

PHYC30300- Advanced Laboratory I



Modulus of Rigidity

Nathan Power

19311361

Abstract

This experiment investigates the temperature variation of modulus of rigidity and internal friction. The frequency response of metal rods of aluminium, steel and brass, clamped at one end and loaded at the other were analysed when forced to execute torsional oscillations at various different temperatures. The width of resonance was recorded and studied as a function of temperature. From this, the temperature variation of modulus of rigidity and internal friction for each sample rod was determined. In our observations, we were able to replicate the findings of Shanker[1] and F Horton[2] for the temperature variation of modulus of rigidity and internal friction.

Introduction

The effect of heat upon the modulus of torsional rigidity was first studied by Kupffer in 1848, who investigated the changes in rigidity by observations on the torsional vibrations of wires. Kupffer experimented on wires of iron, platinum, silver, copper, and gold, but the range of temperature was only the changing temperature of the room which never changed by more than 10°C. [2] In a later paper, experiments were carried out on wires of copper, steel, and brass at the temperature of the room and at the temperature of boiling water. The coefficient of diminution of the rigidity modulus was assumed to be constant between these temperatures, and was calculated from the periods of the torsional oscillations. Napier sky repeated Kupffer's experiments with wires of iron, brass, and silver, using a range of temperature of only a few degrees.[2]

The modulus of rigidity modulus was assumed to be constant between these temperatures, and was calculated from the periods of the torsional oscillations. Napier sky repeated Kupffer's experiments with wires of iron, brass, and silver, using a range of temperature of only a few degrees. Kohlrausch and Loomis experimented on wires of iron, copper, and brass at temperatures between 15°C and 100°C, and came to the conclusion that for these metals the rate of alteration of the rigidity modulus with the temperature was not constant but increased with the temperature.[2] It was found that the decrease of rigidity per degree rise of temperature increased with this temperature.

Theory

When a force is applied on a solid object, its shape and size may be distorted or changed. Solids which regain their initial shape and size when the applied force is removed are called elastic objects. Elasticity is the property of material which allows it to regain its initial shape and size. Modulus of rigidity is the measure of how rigid a material is to oppose the force which attempts to deform the shape of the object. [3]

Let's consider a cylinder of mass, m , and radius, R , which is suspended vertically by a wire. The thin wire has negligible mass compare to the cylinder and has radius r and the modulus

of rigidity, G , If the cylinder is rotated through a small angle about the z axis and then released, due to the rigidity modulus of the wire, the system will also oscillate rotationally due to Hooke's law.[3] Let T be the time period of the circular oscillation. There is a relationship among G , T and the geometrical parameters of the wire. If L is the length of the wire, this relation is given by the following equation:

$$G = \frac{8\pi IL}{T^2 r^4} \quad \text{Eq.(1)}$$

Where, I is the moment of inertia of the cylinder about the z axis. If the mass of the cylinder is m and its radius is R , then its moment of inertia about the z axis is;

$$I = \frac{1}{2} m R^2 \quad \text{Eq.(2)}$$

In our experiment the effective value of the inertia is that of the auxiliary body and the restoring torque is mainly provided by the specimen rod.[1] Thus the torsion constant may be expressed as;

$$C = \frac{\pi G r^4}{2L} \quad \text{Eq.(3)}$$

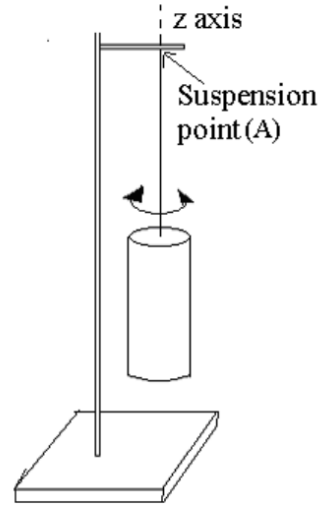


Fig.1 shows a cylinder suspended by a thin wire [3]

For the case of low damping, C becomes $C = I(\omega_0)^2$, where ω_0 is the frequency at which our peak occurs in our frequency response curve. Solving Eq.(3) for G , we find; [1]

$$G = \frac{2IL\omega_0^2}{\pi r^4} \quad \text{Eq.(4)}$$

Internal friction is an internal force that resists the movement between the particles of a material. This internal movement can be due to either external forces or change in temperature and deformation. The internal friction force acts against the shear stress between the particles and tries to keep them stationary and in place. [4]

Full width of the response curve at half the maximum amplitude is given by $\Delta\omega$.

Therefore, a measure of internal friction, is expressed as;

$$Q^{-1} = \frac{\Delta\omega}{\sqrt{3}\omega_0} \quad \text{Eq.(5)}$$

Experimental Procedure

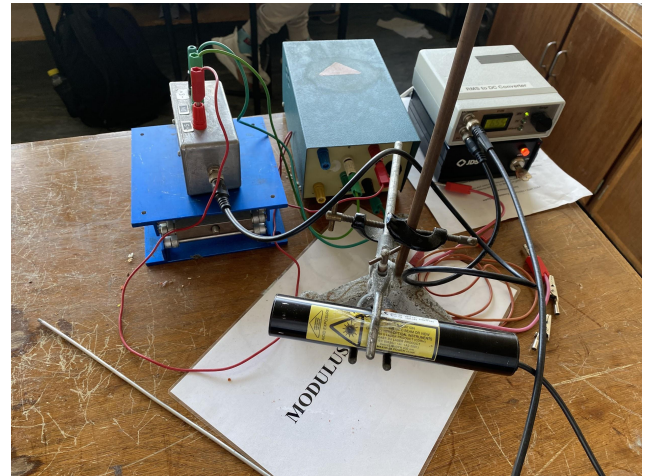


Fig.2a and Fig.2b shows the apparatus used in this experiment

Figure 2a and 2b show the apparatus used in this experiment. A metal rod sample of aluminium, brass and steel of length 0.3 m and diameter 3 mm were, each in turn, vertically clamped in a support. The bottom end of the sample was attached to an auxiliary body with a small cylindrical magnet attached to its lower end. This auxiliary body was used to control the resonant frequency of the system. A small concave mirror was also attached to the auxiliary body. This mirror was used to reflect the laser onto the position sensing detector. The purpose of the position sensing detector was to measure the amplitude of oscillations of each of our sample rods. A Helmholtz coil is used with its axis perpendicular to the axis of the magnet in order to apply an external torque on the system. The coils were fed a signal from our function generator, the National Instruments multi-function USB-6008 module to produce torsional oscillations in the system. This DC signal was first converted to AC and fed into the coils. The module was used as a voltage controlled oscillator to generate sinusoidal waveforms in the frequency range of >10 Hz to around 300 Hz. The image formed on the position sensing detector was then fed back into the National Instruments USB-6008 to be recorded. Each data point recorded was a mean of 500 observations. The sample rod was surrounded by a heater coil to heat and maintain it at a measured temperature.

