plt.ylim(-12, 5)

Out[484.] (-12.0, -5.0)
```



A zoomed-in plot showing dQ/dT (J/K) on the y-axis (ranging from -14 to 0) versus Temperature (C) on the x-axis (ranging from -12 to -5). The plot is titled "Thermogram in freezing of water".

```
In [485.] integrate.simps(dq4[500:950]*1,run4_new[500:950])

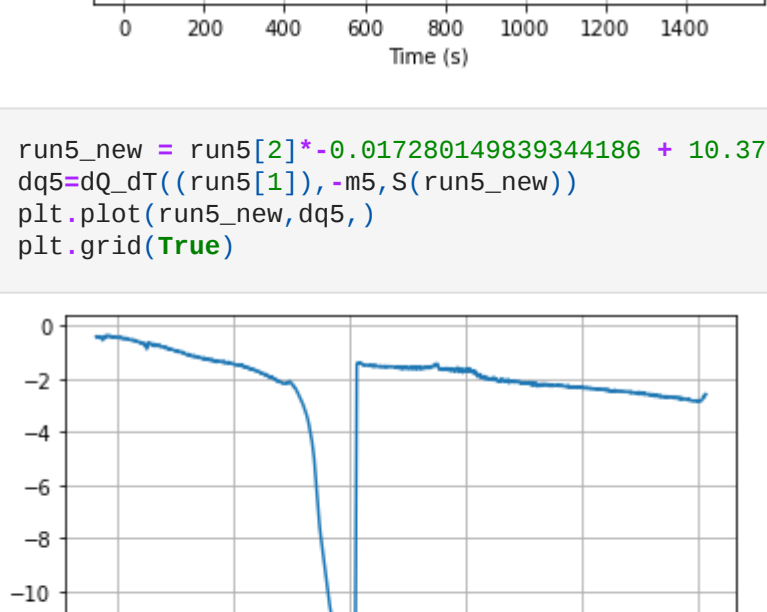
Out[485.] 32.39670581155432
```

Run 5

```
In [310.] run5= np.loadtxt("run10.txt")

In [585.] plt.scatter(run5[2],run5[0])
m5,c5 = np.polyfit(run5[2], run5[0] , 1)
plt.plot(run5[2],run5[2]**m5 + c5, "r")
m5,c5
plt.grid(True)
plt.xlabel("Time (s)")
plt.ylabel("Temperature (C)")
plt.title("Temperature Change Plot")

Out[585.] Text(0.5, 1.0, 'Temperature Change Plot')
```

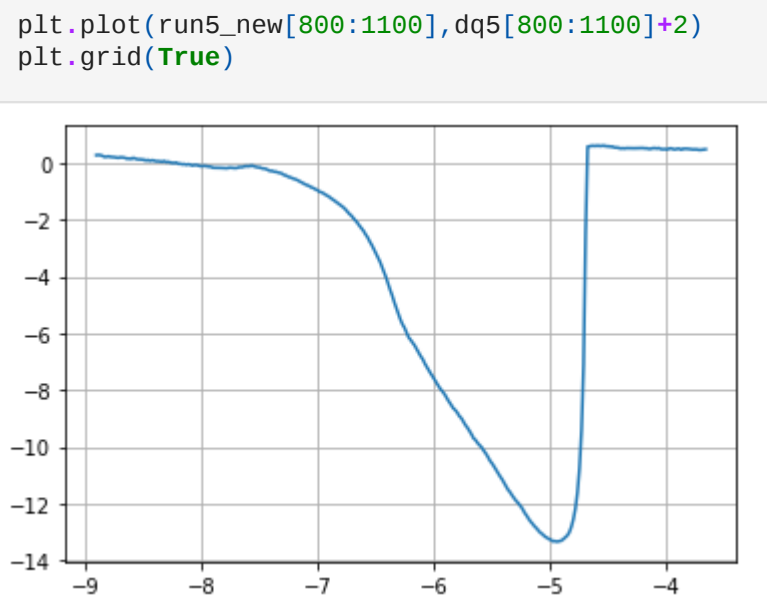


A scatter plot showing Temperature (C) on the y-axis (ranging from -15 to 10) versus Time (s) on the x-axis (ranging from 0 to 1400). The data points are blue dots, and a red line represents a linear fit. The plot is titled "Temperature Change Plot".