Results and Discussion

Frequency Response Curves for our Samples at various temperatures

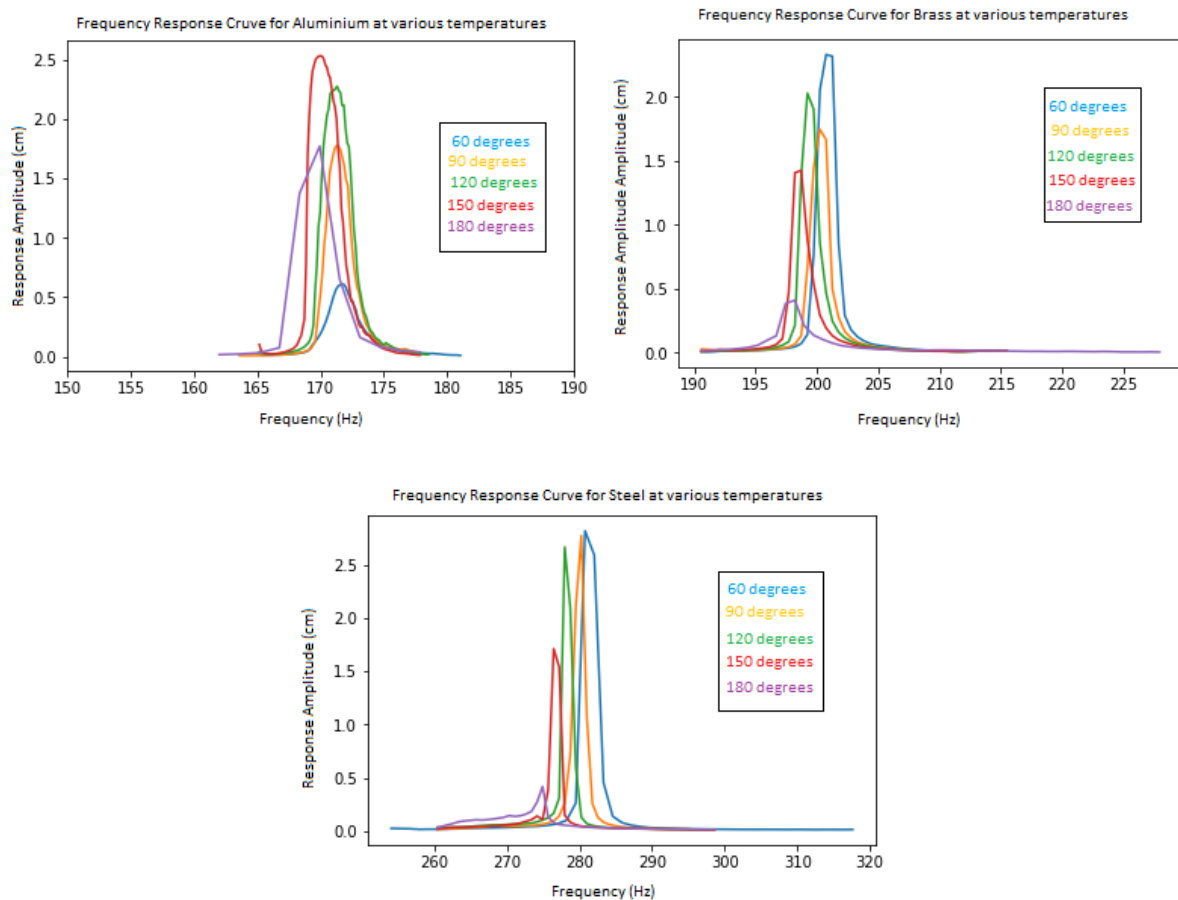


Fig.3a, Fig.3b and Fig3.c show the frequency response curves for aluminium, brass and steel at various temperatures

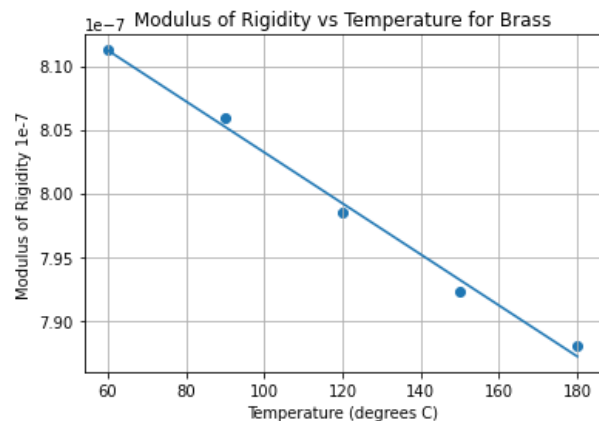
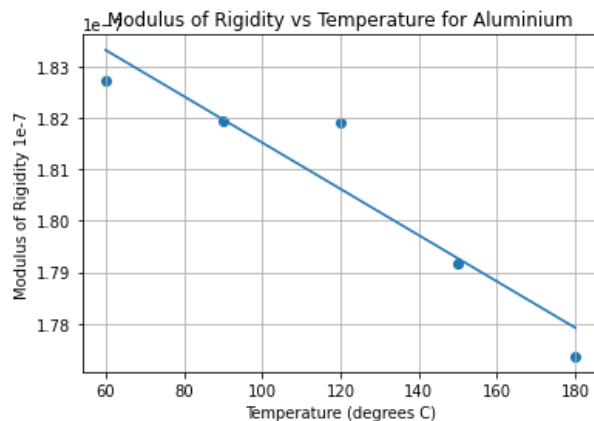
Expected results for our frequency response curves for our three samples would be that as the temperatures increase the peaks are shifted left and that the full width of the peaks at half the maximum would also increase with temperature. This is consistent with our observed results. However, we would also expect the amplitudes of the peaks to decrease as the temperature increases. This was not observed in our findings. The cause for this lies with the varying distance between the mirror and the position sensing detector. As the amplitudes decreased the detector struggled to pick up strong measurements for the amplitudes leaving the frequency curves produced to be inaccurate. To counteract this, we decided to decrease the distance between the detector and the mirror. For other measurements, the frequency curve would be produced with a flat top instead of a sharp peak. This was occurring as the image produced by the mirror would be wider than the screen on the detector meaning the detector could not make any more readings until the image narrowed one again. Due to the inconsistent varying of this distance, not all of the peak amplitudes decrease with increasing temperature.

Table 1. Aluminium					
Temp (degrees C)	60	90	120	150	180
ω_0 (Hz)	171.675	171.300	171.294	169.99	169.134
$\Delta\omega$ (Hz)	2.22	,2.52	2.69	2.71	,3.44

Table 2. Brass					
Temp (degrees C)	60	90	120	150	180
ω_0 (Hz)	200.786	200.129	199.208	198.419	197.891
$\Delta\omega$ (Hz)	1.78	1.428	1.607	2.14	2.368

Table 3. Steel					
Temp (degrees C)	60	90	120	150	180
ω_0 (Hz)	280.778	280.270	277.983	276.459	275.459
$\Delta\omega$ (Hz)	2.66	1.566	1.61	1.76	,2.1

Modulus of Rigidity for our Samples at various temperatures



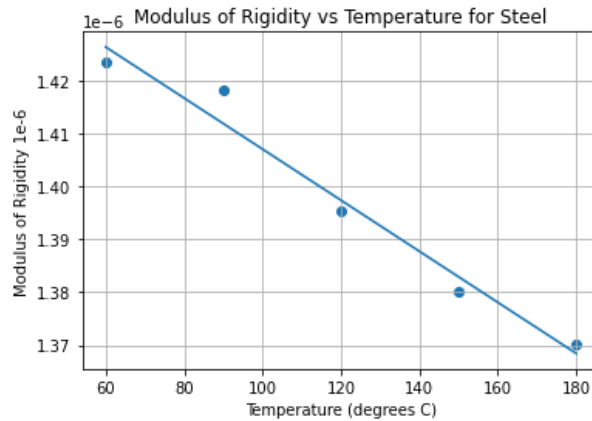
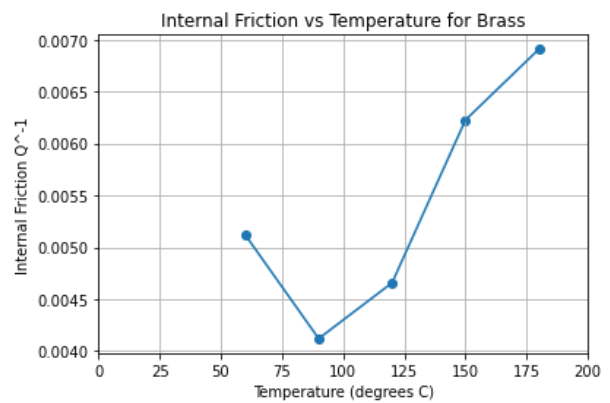
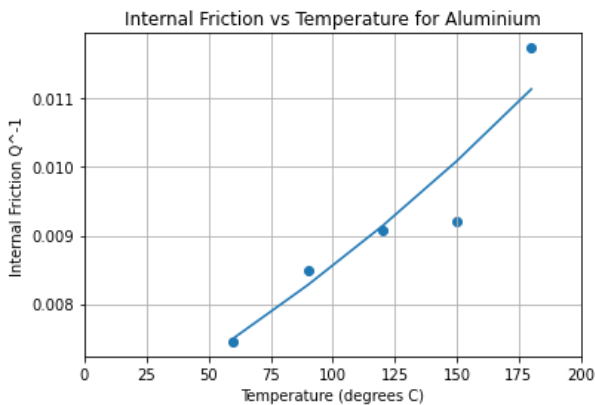


Fig.4a, Fig.4b and Fig.4.c show the modulus of rigidity vs temperature for aluminium, brass and steel

Using Eq.(4), Eq.(3) and our observed measurements from the previous section, the modulus of rigidity was found for each of the samples at various temperatures. In all the materials examined, the modulus of rigidity at one temperature is not constant, but increases as time goes on. It was found that the decrease of rigidity vs temperature increased with this temperature with each of our three sample rods. However, when calculating the moment of inertia for each of our samples, an error was made. Instead of calculating the moment of inertia for the auxiliary body, we calculated the moment of inertia for each sample rod with the auxiliary body detached. This would have skewed the results we received as in our experiment the effective value of the inertia is that of the auxiliary body.

Internal Friction for our Samples at various temperatures



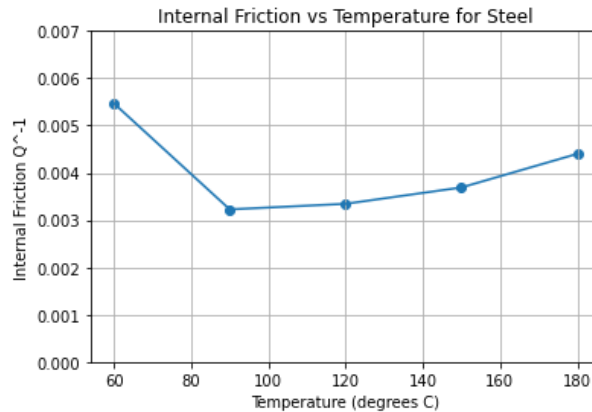


Fig.5a, Fig.5b and Fig5.c show the modulus of rigidity vs temperature for aluminium, brass and steel

Using Eq.(5), Eq.(3) and our observed measurements from the previous sections, the modulus of rigidity was found for each of the samples at various temperatures.

The internal viscosity of all the metals examined, aluminium, brass and steel increase with the temperature as also found by *F Horton* [2] and also consistent with the metals sampled by *Shanker*[1]. This increase varies very much between the different metals, being greatest with aluminium and least with steel. However, there is a slight decrease which precedes the increase in the case of steel and brass. If I was to repeat this experiment, I would take more measurements in the lower ranges and higher ranges of temperatures to provide more context to the increasing slopes we have found for our samples. This decrease in the internal viscosity of steel as was found to be consistent with *F Horton* [2]

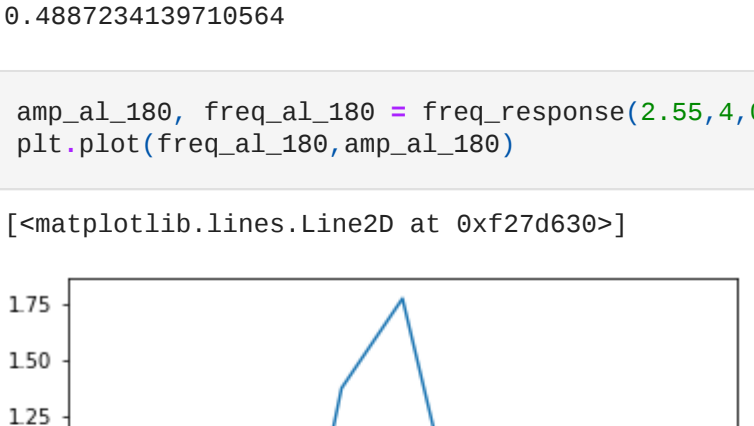
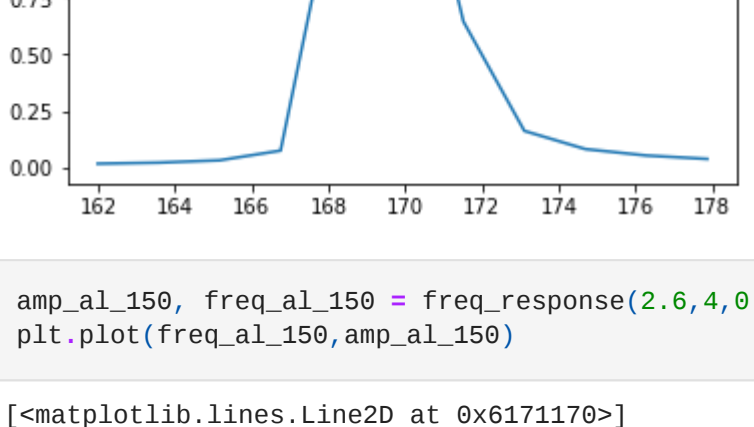
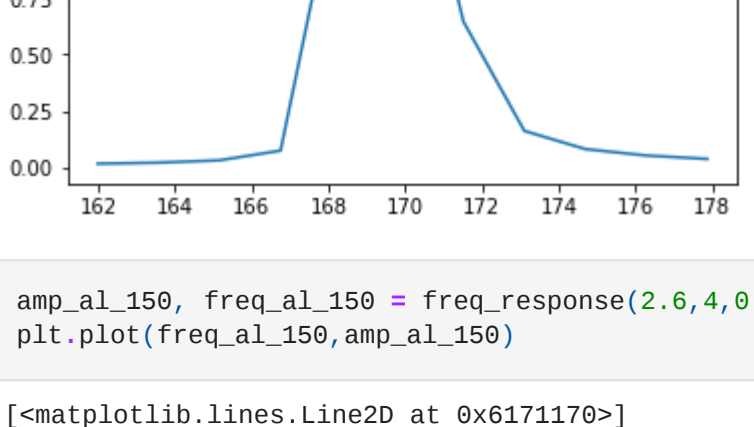
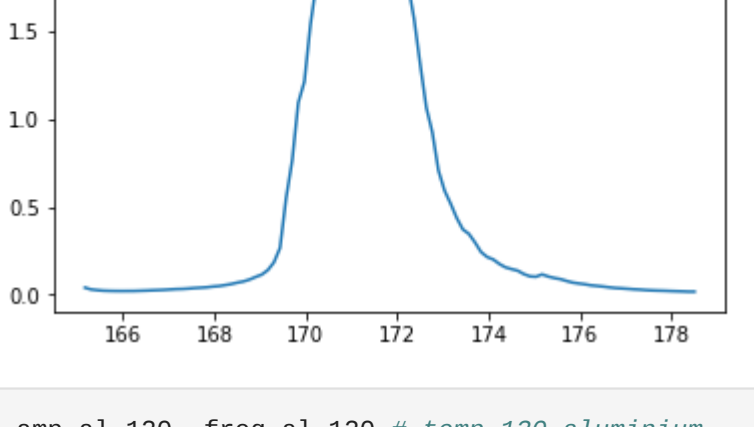
Conclusion

From our results, it can be clearly seen that the majority of our data analysis agrees with the expected results.[2][1] We can clearly see that we were able to replicate the ideas of *Shanker*[1] and *F Horton*[2] in our results for the frequency response curves, modulus of rigidity and internal friction of our samples. However, our conducting of this experiment was not without error. If I was to conduct this experiment again, I would make sure to calculate the moment of inertia correctly for each of our samples and also to factor in the distances at which I vary between the mirror and detector when carrying out the frequency response curves. I would also change the amount of points taken during this stage. I would have preferred to take measurements at more temperature points at both the lower and upper range of temperatures, which was not possible due to time constraints. This way I would be able to put the curves I acquired during this experiment into full context. Although we did not succeed in all areas in this experiment, I believe that it is important that undergraduates develop a strong understanding of the temperature variation of modulus of rigidity and internal friction and how it can be observed in our daily lives such as in the construction sector.