```
In [489.] run5_new = run5[2]**-0.017200149839344180 + 10.373377436502074
dq5=dQ_dT((run5[1]),-m5,S(run5_new))
plt.plot(run5_new,dq5,)
plt.grid(True)

In [464.] plt.plot(run5_new,dq5)
plt.grid(True)
plt.ylim(-20, 5)
plt.ylim(-2,0)

Out[464.] (-2.0, 0.0)
```



A plot showing dQ/dT (J/K) on the y-axis (ranging from -20 to 0.00) versus Temperature (C) on the x-axis (ranging from -20 to 5). The plot is titled "Thermogram in freezing of water".

```
In [368.] plt.plot(run5_new[800:1100],dq5[800:1100]*2)
plt.grid(True)

In [369.] integrate.simps(dq5[800:1100]*2,run4_new[800:1100])

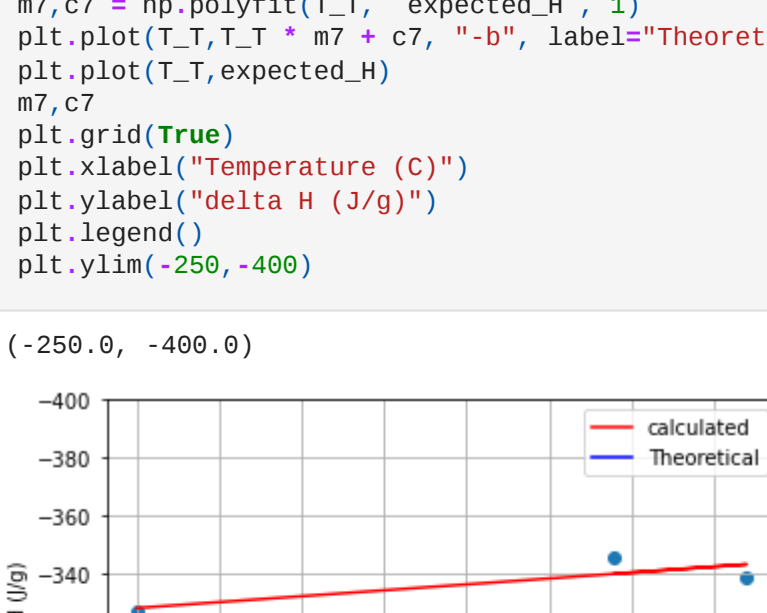
Out[369.] 24.07643360472807
```

```
In [571.] delta_H = [-327.050, 330.02, -345.00]
T_mean = np.array([-7.25, -5.4, -5.8 ])
np.mean(delta_H)
```

-337.25266666666664

```
In [574.] plt.scatter(T,T, delta_H)
expected_H=-336-2.17*(T-T)
m6,c6 = np.polyfit(T,T, delta_H , 1)
plt.plot(T,T,T-T*m6 + c6,"r",label= "calculated", )
m7,c7 = np.polyfit(T,T, expected_H , 1)
plt.plot(T,T,T**m7 + c7, "-b",label="Theoretical" )
m7,c7
plt.grid(True)
plt.xlabel("Temperature (C)")
plt.ylabel("delta_H (J/g)")
plt.legend()
plt.ylim(-250, -400)

Out[574.] (-250.0, -400.0)
```



A plot showing delta H (J/g) on the y-axis (ranging from -260 to -400) versus Temperature (C) on the x-axis (ranging from -7.25 to -5.50). The plot compares calculated (red line) and theoretical (blue line) values. The plot is titled "Comparison of calculated and theoretical delta H".

```
In [581.] c_p = np.array([2.72,2.04,1.07])
np.std(c_p)/np.size(c_p), np.std(expected_H)/np.size(expected_H), np.std(delta_H)/np.size(delta_H)

Out[581.] (0.12241399096992626, 0.5748869615137557, 2.5878536986143748)

In [579.] (c6-c7)/c6

Out[579.] 0.13236902487105684

In [577.] c6

Out[577.] -387.26141600703505

In [578.] c7

Out[578.] -335.99999999999983
```