References

- [1]Shanker, G., Gupta, V.K., Saraf, B. and Sharma, N.K., 1985. Temperature variation of modulus of rigidity and internal friction: An experiment with torsional oscillator. *American Journal of Physics*, 53(12), pp.1192-1195.
- [2] Frank Horton D.Sc., B.A, St. John's College, Cambridge; 1851 The Effects of Changes of Temperature on the Modulus of Torsional Rigidity of Metal Wires. Exhibition Research Scholar of the University of Birmingham. Communicated by Professor J. J. Thomson, F.R.S
- [3]Mohammed Riyadh,2019, BRAC University, Determination of the modulus of rigidity of the element of wire by the method of oscillation
- [4] What is Internal Friction? - Definition, Angle & Coefficient. (2016, November 3). Retrieved from <https://study.com/academy/lesson/what-is-internal-friction-definition-angle-coefficient.html>.

Appendix A Data Acquisition

In []:	<pre>from pydaqmx_helper.dac import DAC import numpy as np import matplotlib.pyplot as plt import time</pre>
In [52]:	<pre>myDAC = DAC(1) myDAC.writeVoltage(.1)</pre>
In [3]:	<pre>#calibration V_in = np.array([0.2,0.3,0.5,0.8,0.9,1.1,2.1,5.1,8.2,2.2]) F_HZ = np.array([13.32,19.77,32.5,51.45, 57.8, 64.16, 77.62, 96.08, 114.98, 127.655,140.447]) plt.scatter(V_input, F_HZ) m,c = np.polyfit(V_input, F_HZ, 1) plt.plot(V_input, V_input*m + c, "r") plt.grid(True) plt.xlabel("Input Voltage (V)") plt.ylabel("Frequency (Hz)") plt.title("Calibration Plot") print(m);</pre>
	<p>63.52465774980332</p> 
In [4]:	<pre>from pydaqmx_helper.adc import ADC myADC = ADC() myADC.addChannels([0])</pre>
	Activated Channel 0
In [5]:	<pre>data = myADC.sampleVoltages(500,500) #takes 500 data points print(np.mean(data[0])) #finds the average of these 500 points 0.016487260254062603</pre>
In [5]:	<pre>def freq_response(start_v, delay ,v_step, v_stop): amplitude=np.array([]) frequency = np.array([]) while start_v < v_stop: #runs from start voltage to desired end voltage myDAC.writeVoltage(start_v) time.sleep(delay) #allows the detector to stabilise before taking measurements data = myADC.sampleVoltages(1000,500) amplitude = np.append(amplitude,np.mean(data[0])) freq_in_hz = start_v*63.52465774980332 #converts voltage to frequency frequency = np.append(frequency, freq_in_hz) start_v = v_step #starts loop again at next step voltage myDAC.writeVoltage(0.2) #takes voltage back down when all measurements have been taken return amplitude, frequency</pre>
In [14]:	<pre>amp_al_room, freq_al_room = freq_response(2.55,4,0.004,2.95) #obtains values and plots them plt.plot(freq_al_room,amp_al_room)</pre>
Out[14]:	<p>[<matplotlib.lines.Line2D at 0x257350d>]</p> 
In [15]:	<pre>amp_al_room, freq_al_room = room temp aluminium np.savetxt("amp_al_room-real", amp_al_room, delimiter = ",") np.savetxt("freq_al_room-real", freq_al_room, delimiter = ",")</pre>
In [62]:	<pre>np.max(amp_al_room)</pre>
Out[62]:	0.0887234139710564
In [50]:	<pre>amp_al_180, freq_al_180 = freq_response(2.55,4,0.025,2.8) #obtains values and plots them plt.plot(freq_al_180,amp_al_180)</pre>
Out[50]:	<p>[<matplotlib.lines.Line2D at 0xf27d630>]</p> 
In [36]:	<pre>amp_al_150, freq_al_150 = freq_response(2.6,4,0.002,2.8) #obtains values and plots them plt.plot(freq_al_150,amp_al_150)</pre>
Out[36]:	<p>[<matplotlib.lines.Line2D at 0x6171170>]</p> 
In [37]:	<pre>amp_al_150, freq_al_150 = temp 150 aluminium np.savetxt("amp_al_150-real", amp_al_150, delimiter = ",") np.savetxt("freq_al_150-real", freq_al_150, delimiter = ",")</pre>
In [29]:	<pre>amp_al_120, freq_al_120 = freq_response(2.6,4,0.0021,2.81) #obtains values and plots them plt.plot(freq_al_120,amp_al_120)</pre>
Out[29]:	<p>[<matplotlib.lines.Line2D at 0x4ad8210>]</p> 
In [30]:	<pre>amp_al_120, freq_al_120 = temp 120 aluminium np.savetxt("amp_al_120-real", amp_al_120, delimiter = ",") np.savetxt("freq_al_120-real", freq_al_120, delimiter = ",")</pre>
In [46]:	<pre>amp_al_180, freq_al_180 = temp 180 aluminium np.savetxt("amp_al_180-real", amp_al_180, delimiter = ",") np.savetxt("freq_al_180-real", freq_al_180, delimiter = ",")</pre>
In [22]:	<pre>amp_al_90, freq_al_90 = freq_response(2.575,4,0.00225,2.8) #obtains values and plots them plt.plot(freq_al_90,amp_al_90)</pre>
Out[22]:	<p>[<matplotlib.lines.Line2D at 0x26f090>]</p> 
In [23]:	<pre>amp_al_90, freq_al_90 = temp 90 aluminium np.savetxt("amp_al_90-real", amp_al_90, delimiter = ",") np.savetxt("freq_al_90-real", freq_al_90, delimiter = ",")</pre>
In [19]:	<pre>amp_al_60, freq_al_60 = freq_response(2.6,4,0.0025,2.85) #obtains values and plots them plt.plot(freq_al_60,amp_al_60)</pre>
Out[19]:	<p>[<matplotlib.lines.Line2D at 0xf26b1b0>]</p> 
In [20]:	<pre>amp_al_60, freq_al_60 = temp 60 aluminium np.savetxt("amp_al_60-real", amp_al_60, delimiter = ",") np.savetxt("freq_al_60-real", freq_al_60, delimiter = ",")</pre>
In [89]:	<pre>plt.plot(freq_al_60,amp_al_60) plt.plot(freq_al_90,amp_al_90) plt.plot(freq_al_120,amp_al_120) plt.plot(freq_al_150,amp_al_150) plt.plot(freq_al_180,amp_al_180) plt.xlim(150,190)</pre>
Out[89]:	<p>(150, 190)</p> 
<h2>Brass</h2>	
In [58]:	<pre>amp_br_60, freq_br_60 = freq_response(3,4,0.008,3.4) #obtains values and plots them plt.plot(freq_br_60,amp_br_60)</pre>
Out[58]:	<p>[<matplotlib.lines.Line2D at 0xf3bc230>]</p> 
In [60]:	<pre>amp_br_60, freq_br_60 = temp 60 brass np.savetxt("amp_br_60-real", amp_br_60, delimiter = ",") np.savetxt("freq_br_60-real", freq_br_60, delimiter = ",")</pre>
In [62]:	<pre>amp_br_90, freq_br_90 = freq_response(3,4,0.008,3.4) #obtains values and plots them plt.plot(freq_br_90,amp_br_90)</pre>
Out[62]:	<p>[<matplotlib.lines.Line2D at 0xf43a7f0>]</p> 
In [63]:	<pre>amp_br_90, freq_br_90 = temp 90 brass np.savetxt("amp_br_90-real", amp_al_90, delimiter = ",") np.savetxt("freq_br_90-real", freq_al_90, delimiter = ",")</pre>
In [65]:	<pre>amp_br_120, freq_br_120 = freq_response(3,4,0.008,3.4) #obtains values and plots them plt.plot(freq_br_120,amp_br_120)</pre>
Out[65]:	<p>[<matplotlib.lines.Line2D at 0xf2b44b0>]</p> 
In [67]:	<pre>amp_br_120, freq_br_120 = temp 120 brass np.savetxt("amp_br_120-real", amp_al_120, delimiter = ",") np.savetxt("freq_br_120-real", freq_al_120, delimiter = ",")</pre>
In [68]:	<pre>amp_br_150, freq_br_150 = freq_response(3,4,0.008,3.4) #obtains values and plots them plt.plot(freq_br_150,amp_br_150)</pre>
Out[68]:	<p>[<matplotlib.lines.Line2D at 0xf236130>]</p> 
In [70]:	<pre>amp_br_150, freq_br_150 = temp 150 brass np.savetxt("amp_br_150-real", amp_al_150, delimiter = ",") np.savetxt("freq_br_150-real", freq_al_150, delimiter = ",")</pre>
In [79]:	<pre>amp_br_180, freq_br_180 = freq_response(3,4,0.012,3.6) #obtains values and plots them plt.plot(freq_br_180,amp_br_180)</pre>
Out[79]:	<p>[<matplotlib.lines.Line2D at 0xf7cc270>]</p> 
In [83]:	<pre>amp_br_180, freq_br_180 = temp 180 brass np.savetxt("amp_br_180-real", amp_al_180, delimiter = ",") np.savetxt("freq_br_180-real", freq_al_180, delimiter = ",")</pre>
In [82]:	<pre>plt.plot(freq_br_60,amp_br_60) plt.plot(freq_br_90,amp_br_90) plt.plot(freq_br_120,amp_br_120) plt.plot(freq_br_150,amp_br_150) plt.plot(freq_br_180,amp_br_180)</pre>
Out[82]:	<p>[<matplotlib.lines.Line2D at 0xf894370>]</p> 
<h2>Steel</h2>	
In [100]:	<pre>amp_st_60, freq_st_60 = freq_response(4,4,0.02,5) #obtains values and plots them plt.plot(freq_st_60,amp_st_60)</pre>
Out[100]:	<p>[<matplotlib.lines.Line2D at 0xf6b1a30>]</p> 
In [101]:	<pre>amp_st_60, freq_st_60 = temp 60 steel np.savetxt("amp_st_60-real", amp_st_60, delimiter = ",") np.savetxt("freq_st_60-real", freq_st_60, delimiter = ",")</pre>
In [102]:	<pre>amp_st_90, freq_st_90 = freq_response(4,1,4,0.012,4.7) #obtains values and plots them plt.plot(freq_st_90,amp_st_90)</pre>
Out[102]:	<p>[<matplotlib.lines.Line2D at 0xfbf94910>]</p> 
In [103]:	<pre>amp_st_90, freq_st_90 = temp 90 steel np.savetxt("amp_st_90-real", amp_st_90, delimiter = ",") np.savetxt("freq_st_90-real", freq_st_90, delimiter = ",")</pre>
In [106]:	<pre>amp_st_120, freq_st_120 = freq_response(4,1,4,0.012,4.7) #obtains values and plots them plt.plot(freq_st_120,amp_st_120)</pre>
Out[106]:	<p>[<matplotlib.lines.Line2D at 0xfcd4330>]</p> 
In [111]:	<pre>amp_st_150, freq_st_150 = temp 150 steel np.savetxt("amp_st_150-real", amp_st_150, delimiter = ",") np.savetxt("freq_st_150-real", freq_st_150, delimiter = ",")</pre>
In [114]:	<pre>amp_st_180, freq_st_180 = freq_response(4,1,4,0.012,4.7) #obtains values and plots them plt.plot(freq_st_180,amp_st_180)</pre>
Out[114]:	<p>[<matplotlib.lines.Line2D at 0xfda0a30>]</p> 
In [115]:	<pre>amp_st_180, freq_st_180 = temp 180 steel np.savetxt("amp_st_180-real", amp_st_180, delimiter = ",") np.savetxt("freq_st_180-real", freq_st_180, delimiter = ",")</pre>
In [116]:	<pre>plt.plot(freq_st_60,amp_st_60) plt.plot(freq_st_90,amp_st_90) plt.plot(freq_st_120,amp_st_120) plt.plot(freq_st_150,amp_st_150) plt.plot(freq_st_180,amp_st_180)</pre>
Out[116]:	<p>[<matplotlib.lines.Line2D at 0xfdef330>]</p> 

Appendix B Data Analysis

```
In [ ]: import numpy as np
import math
import matplotlib.pyplot as plt
from scipy.signal import chirp, find_peaks, peak_widths
```

Import Data

```
In [76]: amp_al_120r = np.loadtxt("amp_al_120-real")
freq_al_120r = np.loadtxt("freq_al_120-real")
amp_al_90r = np.loadtxt("amp_al_90-real")
freq_al_90r = np.loadtxt("freq_al_90-real")
amp_al_60r = np.loadtxt("amp_al_60-real")
freq_al_60r = np.loadtxt("freq_al_-real60")
amp_al_180r = np.loadtxt("amp_al_180-real")
freq_al_180r = np.loadtxt("freq_al_180-real")
amp_al_150r = np.loadtxt("amp_al_150-real")
freq_al_150r = np.loadtxt("freq_al_150-real")
```

```
In [77]: amp_br_120r = np.loadtxt("amp_br_120-real")
freq_br_120r = np.loadtxt("freq_br_120-real")
amp_br_90r = np.loadtxt("amp_br_90-real")
freq_br_90r = np.loadtxt("freq_br_90-real")
amp_br_60r = np.loadtxt("amp_br_60-real")
freq_br_60r = np.loadtxt("freq_br_60-real")
amp_br_180r = np.loadtxt("amp_br_180-real")
freq_br_180r = np.loadtxt("freq_br_180-real")
amp_br_150r = np.loadtxt("amp_br_150-real")
freq_br_150r = np.loadtxt("freq_br_150-real")
```

```
In [260... amp_st_120r = np.loadtxt("amp_st_120-real")
freq_st_120r = np.loadtxt("freq_st_120-real")
amp_st_90r = np.loadtxt("amp_st_90-real")
freq_st_90r = np.loadtxt("freq_st_90-real")
amp_st_60r = np.loadtxt("amp_st_60-real")
freq_st_60r = np.loadtxt("freq_st_60-real")
amp_st_180r = np.loadtxt("amp_st_180-real")
freq_st_180r = np.loadtxt("freq_st_180-real")
amp_st_150r = np.loadtxt("amp_st_150-real")
freq_st_150r = np.loadtxt("freq_st_150-real")

temp = np.array([60,90,120,150,180])
```

```
In [5]: def peak(x): #function finds peak amplitude
return np.max(x)
peak(amp_al_180r)
```

```
Out[5]: 2.2930795346491233
```

```
In [179... def peak_freq(x,y): #function that finds peak frequency
for i in x:
```

Loading [MathJax]/extensions/Safe.js list()


```

i = np.max(x)
i = x.index(i)
return (y[i])

```

```

In [7]: def peaks(x):
        return find_peaks(x)

```

```

In [8]: def delta_W(x):
        peaks, _ = find_peaks(x)
        results_half = peak_widths(x, peaks, rel_height=0.5)
        return results_half[0]

```

```

In [189... W_0_al_180 = peak_freq(amp_al_180r, freq_al_180r) #aluminium peak frequencies
W_0_al_150 = peak_freq(amp_al_150r, freq_al_150r)
W_0_al_120 = peak_freq(amp_al_120r, freq_al_120r)
W_0_al_90 = peak_freq(amp_al_90r, freq_al_90r)
W_0_al_60 = peak_freq(amp_al_60r, freq_al_60r)

w_0_al = np.array([171.675, 171.300, 171.294, 169.99, 169.134])

```

```

In [190... W_0_br_180 = peak_freq(amp_br_180r, freq_br_180r) #brass peak frequenies
W_0_br_150 = peak_freq(amp_br_150r, freq_br_150r)
W_0_br_120 = peak_freq(amp_br_120r, freq_br_120r)
W_0_br_90 = peak_freq(amp_br_90r, freq_br_90r)
W_0_br_60 = peak_freq(amp_br_60r, freq_br_60r)

w_0_br = np.array([200.786, 200.129, 199.208, 198.419, 197.891])

```

```

In [335... W_0_st_180 = peak_freq(amp_st_180r, freq_st_180r) #steel peak frequenies
W_0_st_150 = peak_freq(amp_st_150r, freq_st_150r)
W_0_st_120 = peak_freq(amp_st_120r, freq_st_120r)
W_0_st_90 = peak_freq(amp_st_90r, freq_st_90r)
W_0_st_60 = peak_freq(amp_st_60r, freq_st_60r)

w_0_st = np.array([280.778, 280.270, 277.983, 276.459, 275.459])

```

```

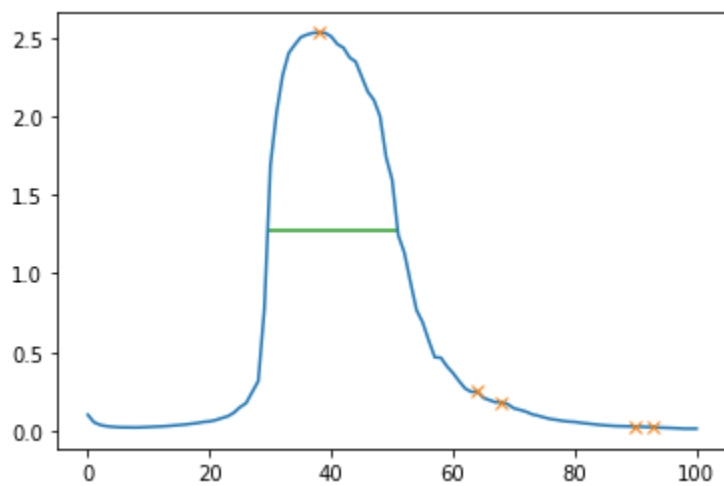
In [158... #finding the FWHH for each of the peaks for each sample
x = amp_al_150r

axis_ratio = (np.max(freq_al_150r) - np.min(freq_al_150r))/100
# to adjust the x axis to the actual difference between each of 100 frequency points take

peaks, _ = find_peaks(x)
results_half = peak_widths(x, peaks, rel_height=0.5)
print(results_half[0]*axis_ratio) # widths
plt.plot(x)
plt.plot(peaks, x[peaks], "x")
plt.hlines(*results_half[1:], color="C2")
plt.show()

```

```
[2.71332079 0.06745946 0.07545805 0.07931331 0.0781681 ]
```



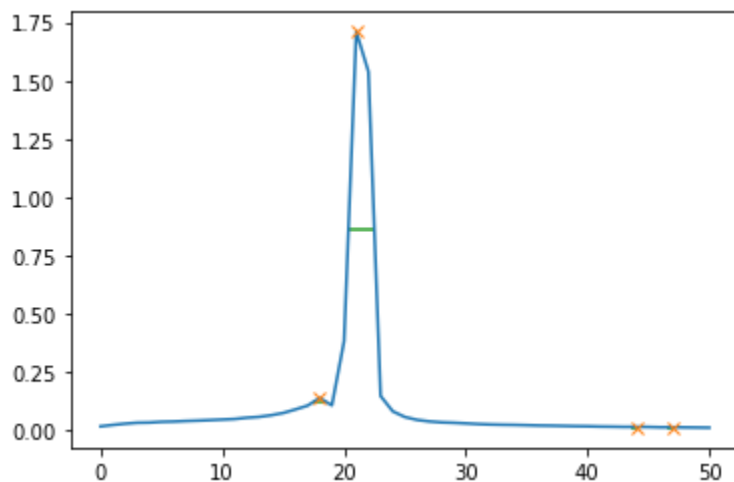
```
In [165... deltaW_al = np.array([2.215,2.52,2.69,2.713,3.44]) #results for aluminium
```

```
In [185... x = amp_st_120r

axis_ratio = (np.max(freq_st_150r) - np.min(freq_st_150r))/50
# to adjust the x axis to the actual difference between each of 50 frequency points for s

peaks, _ = find_peaks(x)
results_half = peak_widths(x, peaks, rel_height=0.5)
print(results_half[0]*axis_ratio) # widths
plt.plot(x)
plt.plot(peaks, x[peaks], "x")
plt.hlines(*results_half[1:], color="C2")
plt.show()
```

```
[0.74808759 1.61802235 0.49468138 0.52021544]
```



```
In [341... deltaW_br = np.array([1.78,1.428,1.607,2.14,2.368])
deltaW_st = np.array([2.66,1.566,1.61,1.767,2.1])
```

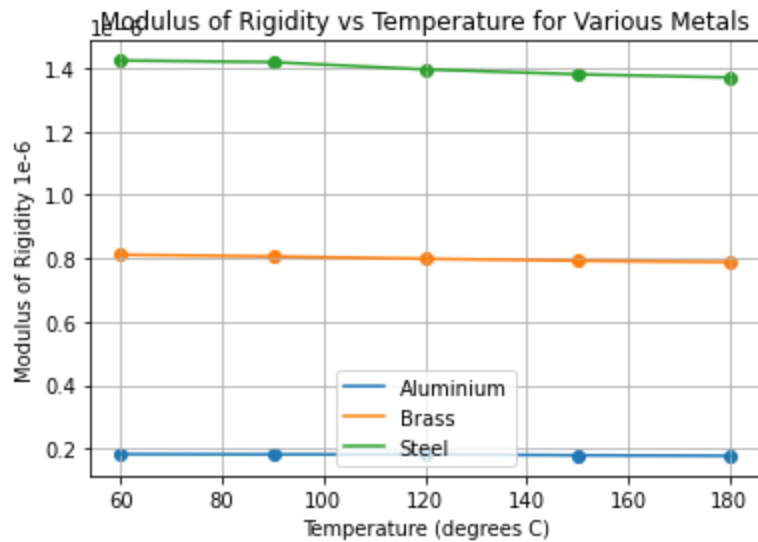
```
In [310... #moment of inertia for our sample rods
I_al = (5.7) * ((1.5)**2) /2
I_br = (18.5) * ((1.5)**2) /2
I_st = (16.6) * ((1.5)**2) /2
```

```
In [337... rigidity_al = 2*I_al*0.3*(w_0_al**2)/np.pi*((1.5/1000)**4) # Internal Friction vs Tempera
rigidity_br = 2*I_br*0.3*(w_0_br**2)/np.pi*((1.5/1000)**4)
```

```

rigidity_st = 2*I_st*0.3*(w_0_st**2)/np.pi*((1.5/1000)**4)
plt.scatter(temp,rigidity_al)
plt.scatter(temp,rigidity_br)
plt.scatter(temp,rigidity_st)
plt.plot(temp,rigidity_al)
plt.plot(temp,rigidity_br)
plt.plot(temp,rigidity_st)
plt.grid(True)
plt.xlabel("Temperature (degrees C)")
plt.ylabel("Modulus of Rigidity 1e-6")
plt.title("Modulus of Rigidity vs Temperature for Various Metals")
plt.legend(["Aluminium","Brass","Steel"]);

```

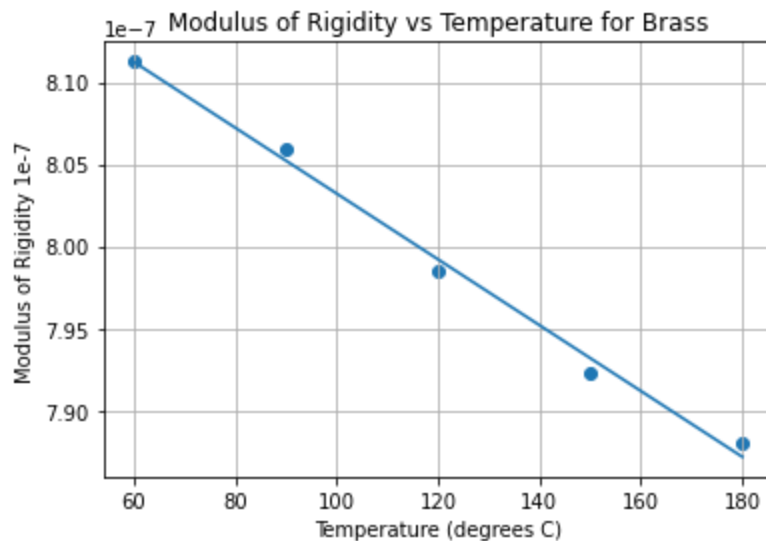


In [327...

```

plt.scatter(temp,rigidity_br) # Internal Friction vs Temperature for Brass
m, c = np.polyfit(temp,rigidity_br, 1)
plt.plot(temp, m*temp+c)
plt.grid(True)
plt.xlabel("Temperature (degrees C)")
plt.ylabel("Modulus of Rigidity 1e-7")
plt.title("Modulus of Rigidity vs Temperature for Brass");

```



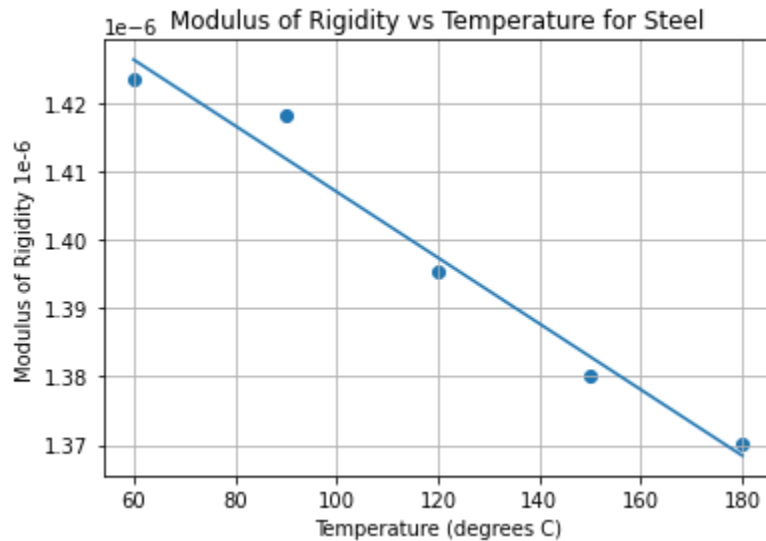
In [326...

```

plt.scatter(temp,rigidity_st) #Internal Friction vs Temperature for Steel
m, c = np.polyfit(temp,rigidity_st, 1)
plt.plot(temp, m*temp+c)
plt.grid(True)
plt.xlabel("Temperature (degrees C)")

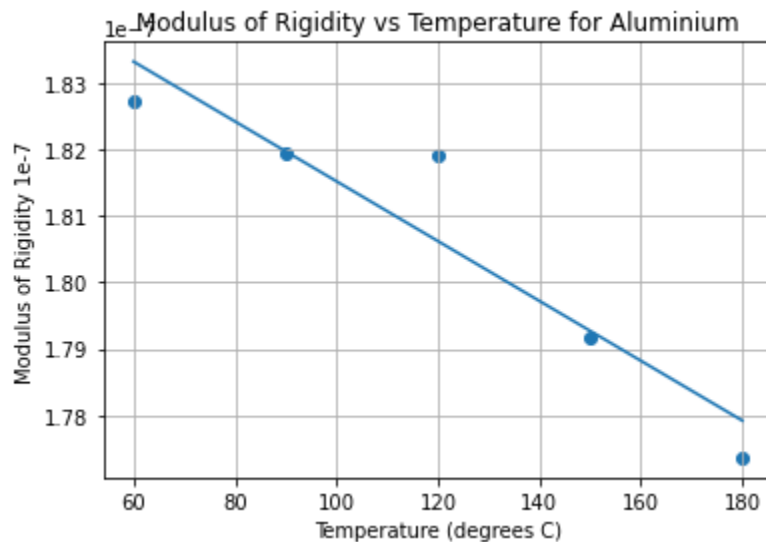
```

```
plt.ylabel("Modulus of Rigidity 1e-6")
plt.title("Modulus of Rigidity vs Temperature for Steel");
```



In [325...

```
plt.scatter(temp,rigidity_al) #Internal Friction vs Temperature for Aluminium
m, c = np.polyfit(temp,rigidity_al, 1)
plt.plot(temp, m*temp+c)
plt.grid(True)
plt.xlabel("Temperature (degrees C)")
plt.ylabel("Modulus of Rigidity 1e-7")
plt.title("Modulus of Rigidity vs Temperature for Aluminium");
```



In [277...

```
def int_frict(delta_w, w_0): #function used to find internal friction
    return (delta_w / (w_0 * (math.sqrt(3))))
```

In [347...

```
int_al = int_frict(deltaW_al,w_0_al) #Internal Friction vs Temperature for Various Metals
int_br = int_frict(deltaW_br,w_0_br)
int_st = int_frict(deltaW_st,w_0_st)
temp = np.array([60,90,120,150,180])

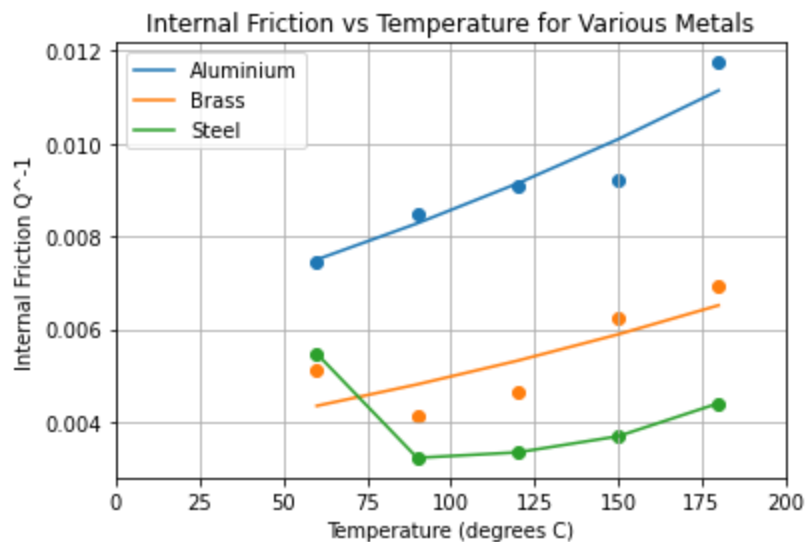
plt.scatter(temp,int_al)
print(np.polyfit(temp, np.log(int_al), 1))
#y ≈ exp(-5.09) * exp(3.3e-03 * temp) = 6.15e-3 * exp(3.3e-03 * temp)
6.15e-3 * np.exp(3.3e-03 * temp)
```

```
plt.scatter(temp,int_br)
print(np.polyfit(temp, np.log(int_br), 1))
#y ≈ exp(-5.64) * exp(3.37e-03 * temp) = 3.55e-3 * exp(3.37e-03 * temp)
plt.plot(temp, 3.55e-3 * np.exp(3.37e-03 * temp))

plt.scatter(temp,int_st)
plt.plot(temp, int_st)

plt.grid(True)
plt.xlim(0,200)
plt.xlabel("Temperature (degrees C)")
plt.ylabel("Internal Friction Q^-1")
plt.title("Internal Friction vs Temperature for Various Metals")
plt.legend(["Aluminium", "Brass", "Steel"]);
```

```
[ 3.30578257e-03 -5.09725062e+00]
[ 3.37674484e-03 -5.64322349e+00]
```



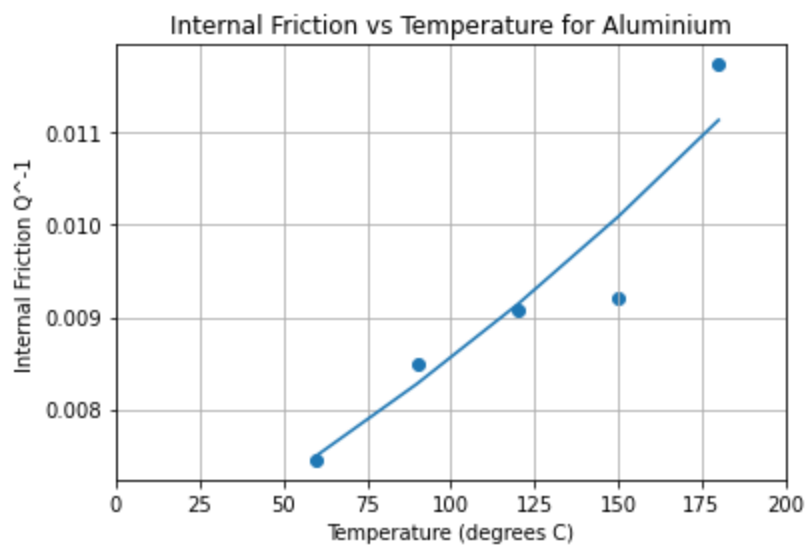
In [321...

```
plt.scatter(temp,int_al)
print(np.polyfit(temp, np.log(int_al), 1))
#y ≈ exp(-5.09) * exp(3.3e-03 * temp) = 6.15e-3 * exp(3.3e-03 * temp)
plt.plot(temp, 6.15e-3 * np.exp(3.3e-03 * temp))
plt.grid(True)
plt.xlim(0,200)
plt.xlabel("Temperature (degrees C)")
plt.ylabel("Internal Friction Q^-1")
plt.title("Internal Friction vs Temperature for Aluminium")
;
```

```
[ 3.30578257e-03 -5.09725062e+00]
```

Out[321...

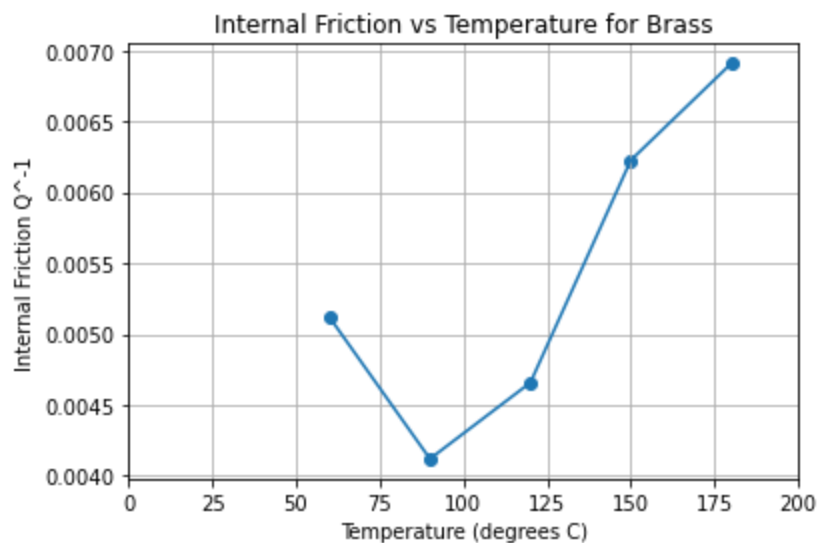
```
''
```

In [349...

```
plt.scatter(temp,int_br)
plt.plot(temp, int_br)
plt.grid(True)
plt.xlim(0,200)
plt.xlabel("Temperature (degrees C)")
plt.ylabel("Internal Friction  $Q^{-1}$ ")
plt.title("Internal Friction vs Temperature for Brass")
;
```

Out[349...

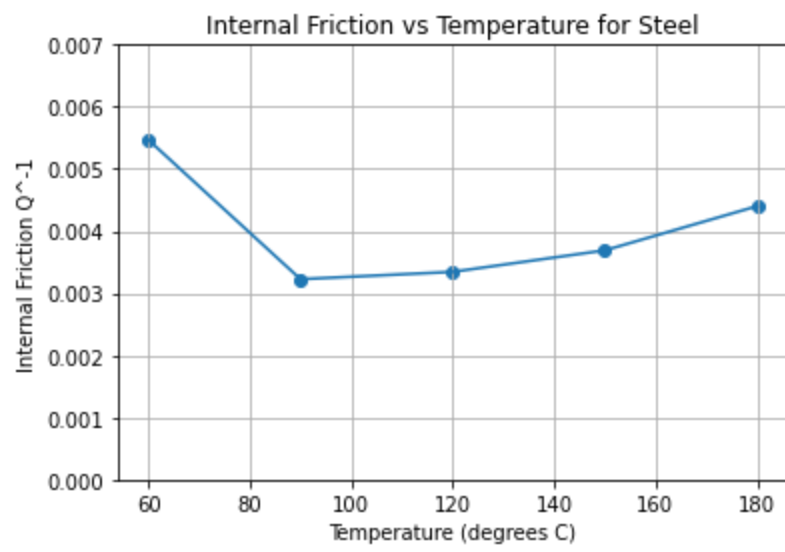


In [346...

```
plt.scatter(temp,int_st)
print(np.polyfit(temp, np.log(int_st), 1))
plt.plot(temp, int_st)
plt.grid(True)
plt.ylim(0,0.007)
plt.xlabel("Temperature (degrees C)")
plt.ylabel("Internal Friction  $Q^{-1}$ ")
plt.title("Internal Friction vs Temperature for Steel")
;
```

```
[-1.00025672e-03 -5.41469006e+00]
```

Out[346...



In []: