UCD School of Physics

PHYC30320 – Advanced Laboratory I

Modulus of Rigidity

By Joseph Anderson

Start Date: 13/09/2022

Table of Contents

Abstract

This experiment aimed to study the variation of modulus of rigidity, internal friction and Young's modulus with temperature. Phenomenon such as the atomic structure of solids and the thermal behaviour of solids were introduced by this experiment. Torsional oscillations are sent into a thin metallic rod (aluminium, brass or steel) clamped at one end and loaded at the other. The frequency response of these oscillations are studied in detail between temperatures of 303K-427K to find the modulus of rigidity, internal friction and Young's modulus of each rod. The modulus of rigidity was found to decrease with an increase of temperature. The internal friction tended to increase with an increase in temperature. The Young's modulus decreased as temperature increased. The elastic constants of each rod were also found in table 2. The values for Young's modulus at room temperature were also compared to theory in table 3.

Keywords: Modulus of rigidity; internal friction; Young's modulus; aluminium; brass; steel.

1 Introduction

Searle's apparatus is commonly used to measure rigidity. It is a versatile setup and can be used for textiles using specimens under standard tensions [1]. Searle's apparatus uses a weight to provide tension to a wire hence, by measuring the period of oscillations: Young's modulus and the modulus of rigidity can be determined [2]. Young's modulus can also be measured by using a mechanical oscillator in the form of a metallic rod with an eddy current driver and a simple crystal transducer which picks up the oscillations [3]. Modulus of rigidity can also be determined by measuring the angular displacement of a cylindrical shaft under an applied torque and then using the mechanics of materials to determine the modulus of rigidity [4].

This experiment aims to introduce the concept of microscopic causes of energy dissipation through the study of solids and their mechanical and thermal behaviours. This experiment aims to study the variation of the modulus of rigidity of rods of brass, steel and aluminium with temperature. The internal friction of each rod will also be analysed as a function of temperature. For a range of temperatures between 303 – 473 K the amplitude of the oscillation will be measured against the frequency of the applied torque.

2 Theory

A specimen rod is connected to an auxiliary body which undergoes oscillations from an external oscillatory torque [5]. The equation of motion for this system can be written as:

$$I\frac{d^2\theta}{dt^2} + \gamma\frac{d\theta}{dt} + C\theta = T_0 e^{iw_0 t} \quad (1)$$

where I is the moment of inertia of the system, γ is the damping coefficient and C is the torsion constant of the system. T_0 and w_0 are both constants [6].

In our system the specimen rod is considerably small relative to the auxiliary body. Hence, it is assumed that the inertia provided by the rod is negligible. Inertia provided by the auxiliary body is:

$$I = mL^2 \quad (2)$$

Where m is the mass of the auxiliary body and L is the distance from the axis of rotation which in our case is the radius of the axillary body as the axis of rotation is through the middle of the rod and auxiliary body [7].

Similarly, the restoring torque is mainly provided by the rod. Thus, it can be derived that solving for the torsion constant we find that [8]:

$$C = \frac{G\pi r^4}{2L} \tag{3}$$

where G is the modulus of rigidity, r is the radius of the specimen rod and L is the length of the specimen rod.

For the steady state solution of equation 1 as on the right-hand side the force is in its complex form the solution is given by [9]:

$$\theta = \theta_0 e^{i(wt-\delta)} \tag{4}$$

This can be substituted into equation 1 so θ_0 can be written in terms of the initial parameters dictated by equation 1. This is completed in [10] which gives us:

$$\theta_0 = \frac{\frac{T_0}{C}}{\sqrt{\left(1-\frac{Iw^2}{C}\right)^2 + \frac{w^2\gamma^2}{C^2}}} \tag{5}$$

In the low damping case, θ_0, will be maximum when the denominator is minimised which leads to the resonant (natural) frequency of the undamped oscillation [11] due to:

$$\left(1-\frac{Iw^2}{C}\right) = 0 \xrightarrow{solving\ for\ w\ gives} w = \sqrt{\frac{C}{I}} = w_0 \tag{6}$$

Substituting C into equation 3 and solving for the modulus of rigidity, G, gives and equation for modulus of rigidity in terms of resonant frequency and parameters of the auxiliary body and rod expressed by:

$$G = \frac{2LI(2\pi {w_0}^2)}{\pi r^4} \tag{7}$$

where r is the radius of the specimen rod, L is the length of the specimen rod, w_0 is the resonant frequency in Hz and I is the moment of inertia of the auxiliary body.

Using the full width of the resonance at half the maximum amplitude in equation 5 gives:

$$\Delta\omega = \frac{\sqrt{3}\,\gamma {w_0}^2}{C} \tag{8}$$

Thus, the internal friction of the specimen rod is given by:

$$Q^{-1} = \frac{\gamma w_0}{C} = \frac{\Delta\omega}{\sqrt{3}w_0} \tag{9}$$

Hence, internal friction is expressed in terms of the resonant frequency and the full width of the response curve at half the maximum amplitude.

The temperature dependence of the elastic constants can be rigorously studied for simple cubic at high temperatures [12]. For the modulus of rigidity however the relation is:

$$G(T) = G_0(1 - D\,\bar{\varepsilon}) \tag{10}$$

where $\bar{\varepsilon}$ is the mean energy per oscillator, G_0 is a free parameter and D is a parameter that depends on the type of the crystal or the model used [5][13].

Assuming an Einstein model for the solid it can be derived that the mean energy per oscillator can be written as a function of absolute temperature [14]:

$$\bar{\varepsilon} = \frac{1}{2}hv + \frac{hv}{e^{\left(\frac{hv}{kT}\right)} - 1} \tag{11}$$

where h is Planck's constant, v is the frequency, k is the Boltzmann constant and T is the absolute temperature [15]. Using equation 11 in equation 10 it can be derived that the following equation holds:

$$G(T) = G_0 - \left(\frac{s}{e^{\left(\frac{t}{T}\right)} - 1}\right) \tag{12}$$

In this s, t and G_0 are free parameters which is determined by the experimental data. Equation 10 gives only the lattice contribution to temperature dependence of the modulus of rigidity [5]. Meanwhile, equation 12 represents the entire temperature dependence meaning that both the lattice and the electronic contributions are accounted for [5]. Later, equation 12 is used to fit theoretical predictions of the variation of the modulus of rigidity and temperature.

In a simpler manner Young's modulus can be calculated using the resonant frequency of the rods. From the study of the propagation of sound waves within rods Young's modulus or the tensile elastic modulus is defined by the ratio of the equation below [16].

$$v = \sqrt{\frac{Y}{\rho}} \tag{13}$$

where Y is the Young's modulus of the rod, v is the speed of the torsional oscillations and ρ is the density of the rod. The speed of the torsional oscillation can be calculated assuming it behaves like a wave. This utilises a wave's velocity to be written as v=fλ and where the wavelength of the wave corresponds to half the length of the rod [17]. Hence, the following equation can be derived using equation 13 and hence Young's modulus can be calculated for the rod.

$$Y = \rho(2Lw_0)^2 \tag{14}$$

where L is the length of the rod and w_0 is the resonant frequency of the rod when that torsional oscillation is applied.

3 Apparatus and Experimental Procedure

3.1 Apparatus Setup

Figure 1 shows the experimental setup where each rod was vertically suspended by clamping the upper end. The auxiliary body and magnet was attached to the free end of the rod. The rod went through a heating coil which controlled the temperature of the rod. A small concave mirror was put on the auxiliary body. The auxiliary body isolated the magnet from the heater coil and also controls

the resonant frequency of the system. There was also a Helmholtz coil acting perpendicularly to the magnet attached to the auxiliary body. The Helmholtz coil applied the external torque which allows oscillations to occur. Laser light was produced towards the mirror which bounces the light towards a position sensing detector. This measured the amplitude of the oscillation.

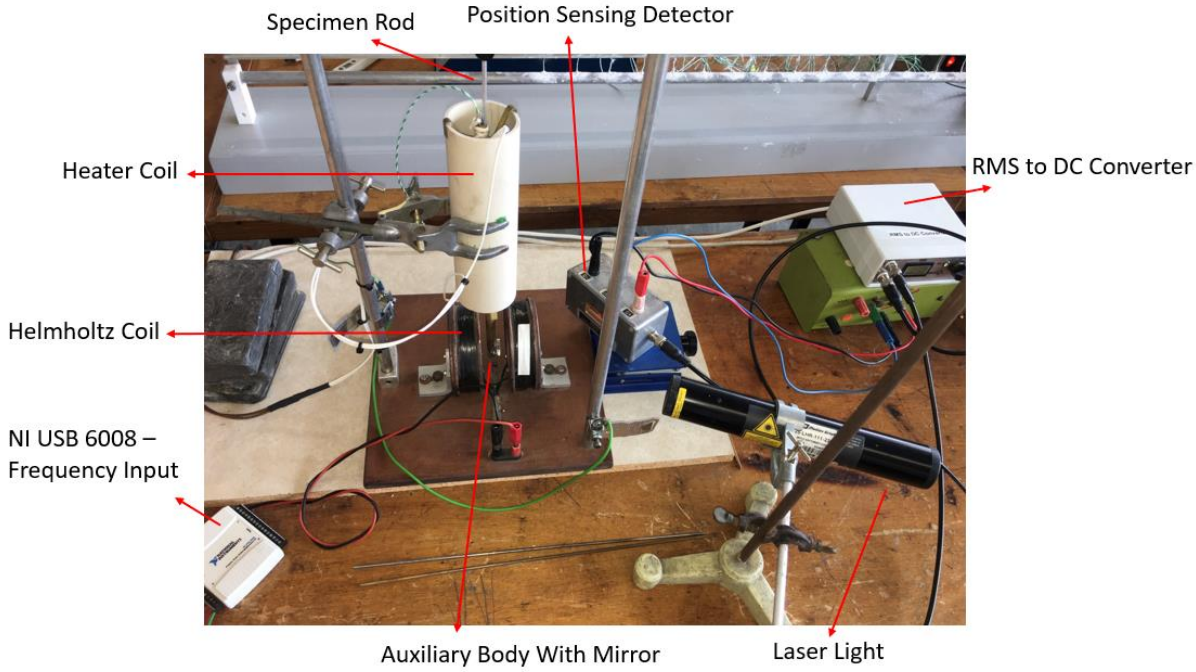


Figure 1 – Configuration of the experiment. The laser light is positioned such that the laser light reflects off the mirror attached to the auxiliary body and then hits the centre of the position sensing detector.

3.2 Experiment Procedure

Voltages were sent into the Helmholtz coil by a National Instruments USB-6008 which produced torsional oscillations on the rod and auxiliary body. These oscillations had corresponding frequencies. An input voltage of 0.1V was sent into the rod and then the corresponding frequency was read of the oscilloscope. The voltage was then increased by an increment of 0.1V and the measurement was recorded. This was done up to 5V. Hence, the relationship between input voltage and the corresponding frequency was defined.

Then, the input voltage was varied for a specific temperature while the amplitude voltages were received from the NI USB-6008 for each input frequency. 10 output voltages were taken and then the mean of these numbers was taken as the amplitude. The input voltage was increased by 0.1V between 0V and 5V. After this was plotted, the graph was visually inspected to see a more appropriate smaller range of voltages around the peak of the plot. Then the input voltages were taken between this smaller range in increments were set to 0.001V which took more amplitudes and produced a more accurately shaped plot, without taking too long. Hence, the resonant frequency, which was the frequency at which the peak was at, could be calculated assuming a Lorentzian fit to each frequency response curve. Data were collected for temperatures of 303, 333, 363, 393, 423, 448 and 473 K. This was repeated for rods of aluminium, brass and steel.

3.3 Precautions Taken During the Experiment

Precautions had to be taken for this experiment. The heater generator was not set above 473K as the heater was not allowed to be operated above this. While applying heat to the rod the heater coil

will get very hot so avoiding skin contact with the heater coil is essential. It was ensured that the thermocouple (used to measure the temperature of the rod) was inside the heating element, as otherwise the heater would not have stopped when the desired temperature was reached. The laser light was carefully managed to ensure it was not pointed in the direction of an eye. A board was placed between the reflection of the laser light off the position sensing detector and the rest of the lab. The Helmholtz coil is not magnetically shielded so magnetic and electrical conducting materials should be kept away from it, so it does not apply a distorted voltage to the rod. For the first temperature it was ensured that the entire line of light could fit in the position sensing detector. This was done by visually looking at the line sensing detector when the resonant frequency of this temperature was sent into the system. If the line didn't fit in the position sensing detector the gain was turned down.

4 Results & Analysis

4.1 Corresponding Input Voltages with Frequencies

Voltages were input into the specimen rod and then a corresponding frequency was produced read off the oscilloscope. This meant the relationship between the input voltage and corresponding frequency could be found. Plotting the data in our code gives us the equation for the line of best fit to 2 decimal places where y is the frequency in Hz and x is the input voltage in V:

$$y = 64.07x + 6.84 \tag{15}$$

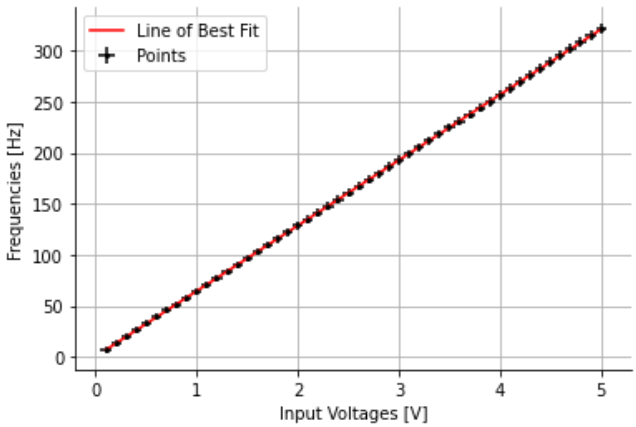


Figure 2 – Shows the linear positive relationship between the input voltage and the corresponding frequency found from the oscilloscope. Each point is plotted with its associated uncertainty. These uncertainties are difficult to see as they're small relative to the axis each is plotted on.

Uncertainties:

The resolution on the ADC input voltage is limited by the number of bits hence the uncertainty on the input voltage is the range divided by 2^N where N is the number of bits. The NI USB 6008 has 12 bits in differential mode and the range studied is 5V which gives the value for the uncertainty on each input voltage to be 0.0012V [18]. The uncertainty on the slope and y-intercept is calculated using the covariance matrix found by our code. The uncertainty on the slope is given by the square root of the first entry [0,0] of the matrix. This gives a value of 0.015 as the uncertainty on the slope of the line. Similarly, taking the square root of the [1,1] element of the matrix the uncertainty of the y-intercept is given by 0.042 to 2 significant figures. Hence, the following propagation of uncertainties formula is used:

$$\left(\frac{\Delta y}{y}\right)^2 = \left(\frac{\Delta m}{m}\right)^2 + \left(\frac{\Delta c}{c}\right)^2 + \left(\frac{\Delta x}{x}\right)^2 \quad (16)$$

where y is the input voltage, m is the slope of the line of best fit, c is the y-intercept of the line of best fit and x is the input voltage. This calculates the uncertainty on each individual measurement of the frequency.

To find the combined uncertainty the uncertainty on each frequency is added together in quadrature and divided by n:

$$\Delta y_{avg} = \frac{\sqrt{(\Delta y_1)^2 + (\Delta y_2)^2 + \ldots + (\Delta y_n)^2}}{n} \quad (17)$$

For our experiment data the value for n is 50. Hence, we find a value for the uncertainty on the frequency to be 0.16Hz to 2 significant figures.

4.2 The Frequency Response Curves for Aluminium, Brass and Steel Rods

The NI USB 6008 rolls through a range of frequencies finding the amplitudes of each one. This is then plotted on a curve then the temperature is changed, and the process is repeated. This leads to the response curves shown in figure 3-5.

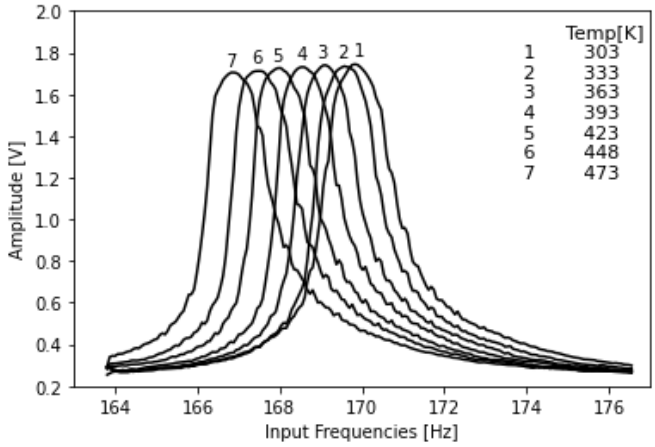


*Figure 3 - Shows the frequency response curves for the **aluminium** rod at various temperatures. The resonant frequency can be seen to be decreasing as the temperature is increased.*

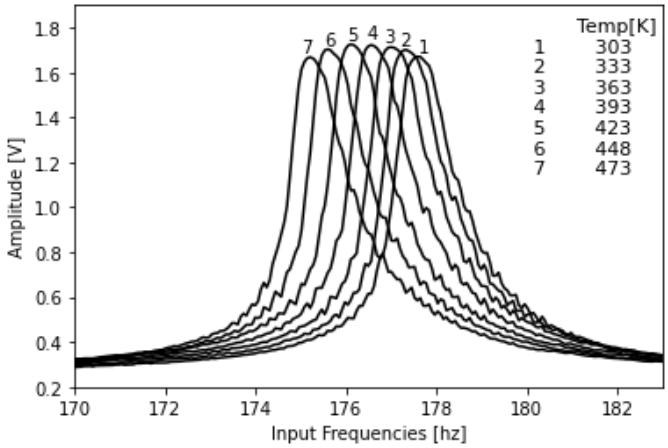


*Figure 4 - Shows the frequency response curves for the **brass** rod at various temperatures. The resonant frequency can also be seen to be decreasing as the temperature is increased.*

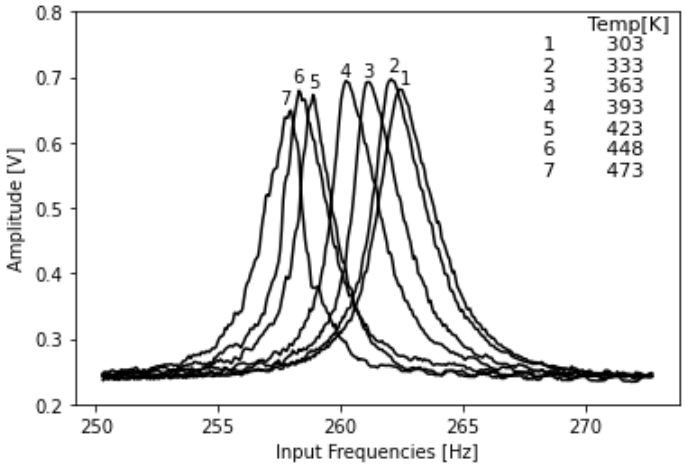


*Figure 5 - Shows the frequency response curves for the **steel** rod at various temperatures. The resonant frequency can also be seen to be decreasing as the temperature is increased.*

<u>Uncertainties:</u>

The uncertainty of the frequency has been calculated before at around 0.16Hz. There is also an uncertainty associated with the amplitude found from the position sensing detector (PSD). The resolution of the PSD is very good and has proven to be better than one part in one million meaning that the uncertainty on it is dictated by external factors [19]. A small uncertainty of 0.05V is taken on the amplitude of the oscillations as the temperature around the PSD was different between measurements depending on how long the apparatus was in use.

Generally, the response curves follow the same pattern. One potential outlier that's clear from figure 5 is the response curve for 423K. This particular measurement was performed on its own hence the position sensing detector could have been positioned slightly differently than the other values.

4.3 Lorentzian Function Fitting

To check whether a Lorentzian function was appropriate to fit our data we visually inspected the fitted Lorentzian curves that are shown below. The response curve is for the temperature 303K of each of the 3 rods of aluminium, brass, and steel.

The experiment points without uncertainties of the response curve are graphed in each (A) figure and the experiment points with uncertainties are graphed in each (B) figure. Each figure is plotted with a Lorentzian function with the same resonant frequency, the same maximum amplitude, and the same bandwidth at half the maximum as that plotted by the experiment values. This is done in order to compare the response curve of each material with a Lorentzian function. It can be seen that the Lorentzian curve matches the experimental data relatively well. Also, when the uncertainties in each (B) figure are plotted the Lorentzian curve is nearly always covered by the uncertainties. Thus, meaning that fitting the response curves as Lorentzian functions was an accurate choice.

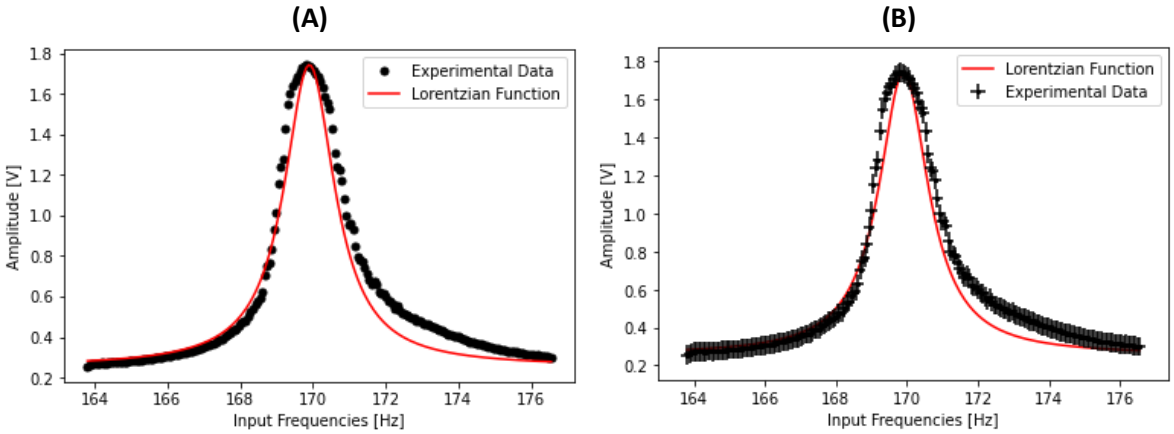


Figure 6 - Shows the experimental data from aluminium at a temperature of 303K. The Lorentzian curve is plotted using the resonant frequency and the bandwidth at half maximum of our data. (A) conveys the experimental points for the frequency response curve for aluminium at 303K. (B) includes the uncertainty on both the frequency and amplitude of the oscillation.

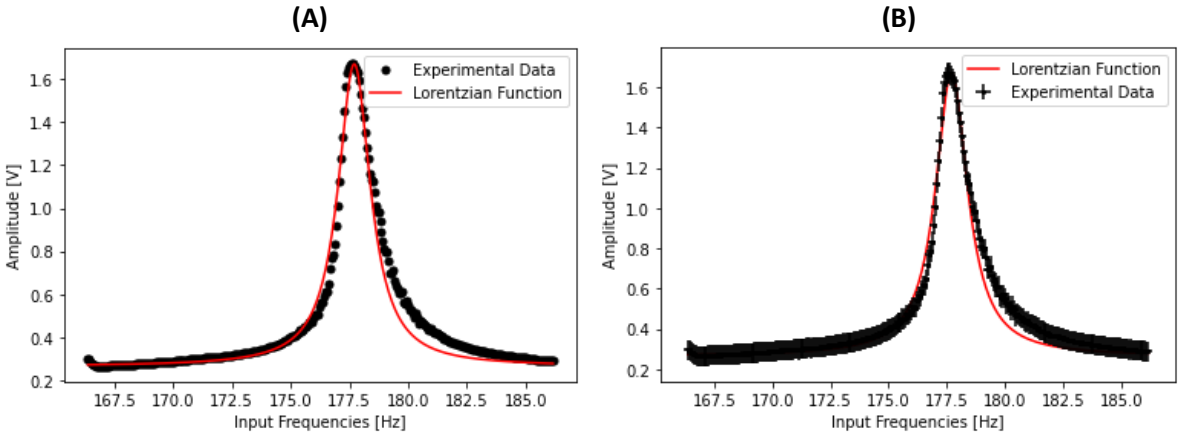


Figure 7 - Shows the experimental data from brass at a temperature of 303K. The Lorentzian curve is plotted using the resonant frequency and the bandwidth at half maximum of our data. (A) conveys the experimental points for the frequency response curve for brass at 303K. (B) includes the uncertainty on both the frequency and amplitude of the oscillation.

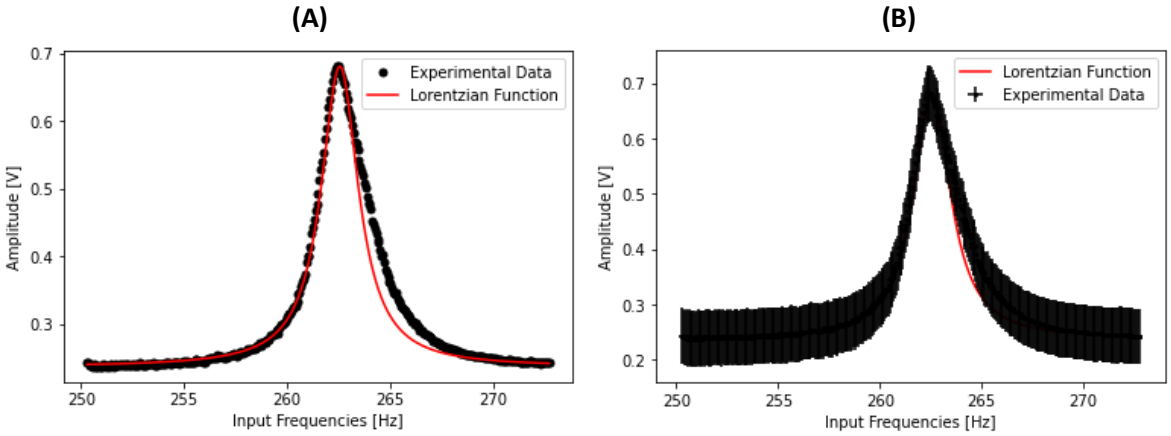


Figure 8 - Shows the experimental data from steel at a temperature of 303K. The Lorentzian curve is plotted using the resonant frequency and the bandwidth at half maximum of our data. (A) conveys the experimental points for the frequency response curve for steel at 303K. (B) includes the uncertainty on both the frequency and amplitude of the oscillation.

4.4 Determining the Moment of Inertia

The moment of inertia is needed to find the modulus of rigidity of each rod for equation 7. To find the moment of inertia equation 2 is used. The moment of inertia of the system is taken to be just the moment of inertia of the auxiliary body as the moment of inertia of the rod is small in comparison to that of the auxiliary body. Hence, the mass of the auxiliary body was measured to be 55.01 g and the radius of the auxiliary was measured as 0.00475 m. The moment of inertia can hence be calculated using equation 2 to be $1.2x10^{-6}\mathrm{kg}m^2$ to 2 significant figures.

Uncertainties:

The digital mass scale that was used to measure the mass increases in increments of 0.01g. Hence, the instrumentation precision is 0.01g which is taken as the uncertainty on the mass. A vernier callipers was used to measure the radius which increases in increments of 0.1mm. The uncertainty on this analogue instrumentation is taken as 0.1mm [20]. To calculate the uncertainty on the moment of inertia the propagation of errors formula is used on equation 2 which gives:

$$\left(\frac{\Delta I}{I}\right)^2 = \left(\frac{\Delta m}{m}\right)^2 + \left(2\frac{\Delta L}{L}\right)^2 \quad (18)$$

Rearranging equation 18 gives us an expression for the uncertainty of the moment of inertia which is calculated to be $0.05x10^{-6}\mathrm{kg}m^2$ rounded to 1 significant figure. Hence, the moment of inertia of the system observed is $(1.24 \pm 0.05)\ x10^{-6}kgm^2$.

4.5 Calculating the Modulus of Rigidity

The modulus of rigidity is calculated using equation 7 for every temperature of each rod. This is done using the resonant frequency which is the frequency at which the graph is at maximum. The length and radius of each rod was measured to give the values below.

Material	Length [m]	Radius [m]
Aluminium	0.355	0.0015
Brass	0.358	0.0015
Steel	0.349	0.0015

Table 1 - Depicts the length of each rod studied. The associated uncertainty on each measurement is displayed as well.

Uncertainties:

The length of each rod was measured using a ruler. There was an associated uncertainty with this as parallax can influence the measurement of a ruler. An uncertainty of 1mm was taken on the values for the length of the rod as the ruler used increased in increments of 1mm. Similarly, the associated uncertainty with the measurement of the radius using a vernier callipers was related to the increments which was 0.1mm. Thus, an uncertainty of 0.1mm was taken on the measurements of the radius. As discussed in section 4.1 the uncertainty on each frequency is 0.16Hz.

To calculate the uncertainty on each modulus of rigidity the uncertainties on the other variables in equation 7 is propagated:

$$\left(\frac{\Delta G}{G}\right)^2 = \left(\frac{\Delta I}{I}\right)^2 + \left(\frac{\Delta L}{L}\right)^2 + \left(2\frac{\Delta w_0}{w_0}\right)^2 + \left(4\frac{\Delta r}{r}\right)^2 \quad (19)$$

There is also an associated uncertainty on the temperature. The heat controller used had increments of 1K hence this is used as the uncertainty on the temperature.

Metal	G_0 $[Nm^{-2}]$	ΔG_0 $[Nm^{-2}]$	s $[Nm^{-2}]$	Δs $[Nm^{-2}]$	t [K]	Δt [K]
Brass	$7.00x10^{10}$	$0.03x10^{10}$	$4.92x10^{10}$	$3.87x10^{10}$	1484	398
Aluminium	$6.36x10^{10}$	$0.03x10^{10}$	$4.92\ x10^{10}$	$3.02\ x10^{10}$	1405	311
Steel	$1.51x10^{11}$	$0.01x10^{11}$	$5.91x10^{10}$	$3.72\ x10^{10}$	1009	316
Aluminium by Varshni [14]	$6.19x10^{10}$	-	$2.00x10^{10}$	-	293.6	-
Longitudinal Waves on Steel by Ledbetter [26]	$2.654x10^{11}$	-	$1.036\ x10^{10}$	-	221.9	-
Transverse Waves on Steel by Ledbetter [26]	$0.7047x10^{11}$	-	$0.05959x10^{11}$	-	264.4	-

Table 2 - Shows the values of the parameters G_0, s and t. The uncertainty on each value has been calculated using the covariance matrix in the analysis code shown in the appendix. Also shows the experimental values found from different sources [14] and [26]. Neither mention any estimation on the specific uncertainties on the elastic parameters.

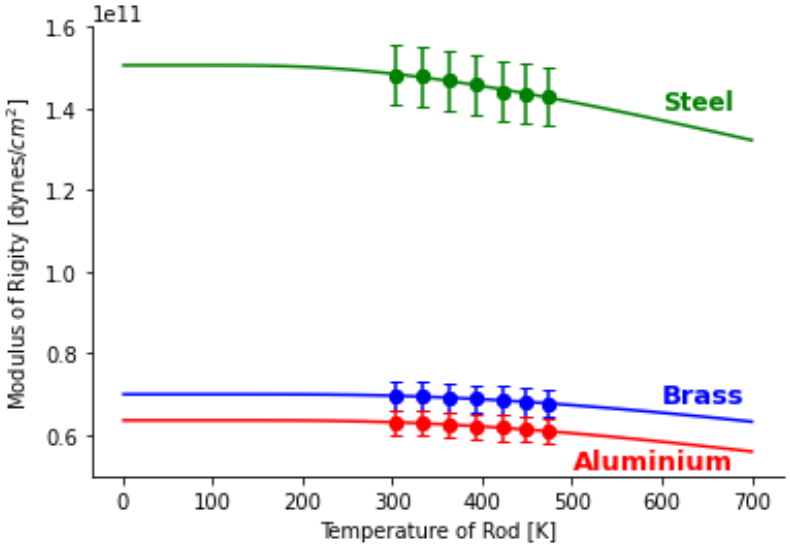


Figure 9 - Displays the variation of the modulus of rigidity of aluminium (bottom), brass (middle) and steel (top) with temperature. Each calculated experiment values are represented by the circles with its associated uncertainty. The continuous curves represent the theoretical behaviour based on equation 12 where the parameters G_0, s and t have been found in our code.

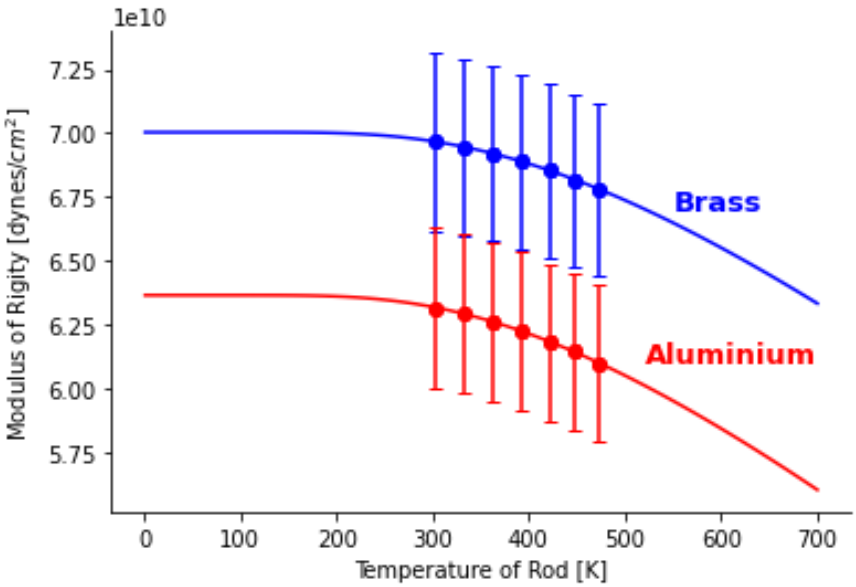


Figure 10 - Shows the modulus of rigidity varying with temperature with the brass and aluminium rods. Again, the continuous curves are the theoretical behaviour based off equation 12 while the circles represent the calculated experiment values. The smaller scale of this plot allows the details of the variation to be seen more closely.

It can be seen that steel has the highest modulus of rigidity over all temperatures. Brass has the next largest modulus of rigidity over every temperature and then aluminium has the lowest modulus of rigidity over all observed temperatures.

4.6 Determination of the Internal Friction

The internal friction of each rod at each specific temperature is calculated using equation 9. The resonant frequency is used along with the bandwidth at half the maximum amplitude to calculate this.

Uncertainties:

There is an uncertainty associated with each resonant frequency which has been previously calculated in section 4.1. There is also an uncertainty associated with each measurement of the bandwidth at half maximum. There is an uncertainty on the frequency measurement on each value of the bandwidth. Both of these have an individual uncertainty of 0.16Hz. These are added in quadrature to find the uncertainty on the bandwidth to be 0.05Hz.

$$\left(\frac{\Delta IF}{IF}\right)^2 = \left(\frac{\Delta w_0}{w_0}\right)^2 + \left(\frac{\Delta w}{w}\right)^2 \qquad (20)$$

where IF is the internal friction, w_0 is the resonant frequency and w is the bandwidth at half the maximum amplitude.

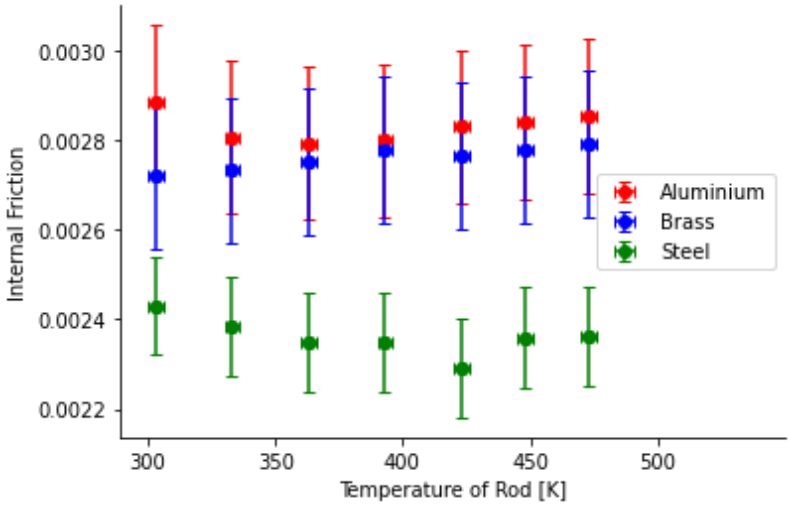


Figure 11 - Shows the internal friction varying with temperature for the three rods. The respective uncertainties on each value of the internal friction and the temperature of the rod is also shown.

Aluminium has the highest internal friction over every temperature. Then brass is the next largest. However, the values for aluminium and brass have overlapping uncertainties. Steel has the lowest internal friction. The aluminium rod seems to experience a small peak around 350K. In both the aluminium and brass rods, it's evident that the internal friction generally seems to be increasing as temperature rises. The steel rod is more ambiguous in that there doesn't seem to be a clear pattern to the variation.

4.7 Young's modulus for Each Rod

As an extension to our experiment, we can calculate the young's modulus of each temperature. This will allow us to study the variation of Young's modulus with temperature. This can be done using equation 14.

The density of aluminium is taken to be (2710 $\pm$ 10) kg/m^3 [21]. The density of brass is taken to be (8550 $\pm$ 150) kg/m^3[22]. The density of steel can be taken to be (7780 $\pm$ 150) kg/m^3[23]. Hence, values for Young's modulus for each rod can be calculated using the resonant frequency and length of each rod. Values for the Young's modulus of each rod at 303K are shown below.

Material	**Young's Modulus [$\mathbf{Nm^{-2}}$]**	**Uncertainty on Young's Modulus [$\mathbf{Nm^{-2}}$]**
Steel	2.1 x10^{11}	0.1 x10^{11}
Brass	1.1 x10^{11}	0.1 x10^{11}
Aluminium	7.5 x10^{10}	0.5 x10^{10}

Table 3 - shows the values calculated for Young's modulus at 303K for each of the three rods. The uncertainty on each value for Young's Modulus has also been calculated.

Uncertainties:

The uncertainty associated with each resonant frequency is 0.16Hz which was calculated in section 4.1. In section 4.5 an uncertainty of 1mm was taken on the values for the length of the rod as the ruler used increased in increments of 1mm. Hence the propagation of errors leads to the following equation.

$$\left(\frac{\Delta Y}{Y}\right)^2 = \left(2\frac{\Delta w_0}{w_0}\right)^2 + \left(2\frac{\Delta L}{L}\right)^2 \qquad (21)$$

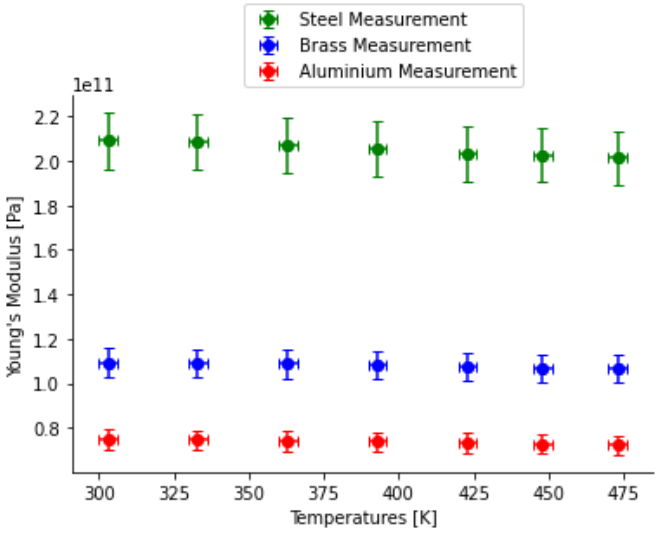


Figure 12 - Shows the Young's modulus varying with temperature for the three rods. The respective uncertainties on each value of Youngs and the temperature of the rod is also shown. The curves are drawn on the same scale for the purpose of comparison.

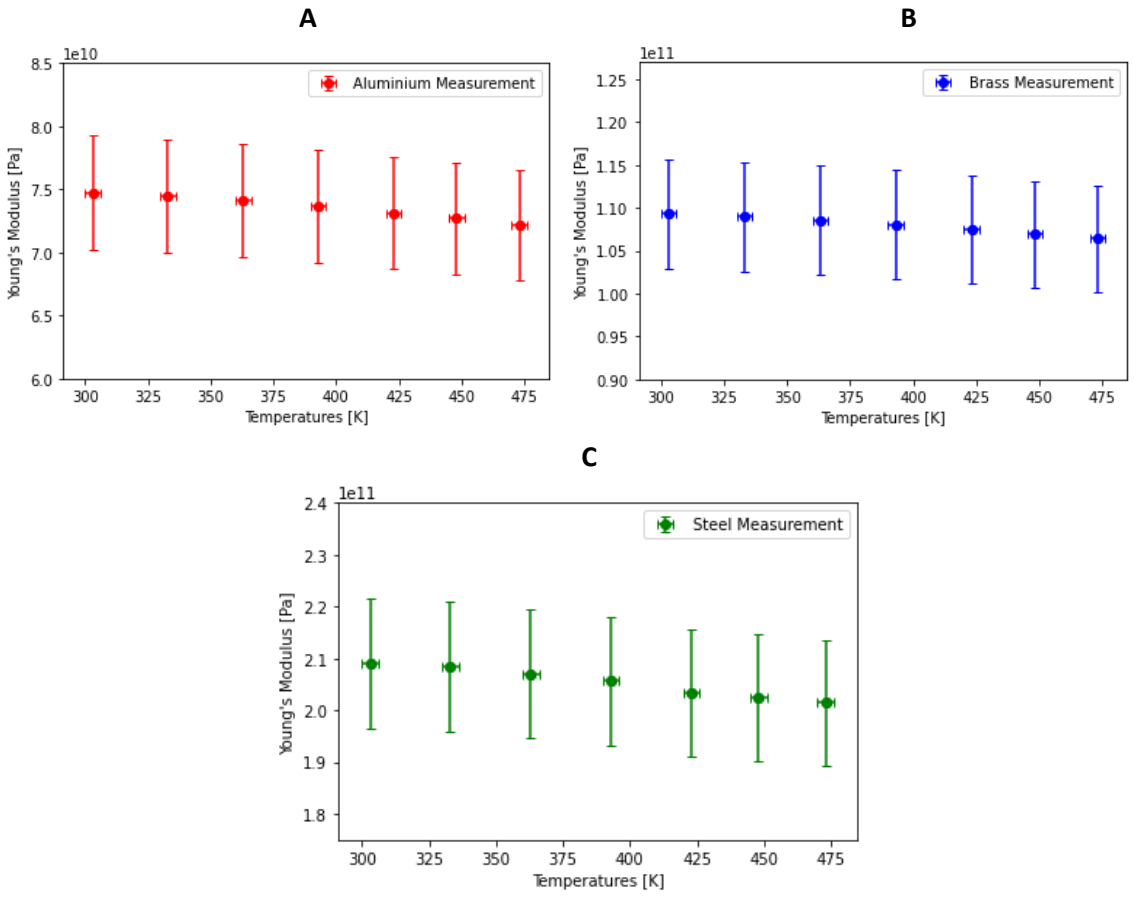


Figure 13 - Shows the Young's modulus varying with temperature for the three rods. (A) Shows the aluminium rod with its respective uncertainties. (B) Shows the brass rod with its associated uncertainties. (C) Shows the steel rod and its respective uncertainties. These are drawn on a smaller scale to emphasise the relationship between Young's modulus and the temperature of the rod.

It can be seen from figure 13 (A), (B) and (C) that for all three rods the Young's modulus decreases with rises in temperature.

5 Discussion

5.1 Comparison of Results with Theory

Generally, we observed that the modulus of rigidity decreased when temperature was increased which is in agreement with theoretical prediction of the properties of modulus of rigidity observed using a torsional oscillation of a wire with a weight attached [24]. The relative size of the modulus of rigidity for each of the rods is in agreement with those given by using the anticlastic plate-bending method [25].

We are not aware of any other work that finds the elastic parameters from Varshni's equation for brass. There have been studies for both copper and zinc which are combined to give the alloy brass [27][28]. The elastic constants G_0 and s for aluminium are in good agreement with those given by Varshni [14]. However, the elastic constant t is not in agreement with Varshni's theoretical calculation. Ledbetter and Read found the elastic constants of annealed 300-grade maraging steel using two different methods one that uses longitudinal waves and one that uses transverse waves [26]. Our value for the steel elastic parameters G_0 and s lies between the different sets of values calculated by Ledbetter and Read. However, the value for t doesn't agree with the values proposed by Ledbetter and Read.

The internal friction for brass seems to increase at a similar rate as that calculated using a beta-brass single crystal [29]. The beta-brass single crystal experiment observed a peak of internal friction at 588K. This was something we were unable to verify due to the temperature limit of the heat controller in use. There is a minimum in internal friction observed around 350K for the aluminium rod. This agrees with an experiment to study the internal friction of some aluminium alloys using high strength and high electrical conductivity [30]. The internal friction of the steel rod is of the same order as theoretical predictions calculated by deep cryogenic treatment on microstructure and evaluation by internal friction [31]. There were no definitive peaks observed for our experimental data on this range. A large peak is expected to occur around 600K (SKK peak) as a result of the composition of the specimen [32]. A smaller peak (Snoek peak) is also expected around 360K, however a peak is not observed in our experimental data [32].

Generally, from our experiment data Young's modulus can be seen to be decreasing as the temperature rises shown in figure 13. This is in agreement with theoretical predictions using a variety of different metals and engineering alloys [33]. Our value of $(1.1 \pm 0.1) \times 10^{11} Nm^{-2}$ for the Young's modulus of brass at a temperature of 303K is in agreement with values found from using a composite piezoelectric oscillator on beta-brass crystals [34]. The Young's modulus at 303K found for aluminium was $(7.5 \pm 0.5) \times 10^{10} Nm^{-2}$. The accepted value for the Young's modulus of aluminium is approximately $7 \times 10^{10} Nm^{-2}$ which falls within the interval of the values we found from our experiment data [35]. We calculated Young's modulus of steel at 303K to be $(2.1 \pm 0.1) \times 10^{11} Nm^{-2}$. The value for Young's modulus is calculated to be $2.02 \times 10^{11} Nm^{-2}$ using mechanical spectroscopy and nanoindentation which falls within the interval found by our experiment [36].

5.2 Reasoning Behind Discrepancies with Theoretical Predictions

Two main differences between our experiment data and theoretical predictions were the elastic constant t and also the failure to observe the Snoek peak for the internal friction of steel.

Our measurement of the elastic constant, t, for steel and aluminium were not in agreement with other experimental values [14] [26]. Mathematically the value of t in equation 12 corresponds to how flat the exponential function is. Variability in the temperature was limited to between 303K-

473K due to the heat controller being used. Hence, the flatness of the curve was dictated by seven points closely spaced together. Thus, there is a high level of inaccuracy on the flatness of the curve as initial points may lead to the wrong value for t rather than incorrect values for G_0 and s. This could be improved by studying higher temperatures of the rods with a better heater controller.

For the internal friction of steel, the Snoek peak was not evident in the data produced from our experiment. This peak can be calculated to be around 360K by using stress induced α' martensite phased steel [37]. Using low frequency, internal friction measurements can be carried out by the free decay method which show the small amplitude of the Snoek peak for steel [38]. Hence, due to the uncertainties on each internal friction calculated it is possible that this small peak is within our uncertainties. However, to verify our values for the internal friction for steal extending the experiment to beyond 600K would give the opportunity to observe the larger SKK peak. This peak has been calculated to be around 580K found from deep cryogenic treated M2 high-speed steel [39].

6 Conclusion

This experiment observed the variation of the modulus of rigidity with temperature for rods made of aluminium, brass and steel. The internal friction and Young's modulus was also studied as a function of the temperature of each rod.

In summary, the elastic constant (from Varshni's equation 12) of each rod were calculated shown in table 2. These were generally in agreement with other calculations using different methods [14] [26]. The elastic constant t was generally not in agreement with the other calculations. To improve this, it is proposed to take higher temperature measurements as outlined in section 5.2.

A minimum in internal friction for the steel rod is observed that matches theoretical predictions [30]. The Snoek peak was not observed around 360K for steel which was where it is expected to be from other experiment methods [37] [38]. This could be improved by checking if the larger peak at 600K matches the theoretical prediction [39]. From this, I learned that the internal friction of metals need to be aided with detailed knowledge of the composition of the metal to understand the reasoning for different peaks.

The values and associated uncertainties for Young's modulus for aluminium, brass and steel were calculated and shown in table 3. Other experimental values fall within the uncertainty interval for each of our calculated Young's modulus [34] [35] [36]. Hence, we can verify our experimental technique for calculating Young's modulus for metal rods.

7 References

[1] J. Owen (1965) 26—The Application of Searle's Single and Double Pendulum Methods to Single Fibre Rigidity Measurements. *Journal of the Textile Institute Transactions*, 56(6), pp.T329-T339. Available at: <https://www.biodiversitylibrary.org/item/121922#page/862/mode/2up> [Accessed 30 September 2022].

[2] G. Searle (1900) XII. On the elasticity of wires. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, [online] 49(297), pp.193-199. Available at: < https://www.biodiversitylibrary.org/item/122317#page/208/mode/2up > [Accessed 30 September 2022].

[3] H. Simpson and P. Wolfe (1975) Young's modulus and internal damping in a vibrating rod. *American Journal of Physics*, [online] 43(6), pp.506-508. Available at: <https://aapt.scitation.org/doi/pdf/10.1119/1.9779> [Accessed 30 September 2022].

[4] P. Joyce (2022) *Two Experiments to Teach Modulus of Elasticity and Modulus of Rigidity*. [ebook] Annapolis: 2004 Annual Conference Proceedings, pp.7-10. Available at: <https://peer.asee.org/two-experiments-to-teach-modulus-of-elasticity-and-modulus-of-rigidity> [Accessed 30 September 2022].

[5] J. Shanker, S. Kushwah and P. Kumar et al. (1997) Theory of thermal expansivity and bulk modulus for MgO and other minerals at high temperatures. *Physica B: Condensed Matter*, [online] 233(1), pp.78-83. Available at: <https://www.sciencedirect.com/science/article/pii/S0921452696012380> [Accessed 1 October 2022].

[6] P. Belotserkovskiy (1997) Forced Oscillations and Resonance of Infinite Periodic Strings. *Journal of Sound and Vibration*, [online] 204(1), pp.41-57. Available at: <https://www.sciencedirect.com/science/article/pii/S0022460X97909184> [Accessed 1 October 2022].

[7] B**.** Oostra (2006) Moment of Inertia Without Integrals. *The Physics Teacher*, [online] 44(5), pp.283-285. Available at: <https://aapt.scitation.org/doi/10.1119/1.2195398> [Accessed 1 October 2022].

[8] P. Woodfield and A. Seagar (2012) Magnetic Damping Effects in Forced-Oscillation Vibrating-Wire Viscometers. *International Journal of Thermophysics*, [online] 33(2), pp.259-278. Available at: <https://link.springer.com/content/pdf/10.1007/s10765-012-1157-5.pdf> [Accessed 1 October 2022].

[9] Y. Bopche (2020) *Forced Oscillations*. [ebook] Gondia: DBscience, pp.1-9. Available at: <https://www.dbscience.org/wp-content/uploads/2020/03/BSC-SEM-II-PAPER-I-UNIT-II-FORCED-OSCILLATOR.pdf> [Accessed 1 October 2022].

[10] S. Alrasheed (2019) *Principles of Mechanics : Fundamental University Physics*. 1st ed. Thuwal, Saudi Arabia: Springer International Publishing, pp.168-171. Available at: <https://link.springer.com/book/10.1007/978-3-030-15195-9> [Accessed 1 October 2022].

[11] C. Kittel, W. Knight and M. Ruderman (1965) *Mechanics: Berkeley physics course: Volume 1*. 1st ed. New York: McGraw-Hill New York, pp.215-243. Available at:

<http://www.astrosen.unam.mx/~posgrado/libros/1_Mechanics_Kittel_BPC.pdf> [Accessed 1 October 2022].

[12] M. Born (1939) Thermodynamics of Crystals and Melting. *The Journal of Chemical Physics*, [online] 7(8), pp.591-603. Available at: <https://aip.scitation.org/doi/abs/10.1063/1.1750497> [Accessed 6 October 2022].

[13] G. Leibfried and W. Ludwig (1961) Theory of Anharmonic Effects in Crystals. *Solid State Physics*, [online] 12(1), pp.275-312. Available at: <https://www.sciencedirect.com/science/article/pii/S0081194708606566> [Accessed 6 October 2022].

[14] Y. Varshni (1970) Temperature Dependence of the Elastic Constants. *Physical Review B*, [online] 2(10), pp.3952-3958. Available at: <https://journals.aps.org/prb/pdf/10.1103/PhysRevB.2.3952> [Accessed 6 October 2022].

[15] M Planck (1900) On the theory of the law of energy distribution in the normal spectrum. *The German Physical Society*, [online] 2(1), pp.237-245. Available at: <http://www.ffn.ub.es/luisnavarro/nuevo_maletin/Planck%20(1900),%20Distribution%20Law.pdf> [Accessed 6 October 2022].

[16] S. Errede (2017) *The Physics of a Longitudinally Vibrating Metal Rod*. 1st ed. [ebook] University of Illinois: UIUC Physics, pp.1-5. Available at: <https://courses.physics.illinois.edu/phys406/sp2017/Lecture_Notes/Vibrating_Rod/Longitudinally_Vibrating_Singing_Rod.pdf> [Accessed 9 October 2022].

[17] C. Liu and D. Jiang (2020) Torsional vibration characteristics and experimental study of cracked rotor system with torsional oscillation. *Engineering Failure Analysis*, [online] 116, p.104737. Available at: <https://www.sciencedirect.com/science/article/pii/S135063072030128X> [Accessed 9 October 2022].

[18] USB 6008 Hardware Manual (2017) *Accuracy Specifications for NI USB-6008 and USB-6009 - NI*. [online] Knowledge.ni.com. Available at: <https://knowledge.ni.com/KnowledgeArticleDetails?id=kA00Z000000P9SjSAK&l=en-IE> [Accessed 4 October 2022].

[19] SiTek PSD Data Sheet (2022) *High Linearity Position Sensing Detector*. [online] Lasercomponents.com. Available at: <https://www.lasercomponents.com/de/?embedded=1&file=fileadmin/user_upload/home/Datasheets/sitek/1l30_su2.pdf&no_cache=1> [Accessed 8 October 2022].

[20] T. Svensson (2014) *Estimation of measurement uncertainty for length measurement with vernier calipers*. [online] Applications of statistics in Measurement & Testing. Available at: <https://metrology.wordpress.com/measurement-process-index/8-presentation-and-interpretation-of-measurement-results-decisions/estimation-of-measurement-uncertainty-for-length-measurement-with-vernier-calipers/> [Accessed 5 October 2022].

[21] M. Assael et al. (2006) Reference Data for the Density and Viscosity of Liquid Aluminum and Liquid Iron. *Journal of Physical and Chemical Reference Data*, [online] 35(1), pp.285-300. Available at: <https://aip.scitation.org/doi/pdf/10.1063/1.2149380> [Accessed 10 October 2022].

[22] R. Walker (2011) *Mass, Weight, Density or Specific Gravity of Different Metals*. [online] Simetric.co.uk. Available at: <https://www.simetric.co.uk/si_metals.htm> [Accessed 10 October 2022].

[23] S. Hawkins (1997) *Re: What is the density of steel?*. [online] Madsci.org. Available at: <http://www.madsci.org/posts/archives/dec97/874069066.Ch.r.html> [Accessed 10 October 2022].

[24] K. Iokibe and S. Sakai (1921) XLIV. The effect of temperature on the modulus of rigidity, and on the viscosity of solid metals. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, [online] 42(249), pp.397-418. Available at: < https://www.biodiversitylibrary.org/item/121924#page/409/mode/1up > [Accessed 10 October 2022].

[25] M. Farshad, M. Wildenberg and P. Flüeler (1997) Determination of shear modulus and Poisson's ratio of polymers and foams by the anticlastic plate-bending method. *Materials and Structures*, [online] 30(6), pp.377-382. Available at: <https://link.springer.com/content/pdf/10.1007/BF02480690.pdf> [Accessed 10 October 2022].

[26] H. Ledbetter and D. Read (1977) Low-temperature elastic properties of a 300-grade maraging steel. *Metallurgical Transactions A*, [online] 8(11), pp.1805-1808. Available at: <https://link.springer.com/content/pdf/10.1007/BF02646886.pdf> [Accessed 10 October 2022].

[27] H. Ledbetter (1981) Elastic constants of polycrystalline copper at low temperatures. Relationship to single-crystal elastic constants. *Physica Status Solidi (a)*, [online] 66(2), pp.477-484. Available at: <https://onlinelibrary.wiley.com/doi/abs/10.1002/pssa.2210660209> [Accessed 11 October 2022].

[28] H. Ledbetter (1977) Elastic properties of zinc: A compilation and a review. *Journal of Physical and Chemical Reference Data*, [online] 6(4), pp.1181-1203. Available at: <https://aip.scitation.org/doi/pdf/10.1063/1.555564> [Accessed 11 October 2022].

[29] R. Artman (1952) Temperature Dependence of Young's Modulus and Internal Friction of Single Crystals of Beta-Brass. *Journal of Applied Physics*, [online] 23(4), pp.475-482. Available at: <https://aip.scitation.org/doi/pdf/10.1063/1.1702230> [Accessed 11 October 2022].

[30] W. Wang et al. (2021) Internal friction and heat resistance of Al, Al–Ce, Al–Ce–Zr and Al–Ce–(Sc)–(Y) aluminum alloys with high strength and high electrical conductivity. *Journal of Materials Research and Technology*, [online] 14, pp.1255-1274. Available at: <https://www.sciencedirect.com/science/article/pii/S2238785421007171> [Accessed 11 October 2022].

[31] S. Li et al. (2010) Influence of deep cryogenic treatment on microstructure and evaluation by internal friction of a tool steel. *Cryogenics*, [online] 50(11-12), pp.754-758. Available at: <https://www.sciencedirect.com/science/article/pii/S0011227510001918> [Accessed 11 October 2022].

[32] M. Fukuhara and A. Sanpei (1993) Elastic Moduli and Internal Friction of Low Carbon and Stainless Steels as a Function of Temperature. *ISIJ International*, [online] 33(4), pp.508-512. Available at: <https://www.jstage.jst.go.jp/article/isijinternational1989/33/4/33_4_508/_pdf/-char/ja> [Accessed 11 October 2022].

[33] H. Ledbetter (1982) Temperature behaviour of Young's moduli of forty engineering alloys. *Cryogenics*, [online] 22(12), pp.653-656. Available at:

<https://www.sciencedirect.com/science/article/abs/pii/0011227582900728> [Accessed 11 October 2022].

[34] J. Rinehart (1940) Temperature Dependence of Young's Modulus of Beta-Brass Single Crystals. *Physical Review*, [online] 58(4), pp.365-371. Available at: <https://journals.aps.org/pr/pdf/10.1103/PhysRev.58.365> [Accessed 11 October 2022].

[35] The Editors of Encyclopaedia Britannica (2022) *Young's modulus | Description, Example, & Facts*. [online] Encyclopedia Britannica. Available at: <https://www.britannica.com/science/Youngs-modulus> [Accessed 11 October 2022].

[36] S Balijepalli et al. (2013) Young's Modulus Profile in Kolsterized AISI 316L Steel. *Materials Science Forum*, [online] 762, pp.183-188. Available at: <https://www.scientific.net/msf.762.183> [Accessed 11 October 2022].

[37] N. Igata, H. Chen and K. Miyahara (1982) AN INTERNAL FRICTION PEAK DUE TO STRESS INDUCED α' MARTENSITE IN A SUS 304 STEEL. *Le Journal de Physique Colloques*, [online] 43(C4), pp.C4-547-C4-550. Available at: <https://hal.archives-ouvertes.fr/jpa-00222205> [Accessed 12 October 2022].

[38] X. Lu et al . (2014) Origin of low-temperature shoulder internal friction peak of Snoek-Köster peak in a medium carbon high alloyed steel. *Solid State Communications*, [online] 195, pp.31-34. Available at: <https://www.sciencedirect.com/science/article/pii/S0038109814002737> [Accessed 12 October 2022].

[39] L. Zhou et al. (2019) Nanoscratch and internal friction investigations of deep cryogenic treated M2 high-speed steel. *Heat Treatment and Surface Engineering*, [online] 1(3-4), pp.109-114. Available at: <https://www.tandfonline.com/doi/full/10.1080/25787616.2020.1748340?scroll=top&needAccess=true> [Accessed 12 October 2022].

[40] B. Prudholm (2014) Fit points to a Lorentzian curve and find center and half maximum bandwidth in Python. [online] Stack Overflow. Available at: <https://stackoverflow.com/questions/24437070/fit-points-to-a-lorentzian-curve-and-find-center-and-half-maximum-bandwidth-in-p> [Accessed 3 October 2022].

8 Appendix

8.1 Raw Data

Voltage Input [V]	Frequency [Hz]	Voltage Input [V]	Frequency [Hz]	Voltage Input [V]	Frequency [Hz]	Voltage Input [V]	Frequency [Hz]
0.1	6.93	1.4	90.19	2.7	173.43	4.0	256.76
0.2	13.32	1.5	96.65	2.8	179.81	4.1	263.17
0.3	19.76	1.6	102.99	2.9	186.29	4.2	269.61
0.4	26.22	1.7	109.34	3.0	192.72	4.3	275.94
0.5	32.56	1.8	115.64	3.1	199.11	4.4	282.36
0.6	38.97	1.9	121.98	3.2	205.51	4.5	288.74
0.7	45.32	2.0	128.41	3.3	211.87	4.6	295.21
0.8	51.68	2.1	134.87	3.4	218.21	4.7	301.62
0.9	58.01	2.2	134.87	3.5	224.68	4.8	308.07
1.0	64.41	2.3	147.74	3.6	231.04	4.9	314.47
1.1	70.89	2.4	154.13	3.7	237.38	5.0	321.03
1.2	77.37	2.5	160.61	3.8	243.73		
1.3	83.81	2.6	167.01	3.9	250.18		

Table 4 - Conveys the raw data of each measurement of frequency on the oscilloscope after the input voltage was sent into the system.

Aluminium		**Brass**		**Steel**	
Temperature [K]	Resonant Freq [Hz]	Temperature [K]	Resonant Freq [Hz]	Temperature [K]	Resonant Freq [Hz]
303	169.9	303	177.7	303	262.6
333	169.2	333	177.4	333	262.3
363	169.1	363	177.1	363	261.4
393	168.6	393	176.7	393	260.5
423	168.1	423	176.2	423	259.0
448	167.6	448	175.7	448	258.5
473	166.9	473	175.3	473	257.8

Table 5 - Shows the resonant frequency (rounded to 1 decimal place) for each temperature for each of the three rods. The resonant frequency is the frequency at which the amplitude of the frequency response curves are at its maximum.

Aluminium		**Brass**		**Steel**	
Temperature [K]	Bandwidth at half maximum amplitude[Hz]	Temperature [K]	Bandwidth at half maximum amplitude[Hz]	Temperature [K]	Bandwidth at half maximum amplitude[Hz]
303	0.850	303	0.838	303	1.105
333	0.825	333	0.840	333	1.084
363	0.818	363	0.844	363	1.063
393	0.817	393	0.851	393	1.061
423	0.824	423	0.844	423	1.028
448	0.824	448	0.847	448	1.056
473	0.826	473	0.848	473	1.055

Table 6 – Shows the bandwidth of the response curve at half maximum amplitude for each temperature of the three rods. Each figure is rounded to 3 decimal places. These values are used in the calculation of internal friction.

Material	Temperature [K]	Modulus of Rigidity $[x10^{10}\mathrm{Nm}^{-2}]$	Uncertainty on Modulus of Rigidity $[x10^{10}\mathrm{Nm}^{-2}]$
Aluminium	303	6.314	0.315
	333	6.294	0.314
	363	6.259	0.313
	393	6.222	0.311
	423	6.179	0.309
	448	6.143	0.307
	473	6.098	0.305
Brass	303	6.966	0.348
	333	6.946	0.347
	363	6.921	0.346
	393	6.888	0.344
	423	6.853	0.342
	448	6.815	0.340
	473	6.782	0.339
Steel	303	14.825	0.740
	333	14.789	0.739
	363	14.688	0.733
	393	14.589	0.729
	423	14.419	0.720
	448	14.369	0.718
	473	14.291	0.714

Table 7 – Shows the modulus of rigidity for each temperature of the three rods. Also shows the uncertainty on each value of the modulus of rigidity. Each figure is rounded to 3 decimal places. These are used in graphing the modulus of rigidity against the temperature of each rod to study its relationship.

Material	Temperature [K]	Internal Friction $[x10^{-3}]$	Uncertainty on Internal Friction $[x10^{-3}\mathrm{N}]$
Aluminium	303	2.887	0.170
	333	2.807	0.170
	363	2.793	0.171
	393	2.800	0.171
	423	2.830	0.172
	448	2.840	0.172
	473	2.855	0.173
Brass	303	2.721	0.162
	333	2.733	0.163
	363	2.751	0.163
	393	2.779	0.163
	423	2.765	0.164
	448	2.781	0.164
	473	2.791	0.165

Steel	303	2.430	0.110
	333	2.385	0.110
	363	2.349	0.110
	393	2.351	0.111
	423	2.292	0.111
	448	2.360	0.112
	473	2.363	0.112

Table 8 – Shows the internal friction for each temperature of the three rods. Also shows the uncertainty on each value of the internal friction. Each figure is rounded to 3 decimal places. These are used in graphing the internal friction against the temperature of each rod to study its relationship.

Material	**Temperature [K]**	**Young's Modulus** $[x10^{11}\mathrm{Nm}^{-2}]$	**Uncertainty on Young's Modulus** $[x10^{11}\mathrm{Nm}^{-2}]$
Aluminium	303	0.747	0.045
	333	0.745	0.045
	363	0.741	0.045
	393	0.736	0.045
	423	0.731	0.044
	448	0.727	0.044
	473	0.722	0.044
Brass	303	1.093	0.064
	333	1.090	0.064
	363	1.086	0.064
	393	1.081	0.063
	423	1.075	0.063
	448	1.069	0.063
	473	1.064	0.062
Steel	303	2.090	0.126
	333	2.085	0.126
	363	2.070	0.125
	393	2.057	0.124
	423	2.033	0.123
	448	2.026	0.122
	473	2.014	0.122

Table 9 – Shows the Young's modulus for each temperature of the three rods. Also shows the uncertainty on each value of the Young's modulus. Each figure is rounded to 3 decimal places. These are used in graphing the Young's modulus against the temperature of each rod to study its relationship.

8.2 Data Acquisition Python Code

Data Acquisition Code

In [ ]:
```
def my_mean(sample):
    return sum(sample)/len(sample)
# Function to calculate the mean of data
```

In [ ]:
```
import matplotlib.pyplot as plt
from time import sleep
import numpy as np

from pydaqmx_helper.dac import DAC
from pydaqmx_helper.adc import ADC
```

In [ ]:
```
voltages = np.arange(0,5,0.1)
# Runs from 0 to 5 at increments of 0.1 first and then change it to run over a smaller interval closer to the peak

myDAC = DAC(1)
# Uses channel 1 for the DAC class

myADC = ADC()
myADC.addChannels([0],minRange=0,maxRange=5)
# Uses channel 0 for the DAC class

delay_s = 0.05
# Adds a delay inbetween each reading

amplitudes = []
# Amplitudes is an empty list


for vout in voltages:
    myDAC.writeVoltage(vout)
    sleep(delay_s)

    data = myADC.sampleVoltages(10,100)
    # Takes 10 readings of the voltage

    amplitudes.append(my_mean(data[0]))
    # Inserts each mean amplitude into the amplitude list

plt.plot(voltages, amplitudes)

plt.xlabel("Input Voltages [V]")
plt.ylabel("Amplitude Voltage [V]")
```

8.3 Experiment Data Analysis Python Code

Modulus of Rigidity Experiment Code

In [1]:
```
import numpy as np
import scipy as sp
import matplotlib.pyplot as plt
from scipy.optimize import curve_fit
```

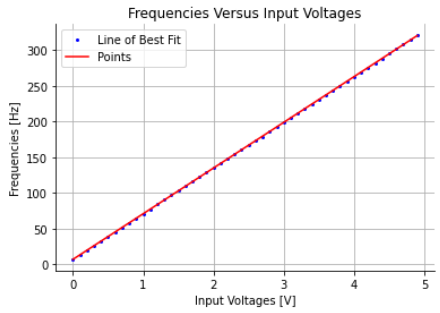
In [2]:
```
plt.rcParams['figure.figsize'] = (6,4)
plt.rcParams['font.size'] = 12
plt.rcParams['savefig.bbox'] = 'tight'
```

Corresponding Input Voltages with Frequencies

In [3]:
```
voltages = np.arange(0,5,0.1)
# Input voltages

freqs_obs = np.array([6.93, 13.32, 19.76,26.22, 32.56, 38.97, 45.32, 51.68, 58.01, 64.41, 70.89, 77.37, 83.81,
                      90.19, 96.65, 102.99, 109.34, 115.64, 121.98, 128.41, 134.87, 141.25, 147.74, 154.13, 160.61,
                      167.01, 173.43, 179, 186.29, 192.72, 199.11, 205.51, 211.87, 218.21, 224.68, 231.04, 237.38,
                      243.73, 250.18, 256.76, 263.17, 269.61, 275.94, 282.36, 288.74, 295.21, 301.62, 308.07,
                      314.47, 321.03])
# Observed frequencies for each input voltage
```

In [4]:
```
plt.plot(voltages, freqs_obs, 'bo',markersize= 2)

def fitfunc(x,m,c):
 return m*x+c
#Plot line of best fit

pars, cov = curve_fit(fitfunc,voltages,freqs_obs)
plt.plot(voltages,fitfunc(voltages,*pars),'r-')

plt.xlabel("Input Voltages [V]")
plt.ylabel("Frequencies [Hz]")
plt.title("Frequencies Versus Input Voltages")

plt.grid(True)

plt.legend(['Line of Best Fit','Points'],loc='upper left')
# Plots the legend

print("The slope of the line of best fit is", pars[0], "\n")
print("The y-intercept of the line of best fit is", pars[1], "\n")

print("Covariance Matrix \n", cov, "\n")
# Gives the Covariance matrix

# Removes the top axis and right axis
ax = plt.subplot(111)
ax.spines["right"].set_visible(False)
ax.spines["top"].set_visible(False)
```

```
The slope of the line of best fit is 64.06859542483018

The y-intercept of the line of best fit is 6.835741220674732

Covariance Matrix
 [[ 0.00021781 -0.00053364]
 [-0.00053364  0.00176103]]
```



Calculating the **Uncertainty on the Frequency Values**:

In [5]:
```
def uncert_Frequency(m, Δm, x, Δx, c, Δc, y):
    return y *(  (Δm/m)**2 + (Δx/x)**2 + (Δc/c)**2 )**(0.5)
# Rearrangemnt of equation 14 to give us a value for uncertainty of frequency
```

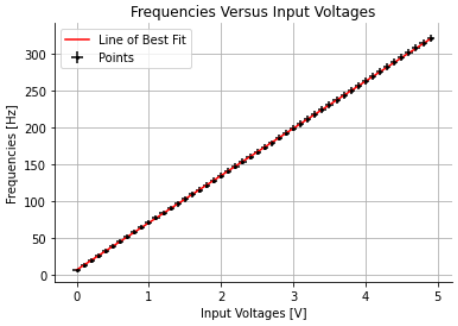
In [6]:
```
m = 64.06859542
Δm = 0.014758529469518425
# Values of the slope and uncertainty using the covariance matrix

x = voltages
Δx = 5/(2**12)
# Values for the input voltages and its associated uncertainty

c = 6.83574122
Δc= 0.041964601124292816
# Values for the y-intercept and its associated uncertainty
```

In [7]:
```
uncertainty_Frequency = uncert_Frequency(m, Δm, x, Δx, c, Δc, freqs_obs)
```

We can then **Graph the experiment values and the line of best fit**

In [9]:
```
plt.errorbar(voltages,freqs_obs , uncertainty_Frequency, Δx,fmt='o', markersize= 2, color = 'black')

def fitfunc(x,m,c):
 return m*x+c
#Plot line of best fit

pars, cov = curve_fit(fitfunc,voltages,freqs_obs)
plt.plot(voltages,fitfunc(voltages,*pars),'r-')

plt.xlabel("Input Voltages [V]")
plt.ylabel("Frequencies [Hz]")
plt.title("Frequencies Versus Input Voltages")

plt.grid(True)

plt.legend(['Line of Best Fit','Points'],loc='upper left')
# Plots the legend

print("The slope of the line of best fit is", pars[0], "\n")
print("The y-intercept of the line of best fit is", pars[1], "\n")

print("Covariance Matrix \n", cov, "\n")
# Gives the Covariance matrix

# Removes the top axis and right axis
ax = plt.subplot(111)
ax.spines["right"].set_visible(False)
ax.spines["top"].set_visible(False)
```

```
The slope of the line of best fit is 64.06859542483018

The y-intercept of the line of best fit is 6.835741220674732

Covariance Matrix
 [[ 0.00021781 -0.00053364]
 [-0.00053364  0.00176103]]
```



Using the equation of this line of best fit we can find the corresponding frequencies to our input volatges

Equation of the line of best fit is y = 64.06859542x + 6.83574122 , where x is the input voltage and y is the corresponding frequency

Using the equation of this line of best fit we can find the corresponding frequencies to our input volatges

Equation of the line of best fit is y = 64.06859542x + 6.83574122 , where x is the input voltage and y is the corresponding frequency

In [10]:
```
print("The uncertainty on the slope is: ", (cov[0,0])**0.5)
print("\nThe uncertainty on the y-intercpt is: ", (cov[1,1])**0.5)
```

```
The uncertainty on the slope is:  0.014758529469518425

The uncertainty on the y-intercpt is:  0.041964601124292816
```

To find the **Uncertainty on the Frequency** we add the uncertainty on each frequency in quadrature:

In [11]:
```
avg_uncertainty_frequency = (0.09470346**2 + 0.1153494 **2 + 0.14560479**2 + 0.1798577 **2 + 0.21524308**2 + 0.25219212**2 +
                             0.289415  **2 + 0.32713344**2 + 0.36495625**2 + 0.40342711**2 + 0.44254816**2 + 0.48178071**2
                             + 0.52085123**2 + 0.55961876**2 + 0.59893868**2 + 0.6375608 **2 + 0.67628332**2 + 0.71472884**2
                             + 0.75344878**2 + 0.79274731**2 + 0.83225013**2 + 0.87127592**2 + 0.91099465**2 + 0.95010937**2
                             + 0.98979003**2 + 1.0289875 **2 + 1.06831704**2 + 1.10241949**2 + 1.14712207**2 + 1.18653444**2
                             + 1.22570631**2 + 1.26494499**2 + 1.30394225**2 + 1.34282083**2 + 1.38250336**2 + 1.42151267**2
                             + 1.46040231**2 + 1.49935661**2 + 1.53892891**2 + 1.57930342**2 + 1.61863464**2 + 1.65815263**2
                             + 1.69699621**2 + 1.73639526**2 + 1.77555016**2 + 1.8152602 **2 + 1.85460287**2 + 1.89419299**2
                             + 1.93347709**2 + 1.97374621**2)**(1/2) / (len(uncertainty_Frequency))
```

In [12]:
```
print("The uncertianty on frequency is calculated as", avg_uncertainty_frequency, "Hz")
```

```
The uncertianty on frequency is calculated as 0.1637956366438766 Hz
```

Aluminium

Rod Readings

In [13]:
```
# Input voltages into the rod
alum_inp_volt = np.arange(2.45,2.65,0.001)

alum_frequency_of_driver = 64.06859542*alum_inp_volt + 6.83574122
# Gives the input voltage as the corresponding frequency
```

In [14]:
```
# Amplitudes received at 303K
alum_amplitude_at_303 = np.array([0.2540923683519833, 0.2612297694179393, 0.2676024489411146, 0.2701515207503844, 0.271935871016
```

In [15]:
```
# Amplitudes received at 333K
alum_amplitude_at_333 = np.array([0.2941127957575221, 0.28774011623434725, 0.2772889218163403, 0.2711711494740923, 0.269131892026
```

In [16]:
```
# Amplitudes received at 363K
alum_amplitude_at_363 = np.array([0.2913088167673251, 0.29028918804361725, 0.28315178697766114, 0.2783085505400483, 0.2747398500
```

In [17]:
```
# Amplitudes received at 393K
alum_amplitude_at_393 = np.array([0.2869753946915662, 0.284426322882296, 0.2793281792637564, 0.27473985000707, 0.2744849428261431
```

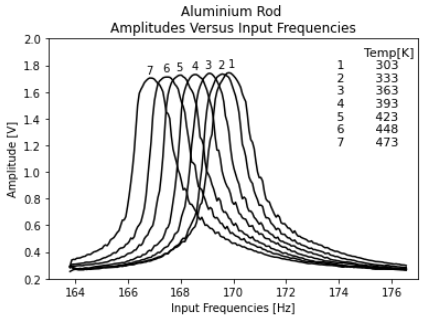
In [18]:
```
# Amplitudes received at 423K
alum_amplitude_at_423 = np.array([0.2821321582539532, 0.29258335267196023, 0.29640696038586495, 0.2936029813956681, 0.2969167747
```

In [19]:
```
# Amplitudes received at 448K
alum_amplitude_at_448 = np.array([0.2887597449580551, 0.3066032476229449, 0.3083875978894338, 0.3086425050703607, 0.3104268553368
```

In [20]:
```
# Amplitudes received at 473K
alum_amplitude_at_473 = np.array([0.2933480742147411, 0.3402509955053086, 0.3448393247619944, 0.34381969603828627, 0.353251261732
```

In [21]:
```
# Plots each temperature's frequency as a function of amplitude
plt.plot(alum_frequency_of_driver, alum_amplitude_at_303, color = 'black')
plt.plot(alum_frequency_of_driver, alum_amplitude_at_333, color = 'black')
plt.plot(alum_frequency_of_driver, alum_amplitude_at_363, color = 'black')
plt.plot(alum_frequency_of_driver, alum_amplitude_at_393, color = 'black' )
plt.plot(alum_frequency_of_driver, alum_amplitude_at_423, color = 'black' )
plt.plot(alum_frequency_of_driver, alum_amplitude_at_448, color = 'black' )
plt.plot(alum_frequency_of_driver, alum_amplitude_at_473, color = 'black' )

# Shows the table with the various temperatures
plt.text(173.8,1.2,'        Temp[K]\n 1        303\n 2        333\n 3        363\n 4        '
         '393\n 5        423\n 6        448\n 7        473',
         color='black', size= 11)

# Shows the label on each plot
plt.text(169.8, 1.79, '1')
plt.text(169.4, 1.78, '2')
plt.text(168.9, 1.78, '3')
plt.text(168.4, 1.77, '4')
plt.text(167.8, 1.76, '5')
plt.text(167.3, 1.75, '6')
plt.text(166.7, 1.74, '7')

plt.xlabel("Input Frequencies [Hz]")
plt.ylabel("Amplitude [V]")
plt.title("Aluminium Rod \nAmplitudes Versus Input Frequencies")

plt.xlim(163,177)
plt.ylim(0.2,2)
```

Out[21]: (0.2, 2.0)



Lorentzian Fitting

In [22]:
```
# Lorentzian parameter fitting for 303
def lorentzian(x, a, x0):
    return (max(alum_amplitude_at_303)/ np.pi) * a / ((x-x0)**2 + a**2)

pguess = [1 / (np.pi * max(alum_amplitude_at_303)), sum(alum_frequency_of_driver * alum_amplitude_at_303) /
          sum(alum_amplitude_at_303)]

# Fit the data
popt, pcov = curve_fit(lorentzian, alum_frequency_of_driver, alum_amplitude_at_303, p0 = pguess)

# Results
alum_Δw_303, alum_w0_303 = 2*popt[0], popt[1]
# Gives the frequency at the maximum amplitude which is the resonant frequecy and also gives the full
# bandwidth at half maximum amplitude as a is half the bandwidth


# Reference for this code. I use this same code for each lorentzian function.

# [40] B. Prudholm (2014) Fit points to a Lorentzian curve and find center and half
# maximum bandwidth in Python. [online] Stack Overflow.
# Available at: <https://stackoverflow.com/questions/24437070/fit-points-to-a-lorentzian-
# curve-and-find-center-and-half-maximum-bandwidth-in-p> [Accessed 3 October 2022].
```

In [23]:
```
# Lorentzian parameter fitting for 333
def lorentzian(x, a, x0):
    return (max(alum_amplitude_at_333)/ np.pi) * a / ((x-x0)**2 + a**2)

pguess = [1 / (np.pi * max(alum_amplitude_at_333)), sum(alum_frequency_of_driver * alum_amplitude_at_333) /
          sum(alum_amplitude_at_333)]

# Fit the data
popt, pcov = curve_fit(lorentzian, alum_frequency_of_driver, alum_amplitude_at_333, p0 = pguess)

# Results
alum_Δw_333, alum_w0_333 = 2*popt[0], popt[1]
# Gives the frequency at the maximum amplitude which is the resonant frequecy and also gives bandwidth at half maximum amplitude
```

In [24]:
```
# Lorentzian parameter fitting for 363
def lorentzian(x, a, x0):
    return (max(alum_amplitude_at_363)/ np.pi) * a / ((x-x0)**2 + a**2)

pguess = [1 / (np.pi * max(alum_amplitude_at_363)), sum(alum_frequency_of_driver * alum_amplitude_at_363) /
          sum(alum_amplitude_at_363)]

# Fit the data
popt, pcov = curve_fit(lorentzian, alum_frequency_of_driver, alum_amplitude_at_363, p0 = pguess)

# Results
alum_Δw_363, alum_w0_363 = 2*popt[0], popt[1]
# Gives the frequency at the maximum amplitude which is the resonant frequecy and also gives bandwidth at half maximum amplitude
```

In [25]:
```
# Lorentzian parameter fitting for 393
def lorentzian(x, a, x0):
    return (max(alum_amplitude_at_393)/ np.pi) * a / ((x-x0)**2 + a**2)

pguess = [1 / (np.pi * max(alum_amplitude_at_393)), sum(alum_frequency_of_driver * alum_amplitude_at_393) /
          sum(alum_amplitude_at_393)]

# Fit the data
popt, pcov = curve_fit(lorentzian, alum_frequency_of_driver, alum_amplitude_at_393, p0 = pguess)

# Results
alum_Δw_393, alum_w0_393 = 2*popt[0], popt[1]
# Gives the frequency at the maximum amplitude which is the resonant frequecy and also gives bandwidth at half maximum amplitude
```

In [26]:
```
# Lorentzian parameter fitting for 423
def lorentzian(x, a, x0):
    return (max(alum_amplitude_at_423)/ np.pi) * a / ((x-x0)**2 + a**2)

pguess = [1 / (np.pi * max(alum_amplitude_at_423)), sum(alum_frequency_of_driver * alum_amplitude_at_423) /
          sum(alum_amplitude_at_423)]

# Fit the data
popt, pcov = curve_fit(lorentzian, alum_frequency_of_driver, alum_amplitude_at_423, p0 = pguess)

# Results
alum_Δw_423, alum_w0_423 = 2*popt[0], popt[1]
# Gives the frequency at the maximum amplitude which is the resonant frequecy and also gives bandwidth at half maximum amplitude
```

In [27]:
```
# Lorentzian parameter fitting for 448
def lorentzian(x, a, x0):
    return (max(alum_amplitude_at_448)/ np.pi) * a / ((x-x0)**2 + a**2)

pguess = [1 / (np.pi * max(alum_amplitude_at_448)), sum(alum_frequency_of_driver * alum_amplitude_at_448) /
          sum(alum_amplitude_at_448)]

# Fit the data
popt, pcov = curve_fit(lorentzian, alum_frequency_of_driver, alum_amplitude_at_448, p0 = pguess)

# Results
alum_Δw_448, alum_w0_448 = 2*popt[0], popt[1]
# Gives the frequency at the maximum amplitude which is the resonant frequecy and also gives bandwidth at half maximum amplitude
```

In [28]:
```
# Lorentzian parameter fitting for 473
def lorentzian(x, a, x0):
    return (max(alum_amplitude_at_473)/ np.pi) * a / ((x-x0)**2 + a**2)

pguess = [1 / (np.pi * max(alum_amplitude_at_473 )), sum(alum_frequency_of_driver * alum_amplitude_at_473 ) /
          sum(alum_amplitude_at_473 )]

# Fit the data
popt, pcov = curve_fit(lorentzian, alum_frequency_of_driver, alum_amplitude_at_473 , p0 = pguess)

# Results
alum_Δw_473 , alum_w0_473  = 2*popt[0], popt[1]
# Gives the frequency at the maximum amplitude which is the resonant frequecy and also gives bandwidth at half maximum amplitude
```

Brass

Rod Readings

In [29]:
```
# Input voltages into the rod
brs_inp_volt = np.arange(2.49,2.8,0.001)

brs_frequency_of_driver = 64.06859542*brs_inp_volt + 6.83574122
# Gives the input voltage as the corresponding frequency
```

In [30]:
```
# Amplitudes received at 303K
brs_amplitude_at_303 = np.array([0.29615205320493787, 0.2969167747477188, 0.284426322882296, 0.27983799362561024, 0.271171149474
```

In [31]:
```
# Amplitudes received at 333K
brs_amplitude_at_333 = np.array([0.2969167747477188, 0.2969167747477187, 0.2907990024054713, 0.28493613724415, 0.2775438289972672
```

In [32]:
```
# Amplitudes received at 363K
brs_amplitude_at_363 = np.array([0.30277963990904, 0.3101719481559228, 0.30328945427089404, 0.29513242448123, 0.29028918804361725
```

In [33]:
```
# Amplitudes received at 393K
brs_amplitude_at_393 = np.array([0.299720753737916, 0.30889741225128786, 0.307622876346653, 0.30150510400440506, 0.29487751730030
```

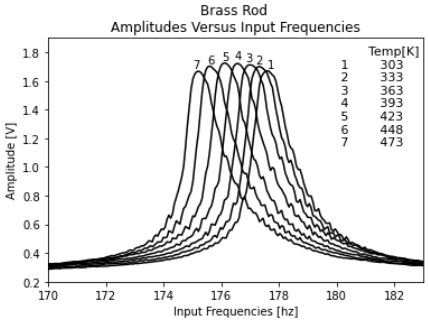
In [34]:
```
# Amplitudes received at 423K
brs_amplitude_at_423 = np.array([0.293857888576595, 0.3040541758136751, 0.30252473272811303, 0.29538733166215697, 0.2905440952245
```

In [35]:
```
# Amplitudes received at 448K
brs_amplitude_at_448 = np.array([0.2907990024054712, 0.3043090829946021, 0.3037992686327481, 0.30252473272811314, 0.2987011250142
```

In [36]:
```
# Amplitudes received at 473K
brs_amplitude_at_473 = np.array([0.2854459516060041, 0.29870112501420776, 0.29615205320493787, 0.2938578885765951, 0.292328445491
```

In [37]:
```
# Plots each temperature's frequency as a function of amplitude
plt.plot(brs_frequency_of_driver, brs_amplitude_at_303, color = 'black')
plt.plot(brs_frequency_of_driver, brs_amplitude_at_333, color = 'black')
plt.plot(brs_frequency_of_driver, brs_amplitude_at_363, color = 'black')
plt.plot(brs_frequency_of_driver, brs_amplitude_at_393, color = 'black' )
plt.plot(brs_frequency_of_driver, brs_amplitude_at_423, color = 'black' )
plt.plot(brs_frequency_of_driver, brs_amplitude_at_448, color = 'black' )
plt.plot(brs_frequency_of_driver, brs_amplitude_at_473, color = 'black' )

# Shows the table with the various temperatures
plt.text(180,1.15,'        Temp[K]\n 1        303\n 2        333\n 3        363\n 4        '
         '393\n 5        423\n 6        448\n 7        473', color='black', size= 11)

# Shows the label on each plot
plt.text(177.6, 1.69, '1')
plt.text(177.2, 1.72, '2')
plt.text(176.85, 1.74, '3')
plt.text(176.45, 1.75, '4')
plt.text(176, 1.745, '5')
plt.text(175.5, 1.72, '6')
plt.text(175, 1.69, '7')


plt.xlabel("Input Frequencies [hz]")
plt.ylabel("Amplitude [V]")
plt.title("Brass Rod \nAmplitudes Versus Input Frequencies")

plt.xlim(170,183)
plt.ylim(0.2,1.9)
```

Out[37]: (0.2, 1.9)



Lorentzian Fitting

In [38]:
```
# Lorentzian parameter fitting for 303
def lorentzian(x, a, x0):
    return (max(brs_amplitude_at_303)/ np.pi) * a / ((x-x0)**2 + a**2)

pguess = [1 / (np.pi * max(brs_amplitude_at_303)), sum(brs_frequency_of_driver * brs_amplitude_at_303) /
          sum(brs_amplitude_at_303)]

# Fit the data
popt, pcov = curve_fit(lorentzian, brs_frequency_of_driver, brs_amplitude_at_303, p0 = pguess)

# Results
brs_Δw_303, brs_w0_303 = 2*popt[0], popt[1]
# Gives the frequency at the maximum amplitude which is the resonant frequecy and also gives bandwidth at half maximum amplitude
```

In [39]:
```
# Lorentzian parameter fitting for 333
def lorentzian(x, a, x0):
    return (max(brs_amplitude_at_333)/ np.pi) * a / ((x-x0)**2 + a**2)

pguess = [1 / (np.pi * max(brs_amplitude_at_333)), sum(brs_frequency_of_driver * brs_amplitude_at_333) /
          sum(brs_amplitude_at_333)]

# Fit the data
popt, pcov = curve_fit(lorentzian, brs_frequency_of_driver, brs_amplitude_at_333, p0 = pguess)

# Results
brs_Δw_333, brs_w0_333 = 2*popt[0], popt[1]
# Gives the frequency at the maximum amplitude which is the resonant frequecy and also gives bandwidth at half maximum amplitude
```

In [40]:
```
# Lorentzian parameter fitting for 363
def lorentzian(x, a, x0):
    return (max(brs_amplitude_at_363)/ np.pi) * a / ((x-x0)**2 + a**2)

pguess = [1 / (np.pi * max(brs_amplitude_at_363)), sum(brs_frequency_of_driver * brs_amplitude_at_363) /
          sum(brs_amplitude_at_363)]

# Fit the data
popt, pcov = curve_fit(lorentzian, brs_frequency_of_driver, brs_amplitude_at_363, p0 = pguess)

# Results
brs_Δw_363, brs_w0_363 = 2*popt[0], popt[1]
# Gives the frequency at the maximum amplitude which is the resonant frequecy and also gives bandwidth at half maximum amplitude
```

In [41]:
```
# Lorentzian parameter fitting for 393
def lorentzian(x, a, x0):
    return (max(brs_amplitude_at_393)/ np.pi) * a / ((x-x0)**2 + a**2)

pguess = [1 / (np.pi * max(brs_amplitude_at_393)), sum(brs_frequency_of_driver * brs_amplitude_at_393) /
          sum(brs_amplitude_at_393)]

# Fit the data
popt, pcov = curve_fit(lorentzian, brs_frequency_of_driver, brs_amplitude_at_393, p0 = pguess)

# Results
brs_Δw_393, brs_w0_393 = 2*popt[0], popt[1]
# Gives the frequency at the maximum amplitude which is the resonant frequecy and also gives bandwidth at half maximum amplitude
```

In [42]:
```
# Lorentzian parameter fitting for 423
def lorentzian(x, a, x0):
    return (max(brs_amplitude_at_423)/ np.pi) * a / ((x-x0)**2 + a**2)

pguess = [1 / (np.pi * max(brs_amplitude_at_423)), sum(brs_frequency_of_driver * brs_amplitude_at_423) /
          sum(brs_amplitude_at_423)]

# Fit the data
popt, pcov = curve_fit(lorentzian, brs_frequency_of_driver, brs_amplitude_at_423, p0 = pguess)

# Results
brs_Δw_423, brs_w0_423 = 2*popt[0], popt[1]
# Gives the frequency at the maximum amplitude which is the resonant frequecy and also gives bandwidth at half maximum amplitude
```

In [43]:
```
# Lorentzian parameter fitting for 448
def lorentzian(x, a, x0):
    return (max(brs_amplitude_at_448)/ np.pi) * a / ((x-x0)**2 + a**2)

pguess = [1 / (np.pi * max(brs_amplitude_at_448)), sum(brs_frequency_of_driver * brs_amplitude_at_448) /
          sum(brs_amplitude_at_448)]

# Fit the data
popt, pcov = curve_fit(lorentzian, brs_frequency_of_driver, brs_amplitude_at_448, p0 = pguess)

# Results
brs_Δw_448, brs_w0_448 = 2*popt[0], popt[1]
# Gives the frequency at the maximum amplitude which is the resonant frequecy and also gives bandwidth at half maximum amplitude
```

In [44]:
```
# Lorentzian parameter fitting for 473
def lorentzian(x, a, x0):
    return (max(brs_amplitude_at_473)/ np.pi) * a / ((x-x0)**2 + a**2)

pguess = [1 / (np.pi * max(brs_amplitude_at_473)), sum(brs_frequency_of_driver * brs_amplitude_at_473) /
          sum(brs_amplitude_at_473)]

# Fit the data
popt, pcov = curve_fit(lorentzian, brs_frequency_of_driver, brs_amplitude_at_473, p0 = pguess)

# Results
brs_Δw_473, brs_w0_473 = 2*popt[0], popt[1]
# Gives the frequency at the maximum amplitude which is the resonant frequecy and also gives bandwidth at half maximum amplitude
```

Steel

Rod Readings

In [45]:
```
# Input voltages into the rod
ste_inp_volt = np.arange(3.8,4.15,0.001)

ste_frequency_of_driver = 64.06859542*ste_inp_volt + 6.83574122
# Gives the input voltage as the corresponding frequency
```

In [46]:
```
# Amplitudes received at 303K
ste_amplitude_at_303 = np.array([0.24287645239119585, 0.24262154521026877, 0.23905284467729054, 0.24007247340099847, 0.2375234015
```

In [47]:
```
# Amplitudes received at 333K
ste_amplitude_at_333 = np.array([0.2454255242004658, 0.24185682366748756, 0.24134700930563363, 0.2431313595721229, 0.246445152924
```

In [48]:
```
# Amplitudes received at 363K
ste_amplitude_at_363 = np.array([0.24134700930563352, 0.24083719494377948, 0.2398175662200714, 0.2398175662200714, 0.241856823667
```

In [49]:
```
# Amplitudes received at 393K
ste_amplitude_at_393 = np.array([0.2464451529241737, 0.2489942247334435, 0.2489942247334435, 0.24848441037158953, 0.2492491319143
```

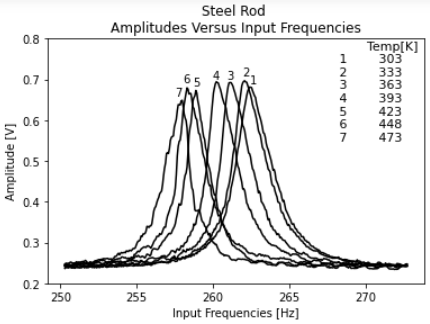
In [50]:
```
# Amplitudes received at 423K
ste_amplitude_at_423 = np.array([0.23650377286802068, 0.2398175662200714, 0.2385430303154365, 0.2387979374963635, 0.239817566220
```

In [51]:
```
# Amplitudes received at 448K
ste_amplitude_at_448 = np.array([0.24568043138139278, 0.24797459600973557, 0.2441509882958309, 0.24364117393397694, 0.2436411739
```

In [52]:
```
# Amplitudes received at 473K
ste_amplitude_at_473 = np.array([0.2433862667530499, 0.24695496728602767, 0.24695496728602767, 0.2444058954767579, 0.244150988295
```

In [53]:
```
# Plots each temperature's frequency as a function of amplitude
plt.plot(ste_frequency_of_driver, ste_amplitude_at_303, color = 'black')
plt.plot(ste_frequency_of_driver, ste_amplitude_at_333, color = 'black')
plt.plot(ste_frequency_of_driver, ste_amplitude_at_363, color = 'black')
plt.plot(ste_frequency_of_driver, ste_amplitude_at_393, color = 'black' )
plt.plot(ste_frequency_of_driver, ste_amplitude_at_423, color = 'black' )
plt.plot(ste_frequency_of_driver, ste_amplitude_at_448, color = 'black' )
plt.plot(ste_frequency_of_driver, ste_amplitude_at_473, color = 'black' )

# Shows the table with the various temperatures
plt.text(268,0.55,'        Temp[K]\n 1        303\n 2        333\n 3        363\n 4        '
         '393\n 5        423\n 6        448\n 7        473', color='black', size= 11)

# Shows the label on each plot
plt.text(262.4, 0.69, '1')
plt.text(261.9, 0.71, '2')
plt.text(260.9, 0.7, '3')
plt.text(260, 0.7, '4')
plt.text(258.7, 0.685, '5')
plt.text(258, 0.693, '6')
plt.text(257.5, 0.66, '7')

plt.xlabel("Input Frequencies [Hz]")
plt.ylabel("Amplitude [V]")
plt.title("Steel Rod \nAmplitudes Versus Input Frequencies")

plt.ylim(0.2,0.8)
```

Out[53]: (0.2, 0.8)



Lorentzian Fitting

In [54]:
```
# Lorentzian parameter fitting for 303
def lorentzian(x, a, x0):
    return (max(ste_amplitude_at_303)/ np.pi) * a / ((x-x0)**2 + a**2)

pguess = [1 / (np.pi * max(ste_amplitude_at_303)), sum(ste_frequency_of_driver * ste_amplitude_at_303) /
          sum(ste_amplitude_at_303)]

# Fit the data
popt, pcov = curve_fit(lorentzian, ste_frequency_of_driver, ste_amplitude_at_303, p0 = pguess)

# Results
ste_Δw_303, ste_w0_303 = 2*popt[0], popt[1]
# Gives the frequency at the maximum amplitude which is the resonant frequecy and also gives bandwidth at half maximum amplitude
```

In [55]:
```
# Lorentzian parameter fitting for 333
def lorentzian(x, a, x0):
    return (max(ste_amplitude_at_333)/ np.pi) * a / ((x-x0)**2 + a**2)

pguess = [1 / (np.pi * max(ste_amplitude_at_333)), sum(ste_frequency_of_driver * ste_amplitude_at_333) /
          sum(ste_amplitude_at_333)]

# Fit the data
popt, pcov = curve_fit(lorentzian, ste_frequency_of_driver, ste_amplitude_at_333, p0 = pguess)

# Results
ste_Δw_333, ste_w0_333 = 2*popt[0], popt[1]
# Gives the frequency at the maximum amplitude which is the resonant frequecy and also gives bandwidth at half maximum amplitude
```

In [56]:
```
# Lorentzian parameter fitting for 363
def lorentzian(x, a, x0):
    return (max(ste_amplitude_at_363)/ np.pi) * a / ((x-x0)**2 + a**2)

pguess = [1 / (np.pi * max(ste_amplitude_at_363)), sum(ste_frequency_of_driver * ste_amplitude_at_363) /
          sum(ste_amplitude_at_363)]

# Fit the data
popt, pcov = curve_fit(lorentzian, ste_frequency_of_driver, ste_amplitude_at_363, p0 = pguess)

# Results
ste_Δw_363, ste_w0_363 = 2*popt[0], popt[1]
# Gives the frequency at the maximum amplitude which is the resonant frequecy and also gives bandwidth at half maximum amplitude
```

In [57]:
```
# Lorentzian parameter fitting for 393
def lorentzian(x, a, x0):
    return (max(ste_amplitude_at_393)/ np.pi) * a / ((x-x0)**2 + a**2)

pguess = [1 / (np.pi * max(ste_amplitude_at_393)), sum(ste_frequency_of_driver * ste_amplitude_at_393) /
          sum(ste_amplitude_at_393)]

# Fit the data
popt, pcov = curve_fit(lorentzian, ste_frequency_of_driver, ste_amplitude_at_393, p0 = pguess)

# Results
ste_Δw_393, ste_w0_393 = 2*popt[0], popt[1]
# Gives the frequency at the maximum amplitude which is the resonant frequecy and also gives bandwidth at half maximum amplitude
```

In [58]:
```
# Lorentzian parameter fitting for 423
def lorentzian(x, a, x0):
    return (max(ste_amplitude_at_423)/ np.pi) * a / ((x-x0)**2 + a**2)

pguess = [1 / (np.pi * max(ste_amplitude_at_423)), sum(ste_frequency_of_driver * ste_amplitude_at_423) /
          sum(ste_amplitude_at_423)]

# Fit the data
popt, pcov = curve_fit(lorentzian, ste_frequency_of_driver, ste_amplitude_at_423, p0 = pguess)

# Results
ste_Δw_423, ste_w0_423 = 2*popt[0], popt[1]
# Gives the frequency at the maximum amplitude which is the resonant frequecy and also gives bandwidth at half maximum amplitude
```

In [59]:
```
# Lorentzian parameter fitting for 448
def lorentzian(x, a, x0):
    return (max(ste_amplitude_at_448)/ np.pi) * a / ((x-x0)**2 + a**2)

pguess = [1 / (np.pi * max(ste_amplitude_at_448)), sum(ste_frequency_of_driver * ste_amplitude_at_448) /
          sum(ste_amplitude_at_448)]

# Fit the data
popt, pcov = curve_fit(lorentzian, ste_frequency_of_driver, ste_amplitude_at_448, p0 = pguess)

# Results
ste_Δw_448, ste_w0_448 = 2*popt[0], popt[1]
# Gives the frequency at the maximum amplitude which is the resonant frequecy and also gives bandwidth at half maximum amplitude
```

In [60]:
```
# Lorentzian parameter fitting for 473
def lorentzian(x, a, x0):
    return (max(ste_amplitude_at_473)/ np.pi) * a / ((x-x0)**2 + a**2)

pguess = [1 / (np.pi * max(ste_amplitude_at_473)), sum(ste_frequency_of_driver * ste_amplitude_at_473) /
          sum(ste_amplitude_at_473)]

# Fit the data
popt, pcov = curve_fit(lorentzian, ste_frequency_of_driver, ste_amplitude_at_473, p0 = pguess)

# Results
ste_Δw_473, ste_w0_473 = 2*popt[0], popt[1]
# Gives the frequency at the maximum amplitude which is the resonant frequecy and also gives bandwidth at half maximum amplitude
```

Lorentzian Curve Fitting onto Data

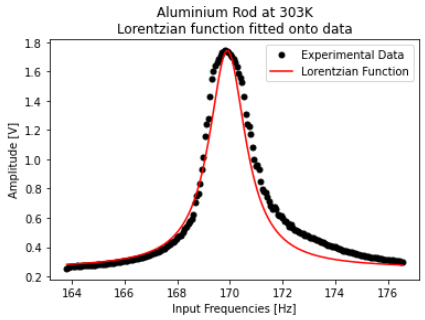
Using the paramters found in the above code we can **Fit Lorentzian Functions** onto our graphs to test how well they match.

Aluminium at temperature 303K

In [61]:
```
y_alum = ((max(alum_amplitude_at_303)-min(alum_amplitude_at_303) )
          / ( 1+ ((alum_frequency_of_driver - alum_w0_303)/(alum_Δw_303))**2) + min(alum_amplitude_at_303) )
```

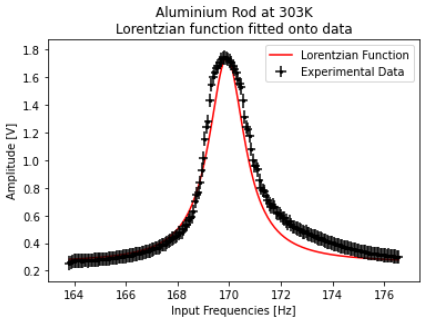
In [62]:
```
plt.plot(alum_frequency_of_driver, alum_amplitude_at_303, 'o', color = 'black', markersize =5)
plt.plot(alum_frequency_of_driver , y_alum, color = 'red')

plt.legend(['Experimental Data', 'Lorentzian Function'])

plt.xlabel("Input Frequencies [Hz]")
plt.ylabel("Amplitude [V]")
plt.title("Aluminium Rod at 303K\nLorentzian function fitted onto data")
```

Out[62]:
```
Text(0.5, 1.0, 'Aluminium Rod at 303K\nLorentzian function fitted onto data')
```



In [63]:
```
plt.plot(alum_frequency_of_driver, alum_amplitude_at_303, 'o', color = 'black', markersize = 2)
plt.plot(alum_frequency_of_driver , y_alum, color = 'red', label ='Lorentzian Function')

plt.errorbar(alum_frequency_of_driver, alum_amplitude_at_303, 0.05, 0.16,capsize=0, fmt='o', markersize = 2,
             color = 'black', label = "Experimental Data")

plt.legend(loc = 'upper right')

plt.xlabel("Input Frequencies [Hz]")
plt.ylabel("Amplitude [V]")
plt.title("Aluminium Rod at 303K\nLorentzian function fitted onto data")
```

Out[63]:
```
Text(0.5, 1.0, 'Aluminium Rod at 303K\nLorentzian function fitted onto data')
```

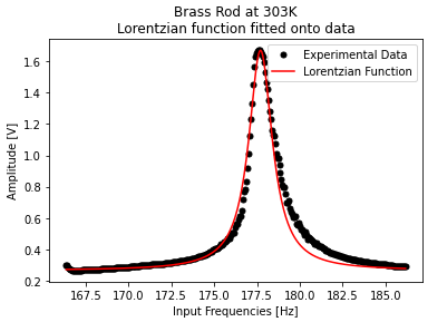


Brass at temperature 303K

In [64]:
```
y_brs = ((max(brs_amplitude_at_303)-min(brs_amplitude_at_303) ) /
         ( 1+ ((brs_frequency_of_driver - brs_w0_303)/(brs_Δw_303))**2) + min(brs_amplitude_at_303) )
```

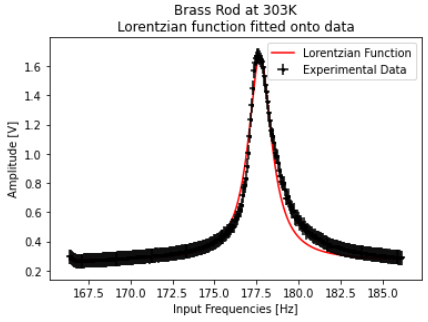
In [65]:
```
plt.plot(brs_frequency_of_driver, brs_amplitude_at_303, 'o', color = 'black', markersize =5)
plt.plot(brs_frequency_of_driver , y_brs, color = 'red')

plt.legend(['Experimental Data', 'Lorentzian Function'])

plt.xlabel("Input Frequencies [Hz]")
plt.ylabel("Amplitude [V]")
plt.title("Brass Rod at 303K\nLorentzian function fitted onto data")
```

Out[65]:
```
Text(0.5, 1.0, 'Brass Rod at 303K\nLorentzian function fitted onto data')
```



In [66]:
```
plt.plot(brs_frequency_of_driver, brs_amplitude_at_303, 'o', color = 'black', markersize = 2)
plt.plot(brs_frequency_of_driver , y_brs, color = 'red', label ='Lorentzian Function')

plt.errorbar(brs_frequency_of_driver, brs_amplitude_at_303, 0.05, 0.16,capsize=0, fmt='o', markersize = 2,
             color = 'black', label = "Experimental Data")

plt.legend(loc = 'upper right')

plt.xlabel("Input Frequencies [Hz]")
plt.ylabel("Amplitude [V]")
plt.title("Brass Rod at 303K\nLorentzian function fitted onto data")
```

Out[66]:
```
Text(0.5, 1.0, 'Brass Rod at 303K\nLorentzian function fitted onto data')
```

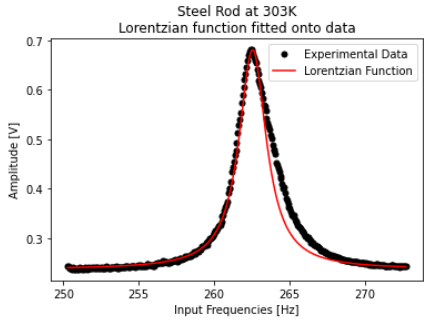


Steel at temperature 303K

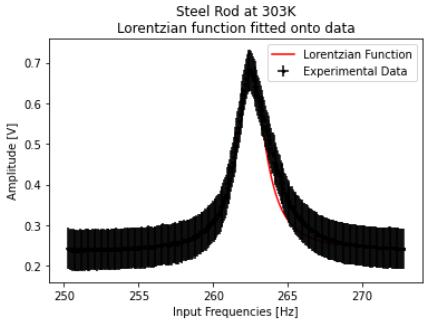
In [67]:
```
y_ste = ((max(ste_amplitude_at_303)-min(ste_amplitude_at_303) ) /
         ( 1+ ((ste_frequency_of_driver - ste_w0_303)/(ste_Δw_303))**2) + min(ste_amplitude_at_303) )
```

In [68]:
```
plt.plot(ste_frequency_of_driver, ste_amplitude_at_303, 'o', color = 'black', markersize =5)
plt.plot(ste_frequency_of_driver , y_ste, color = 'red')

plt.legend(['Experimental Data', 'Lorentzian Function'])

plt.xlabel("Input Frequencies [Hz]")
plt.ylabel("Amplitude [V]")
plt.title("Steel Rod at 303K\nLorentzian function fitted onto data")
```

Out[68]:
```
Text(0.5, 1.0, 'Steel Rod at 303K\nLorentzian function fitted onto data')
```



In [69]:
```
plt.plot(ste_frequency_of_driver, ste_amplitude_at_303, 'o', color = 'black', markersize = 2)
plt.plot(ste_frequency_of_driver , y_ste, color = 'red', label ='Lorentzian Function')

plt.errorbar(ste_frequency_of_driver, ste_amplitude_at_303, 0.05, 0.16,capsize=0, fmt='o', markersize = 2,
             color = 'black', label = "Experimental Data")

plt.legend(loc = 'upper right')

plt.xlabel("Input Frequencies [Hz]")
plt.ylabel("Amplitude [V]")
plt.title("Steel Rod at 303K\nLorentzian function fitted onto data")
```



Determining Modulus of Rigidity

Determine moment of inertia of auxillary body

$I = ml^2$ where m is the mass of the auxillary body and l is the length of auxillary body

In [70]:
```
m = 0.05501
# mass of auxilary body in kg

l = (9.5*10**-3)/2
# radius of the auxilliary body which correpsonds to the distance from the axis

I = m*l**2
# Formula for moment of inertia
```

In [71]:
```
I
```

Out[71]:
```
1.241163125e-06
```

The calculation for the uncertainty on the moment of inertia is calculated as follows:

In [72]:
```
I = 1.241163125e-06

m = 55.01
Δm = 0.01

l = 0.00475
Δl = 0.0001
```

In [73]:
```
# Using the propogation of errors in equation 18 of lab report:
ΔI = I*((Δm/m)**2+(2*Δl/l)**2)**(1/2)
```

In [74]:
```
ΔI
# Gives the uncertainty on the calculated value for moment of inertia
```

Out[74]:
```
5.225998705406103e-08
```

Determine the Modulus of Rigidity and Graph it

$G = \frac{2Il\omega_0^2}{\pi r^4}$ where I = inertia of auxillary body, l = length of rod, ω_0 is resonant frequency and r is radius of the rod

In [75]:
```
def mod_of_rigid(I,l,w0,r):
    return (2*I*l*(w0*2*np.pi)**2)/(np.pi*r**4)
# Function to find the modulus of rigidity
```

In [76]:
```
# Value for Aluminium Rod
alum_l = 0.355
alum_r = 1.5*10**-3

# Value for Brass Rod
brs_l = 0.358
brs_r = 1.5*10**-3

# Value for Steel Rod
ste_l = 0.349
ste_r = 1.5*10**-3
```

In [77]:
```
# Measure of G for each rod at each temperature

# Aluminium Readings
G_303_alum = mod_of_rigid(I, alum_l,  alum_w0_303, alum_r )
G_333_alum = mod_of_rigid(I, alum_l,  alum_w0_333, alum_r)
G_363_alum = mod_of_rigid(I, alum_l,  alum_w0_363, alum_r)
G_393_alum = mod_of_rigid(I, alum_l,  alum_w0_393, alum_r)
G_423_alum = mod_of_rigid(I, alum_l,  alum_w0_423, alum_r)
G_448_alum = mod_of_rigid(I, alum_l,  alum_w0_448, alum_r)
G_473_alum = mod_of_rigid(I, alum_l,  alum_w0_473, alum_r)

# Brass Readings
G_303_brs = mod_of_rigid(I, brs_l,  brs_w0_303, brs_r )
G_333_brs = mod_of_rigid(I, brs_l,  brs_w0_333, brs_r)
G_363_brs = mod_of_rigid(I, brs_l,  brs_w0_363, brs_r)
G_393_brs = mod_of_rigid(I, brs_l,  brs_w0_393, brs_r)
G_423_brs = mod_of_rigid(I, brs_l,  brs_w0_423, brs_r)
G_448_brs = mod_of_rigid(I, brs_l,  brs_w0_448, brs_r)
G_473_brs = mod_of_rigid(I, brs_l,  brs_w0_473, brs_r)

# Steel Readings ste
G_303_ste = mod_of_rigid(I, ste_l,  ste_w0_303, ste_r )
G_333_ste = mod_of_rigid(I, ste_l,  ste_w0_333, ste_r)
G_363_ste = mod_of_rigid(I, ste_l,  ste_w0_363, ste_r)
G_393_ste = mod_of_rigid(I, ste_l,  ste_w0_393, ste_r)
G_423_ste = mod_of_rigid(I, ste_l,  ste_w0_423, ste_r)
G_448_ste = mod_of_rigid(I, ste_l,  ste_w0_448, ste_r)
G_473_ste = mod_of_rigid(I, ste_l,  ste_w0_473, ste_r)
```

In [78]:
```
# Puts the temperatures into an array
temperatures = np.array([303,333, 363, 393, 423, 448, 473])

# Puts the aluminium modulus of rigidity into an array
alum_G = np.array([G_303_alum, G_333_alum, G_363_alum, G_393_alum, G_423_alum, G_448_alum, G_473_alum])

# Puts the brass modulus of rigidity into an array
brs_G = np.array([G_303_brs, G_333_brs, G_363_brs, G_393_brs, G_423_brs, G_448_brs, G_473_brs])

# Puts the steel modulus of rigidity into an array
ste_G = np.array([G_303_ste, G_333_ste, G_363_ste, G_393_ste, G_423_ste, G_448_ste, G_473_ste])
```

Calculating the **Uncertainty on Modulus of Rigidity** for each value for using propogation of errors

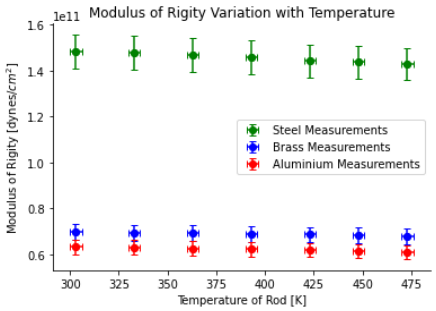
In [79]:
```
I = 1.241163125*10**-6
ΔI = 5.225998705406103*10**-8

# Values of lengths
alum_l = 0.355
alum_r = 1.5*10**-3
brs_l = 0.358
brs_r = 1.5*10**-3
ste_l = 0.349
ste_r = 1.5*10**-3
```

In [80]:
```
def uncert_G(I, ΔI, l, Δl, w0, Δw0, r, Δr, G):
    return G *(  (ΔI/I)**2 + (Δl/l)**2 + (2*(Δw0/w0))**2 + ((4*(Δr/r))**2) )**(0.5)
# Rearrangemnt of equation 12 to give us a value for uncertainty of modulus of rigidity
```

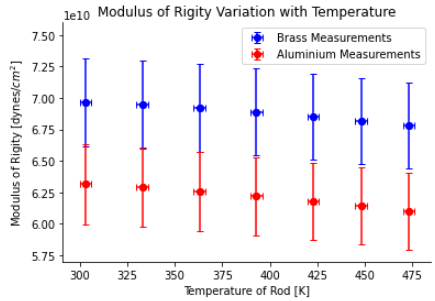
In [81]:
```
alum_w0 = np.array([alum_w0_303, alum_w0_333, alum_w0_363, alum_w0_393, alum_w0_423, alum_w0_448, alum_w0_473])
# Puts the aluminium resonant frequency for each temperature into an array

brs_w0 = np.array([brs_w0_303, brs_w0_333, brs_w0_363, brs_w0_393, brs_w0_423, brs_w0_448, brs_w0_473])
# Puts the brass resonant frequency for each temperature into an array

ste_w0 = np.array([ste_w0_303, ste_w0_333, ste_w0_363, ste_w0_393, ste_w0_423, ste_w0_448, ste_w0_473])
# Puts the steel resonant frequency for each temperature into an array
```

In [82]:
```
# Puts the uncertainty on each Aluminium rod into an array
alum_uncert_G = np.array(uncert_G(I, ΔI, alum_l, 0.001, alum_w0, 0.16, alum_r, 0.001, alum_G))

# Puts the uncertainty on each Brass rod into an array
brs_uncert_G = np.array(uncert_G(I, ΔI, brs_l, 0.001, brs_w0, 0.16, brs_r, 0.001, brs_G))

# Puts the uncertainty on each Steel rod into an array
ste_uncert_G = np.array(uncert_G(I, ΔI, ste_l, 0.001, ste_w0, 0.16, ste_r, 0.001, ste_G))
```

In [83]:
```
# Plotting the three modulus of rigidity's
plt.plot(temperatures, alum_G , 'ro')
plt.plot(temperatures, brs_G , 'bo')
plt.plot(temperatures, ste_G , 'go')

plt.xlabel("Temperature of Rod [K]")
plt.ylabel("Modulus of Rigity [dynes/$cm^2$]")
plt.title("Modulus of Rigity Variation with Temperature")
plt.grid(False)

# Plots the uncertainty bars on each modulus of rigidity
plt.errorbar(temperatures, ste_G, ste_uncert_G, 1, capsize=3, fmt='o', color = 'green',  label = "Steel Measurements")
plt.errorbar(temperatures, brs_G, brs_uncert_G, 1, capsize=3, fmt='o', color = 'blue',  label = "Brass Measurements")
plt.errorbar(temperatures, alum_G, alum_uncert_G, 1,capsize=3, fmt='o', color = 'red', label = "Aluminium Measurements")

plt.legend(loc='center right')

# Removes the top axis and right axis
ax = plt.subplot(111)
ax.spines["right"].set_visible(False)
ax.spines["top"].set_visible(False)
```



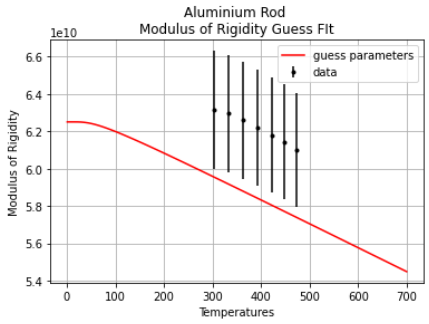
In [84]:
```
plt.plot(temperatures, alum_G , 'ro')
plt.plot(temperatures, brs_G , 'bo')

plt.xlabel("Temperature of Rod [K]")
plt.ylabel("Modulus of Rigity [dynes/$cm^2$]")
plt.grid(False)
plt.title("Modulus of Rigity Variation with Temperature")

plt.errorbar(temperatures, brs_G, brs_uncert_G, 1, capsize=3, fmt='o', color = 'blue',  label = "Brass Measurements")
plt.errorbar(temperatures, alum_G, alum_uncert_G, 1,capsize=3, fmt='o', color = 'red', label = "Aluminium Measurements")

plt.ylim(5.7*10**10, 7.6*10**10)

plt.legend(loc = 'upper right')

# Removes the top axis and right axis
ax = plt.subplot(111)
ax.spines["right"].set_visible(False)
ax.spines["top"].set_visible(False)
```



Fitting Aluminium G_0, s and t Parameters

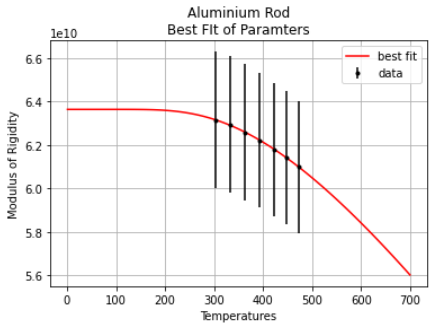
In [85]:
```
def G(x, G0, s, t):
    return G0 - ( s / (np.exp(t/x)-1) )
# Equation 12
```

In [86]:
```
# plot the raw data
plt.errorbar(temperatures, alum_G, yerr=alum_uncert_G, fmt=".", color = 'black', label = 'data')
plt.grid(True)
plt.xlabel('Temperatures')
plt.ylabel('Modulus of Rigidity')


# initial guess
pars0 = (6.25*10**10, 0.21*10**10, 163)

# overlay exponential

x = np.linspace(1, 700, 100)

plt.plot(x, G(x,*pars0), color = 'red', label = 'guess parameters')

plt.legend();
plt.title("Aluminium Rod \nModulus of Rigidity Guess FIt");
```



In [87]:
```
# perform fitting

popt, pcov = curve_fit(
    G, temperatures, alum_G,  sigma = alum_uncert_G,  absolute_sigma=True, p0=pars0
)

# extract each best-fit parameter and its error
G0_opt_alum = popt[0]
G0_opt_err_alum = np.sqrt(pcov[0, 0])
print(f"Go (best-fit) = {G0_opt_alum:.3g} ± {G0_opt_err_alum:.1g}")

s_opt_alum = popt[1]
s_opt_err_alum = np.sqrt(pcov[1, 1])
print(f"s (best-fit) = {s_opt_alum:.3g} ± {s_opt_err_alum:.3g}")

t_opt_alum = popt[2]
t_opt_err_alum = np.sqrt(pcov[2, 2])
print(f"t (best-fit) = {t_opt_alum:.4g} ± {t_opt_err_alum:.3g}")

# plot data
plt.errorbar(temperatures, alum_G, yerr=alum_uncert_G, fmt=".", color = 'black', label = 'data')

# creating x interval to include in y fit
x = np.linspace(1, 700, 100)
y_fit = G(x, *popt)
plt.plot(x, y_fit, color="red", label="best fit")


plt.grid(True)
plt.xlabel("Temperatures")
plt.ylabel("Modulus of Rigidity")
plt.title("Aluminium Rod\nBest FIt of Paramters")
plt.legend();
```

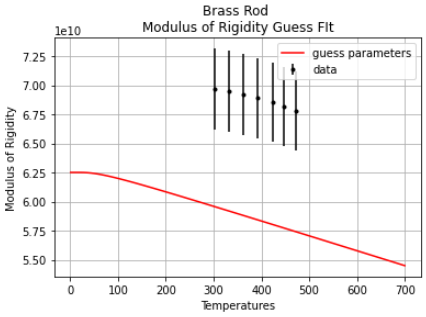
```
Go (best-fit) = 6.36e+10 ± 3e+08
s (best-fit) = 4.92e+10 ± 3.02e+10
t (best-fit) = 1405 ± 311
```



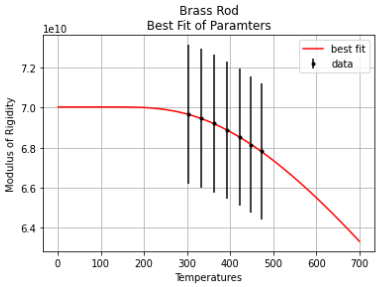
Fitting Brass G_0, s and t Parameters

In [88]:
```
def G(x, G0, s, t):
    return G0 - ( s / (np.exp(t/x)-1) )
```

In [89]:
```
# plot the raw data
plt.errorbar(temperatures, brs_G, yerr=brs_uncert_G, fmt=".", color = 'black', label = 'data')
plt.grid(True)
plt.xlabel('Temperatures')
plt.ylabel('Modulus of Rigidity')


# initial guess
pars0 = (6.25*10**10, 0.21*10**10, 163)

# overlay exponential

x = np.linspace(1, 700, 100)

plt.plot(x, G(x,*pars0), color = 'red', label = 'guess parameters')

plt.legend();
plt.title("Brass Rod \nModulus of Rigidity Guess FIt");
```

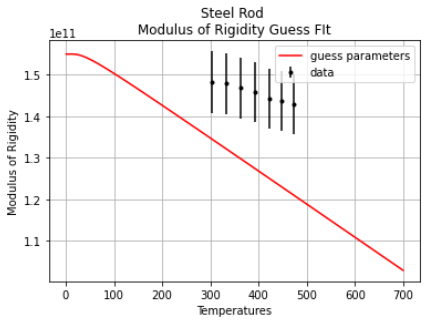


In [90]:
```
# perform fitting

popt, pcov = curve_fit(
    G, temperatures, brs_G, sigma = brs_uncert_G, absolute_sigma=True, p0=pars0
)

# extract each best-fit parameter and its error
G0_opt_brs = popt[0]
G0_opt_err_brs  = np.sqrt(pcov[0, 0])
print(f"Go (best-fit) = {G0_opt_brs:.3g} ± {G0_opt_err_brs:.1g}")

s_opt_brs = popt[1]
s_opt_err_brs = np.sqrt(pcov[1, 1])
print(f"s (best-fit) = {s_opt_brs:.3g} ± {s_opt_err_brs:.3g}")

t_opt_brs = popt[2]
t_opt_err_brs = np.sqrt(pcov[2, 2])
print(f"t (best-fit) = {t_opt_brs:.4g} ± {t_opt_err_brs:.3g}")

# plot data
plt.errorbar(temperatures, brs_G, yerr=brs_uncert_G, fmt=".", color = 'black', label = 'data')

# creating x interval to include in y fit
x = np.linspace(1, 700, 100)
y_fit = G(x, *popt)
plt.plot(x, y_fit, color="red", label="best fit")


plt.grid(True)
plt.xlabel("Temperatures")
plt.ylabel("Modulus of Rigidity")
plt.title("Brass Rod\nBest Fit of Paramters")
plt.legend();
```



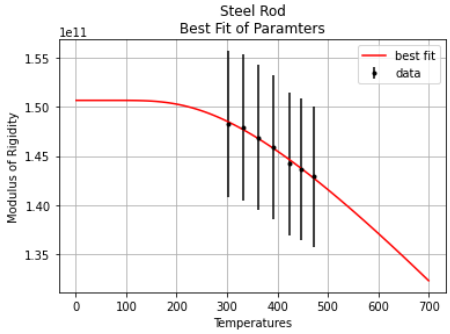
Fitting Steel G_0, s and t Parameters

In [91]:
```
# plot the raw data
plt.errorbar(temperatures, ste_G, yerr=ste_uncert_G, fmt=".", color = 'black', label = 'data')
plt.grid(True)
plt.xlabel('Temperatures')
plt.ylabel('Modulus of Rigidity')


# initial guess
pars0 = (1.55*10**11, 0.8*10**10, 100)

# overlay exponential

x = np.linspace(1, 700, 100)

plt.plot(x, G(x,*pars0), color = 'red', label = 'guess parameters')

plt.legend();
plt.title("Steel Rod \nModulus of Rigidity Guess FIt");
```



In [92]:
```
# perform fitting

popt, pcov = curve_fit(
    G, temperatures, ste_G, sigma = ste_uncert_G, absolute_sigma=True, p0=pars0
)

# extract each best-fit parameter and its error
G0_opt_ste = popt[0]
G0_opt_err_ste = np.sqrt(pcov[0, 0])
print(f"Go (best-fit) = {G0_opt_ste:.3g} ± {G0_opt_err_ste:.1g}")

s_opt_ste = popt[1]
s_opt_err_ste = np.sqrt(pcov[1, 1])
print(f"s (best-fit) = {s_opt_ste:.3g} ± {s_opt_err_ste:.3g}")

t_opt_ste = popt[2]
t_opt_err_ste = np.sqrt(pcov[2, 2])
print(f"t (best-fit) = {t_opt_ste:.4g} ± {t_opt_err_ste:.3g}")

# plot data
plt.errorbar(temperatures, ste_G, yerr=ste_uncert_G, fmt=".", color = 'black', label = 'data')

# creating x interval to include in y fit
x = np.linspace(1, 700, 100)
y_fit = G(x, *popt)
plt.plot(x, y_fit, color="red", label="best fit")


plt.grid(True)
plt.xlabel("Temperatures")
plt.ylabel("Modulus of Rigidity")
plt.title("Steel Rod\nBest Fit of Paramters")
plt.legend();
```

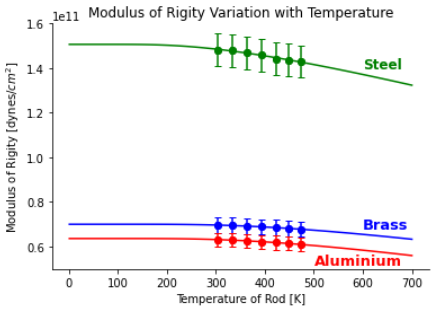
```
Go (best-fit) = 1.51e+11 ± 1e+09
s (best-fit) = 5.91e+10 ± 3.72e+10
t (best-fit) = 1009 ± 316
```



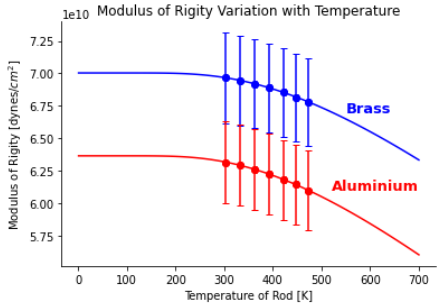
Theoretical Curves added to Modulus of Rigidity Plot

In [93]:
```
def G(x, G0, s, t):
    return G0 - ( s / (np.exp(t/x)-1) )
```

In [94]:
```
# Plotting the three modulus of rigidity's
plt.plot(temperatures, alum_G , 'ro')
plt.plot(temperatures, brs_G , 'bo')
plt.plot(temperatures, ste_G , 'go')

plt.xlabel("Temperature of Rod [K]")
plt.ylabel("Modulus of Rigity [dynes/$cm^2$]")
plt.title("Modulus of Rigity Variation with Temperature")
plt.grid(False)

# Plotting theoretical curves based on equation 12
plt.plot(x, G(x, G0_opt_alum, s_opt_alum, t_opt_alum), color = 'red')
plt.plot(x, G(x, G0_opt_ste, s_opt_ste, t_opt_ste), color = 'green')
plt.plot(x, G(x, G0_opt_brs, s_opt_brs, t_opt_brs), color = 'blue')

# Plots the uncertainty bars on each modulus of rigidity
plt.errorbar(temperatures, ste_G, ste_uncert_G, 1, capsize=3, fmt='o', color = 'green',  label = "Steel Measurements")
plt.errorbar(temperatures, brs_G, brs_uncert_G, 1, capsize=3, fmt='o', color = 'blue',  label = "Brass Measurements")
plt.errorbar(temperatures, alum_G, alum_uncert_G, 1,capsize=3, fmt='o', color = 'red', label = "Aluminium Measurements")

# Removes the top axis and right axis
ax = plt.subplot(111)
ax.spines["right"].set_visible(False)
ax.spines["top"].set_visible(False)

plt.text(500,0.52*10**11,'Aluminium', color='red', size= 13, weight = 'bold')
plt.text(600,0.68*10**11,'Brass', color='blue', size= 13, weight = 'bold')
plt.text(600,1.40*10**11,'Steel', color='green', size= 12, weight = 'bold')

plt.ylim(0.5*10**11,1.6*10**11)
```



In [95]:
```
# Plotting the three modulus of rigidity's
plt.plot(temperatures, alum_G , 'ro')
plt.plot(temperatures, brs_G , 'bo')

plt.xlabel("Temperature of Rod [K]")
plt.ylabel("Modulus of Rigity [dynes/$cm^2$]")
plt.title("Modulus of Rigity Variation with Temperature")
plt.grid(False)

# Plotting theoretical curves based on equation 12
plt.plot(x, G(x, G0_opt_alum, s_opt_alum, t_opt_alum), color = 'red')
plt.plot(x, G(x, G0_opt_brs, s_opt_brs, t_opt_brs), color = 'blue')

# Plots the uncertainty bars on each modulus of rigidity
plt.errorbar(temperatures, brs_G, brs_uncert_G, 1, capsize=3, fmt='o', color = 'blue',  label = "Brass Measurements")
plt.errorbar(temperatures, alum_G, alum_uncert_G, 1,capsize=3, fmt='o', color = 'red', label = "Aluminium Measurements")

# Removes the top axis and right axis
ax = plt.subplot(111)
ax.spines["right"].set_visible(False)
ax.spines["top"].set_visible(False)

plt.text(520,0.61*10**11,'Aluminium', color='red', size= 13, weight = 'bold')
plt.text(550,0.67*10**11,'Brass', color='blue', size= 13, weight = 'bold')
```



Determine the Internal Friction

Use the equation $\frac{\Delta w}{\sqrt{3}w_0}$ where Δw is bandwidth at half maxima of response curve and w_0 is the resonant frequency

In [96]:
```
def int_fric(w0, Δw):
    return (Δw/(np.sqrt(3)*w0))
# Defines function for internal frcition
```

In [97]:
```
# Measure of Internal Friction for each rod at each temperature

# Aluminium Readings
alum_IF_303 = int_fric(alum_w0_303, alum_Δw_303)
alum_IF_333 = int_fric(alum_w0_333, alum_Δw_333)
alum_IF_363 = int_fric(alum_w0_363, alum_Δw_363)
alum_IF_393 = int_fric(alum_w0_393, alum_Δw_393)
alum_IF_423 = int_fric(alum_w0_423, alum_Δw_423)
alum_IF_448 = int_fric(alum_w0_448, alum_Δw_448)
alum_IF_473 = int_fric(alum_w0_473, alum_Δw_473)

# Brass Readings
brs_IF_303 = int_fric(brs_w0_303, brs_Δw_303)
brs_IF_333 = int_fric(brs_w0_333, brs_Δw_333)
brs_IF_363 = int_fric(brs_w0_363, brs_Δw_363)
brs_IF_393 = int_fric(brs_w0_393, brs_Δw_393)
brs_IF_423 = int_fric(brs_w0_423, brs_Δw_423)
brs_IF_448 = int_fric(brs_w0_448, brs_Δw_448)
brs_IF_473 = int_fric(brs_w0_473, brs_Δw_473)

# Steel Readings
ste_IF_303 = int_fric(ste_w0_303, ste_Δw_303)
ste_IF_333 = int_fric(ste_w0_333, ste_Δw_333)
ste_IF_363 = int_fric(ste_w0_363, ste_Δw_363)
ste_IF_393 = int_fric(ste_w0_393, ste_Δw_393)
ste_IF_423 = int_fric(ste_w0_423, ste_Δw_423)
ste_IF_448 = int_fric(ste_w0_448, ste_Δw_448)
ste_IF_473 = int_fric(ste_w0_473, ste_Δw_473)
```

In [98]:
```
brs_IF_423
```

Out[98]: 0.0027651199905780626

In [99]:
```
# Puts the aluminium internal frictions into an array
alum_intfric = np.array([alum_IF_303, alum_IF_333, alum_IF_363, alum_IF_393, alum_IF_423, alum_IF_448, alum_IF_473])

# Puts the brass internal frictions into an array
brs_intfric = np.array([brs_IF_303, brs_IF_333, brs_IF_363, brs_IF_393, brs_IF_423, brs_IF_448, brs_IF_473])

# Puts the steel internal frictions into an array
ste_intfric = np.array([ste_IF_303, ste_IF_333, ste_IF_363, ste_IF_393, ste_IF_423, ste_IF_448, ste_IF_473])
```

Calculating the **Uncertainty on Internal Friction** for each value for using propogation of errors

In [100]:
```
def uncertainty_IF(w0, Δw0, w, Δw, IF):
    return IF * ( (Δw0/w0)**2 + (Δw/w)**2 )**(0.5)
```

In [101]:
```
# Puts each rods resonant frequency for each temperature into an array
alum_w0 = np.array([alum_w0_303, alum_w0_333, alum_w0_363, alum_w0_393, alum_w0_423, alum_w0_448, alum_w0_473])
brs_w0 = np.array([brs_w0_303, brs_w0_333, brs_w0_363, brs_w0_393, brs_w0_423, brs_w0_448, brs_w0_473])
ste_w0 = np.array([ste_w0_303, ste_w0_333, ste_w0_363, ste_w0_393, ste_w0_423, ste_w0_448, ste_w0_473])

# Puts each rods bandwidth frequency for each temperature into an array
alum_Δw = np.array([alum_Δw_303, alum_Δw_333, alum_Δw_363, alum_Δw_393, alum_Δw_423, alum_Δw_448, alum_Δw_473])
brs_Δw = np.array([brs_Δw_303, brs_Δw_333, brs_Δw_363, brs_Δw_393, brs_Δw_423, brs_Δw_448, brs_Δw_473])
ste_Δw = np.array([ste_Δw_303, ste_Δw_333, ste_Δw_363, ste_Δw_393, ste_Δw_423, ste_Δw_448, ste_Δw_473])

# Puts each rods internal friction for each temperature into an array
alum_IF = np.array([alum_IF_303, alum_IF_333, alum_IF_363, alum_IF_393, alum_IF_423, alum_IF_448, alum_IF_473])
brs_IF = np.array([brs_IF_303, brs_IF_333, brs_IF_363, brs_IF_393, brs_IF_423, brs_IF_448, brs_IF_473])
ste_IF = np.array([ste_IF_303, ste_IF_333, ste_IF_363, ste_IF_393, ste_IF_423, ste_IF_448, ste_IF_473])
```

In [102]:
```
# Calculates each specific uncertainty on each measurment for each rod
alum_uncert_IF = uncertainty_IF(alum_w0, 0.16, alum_Δw, 0.05, alum_IF )
brs_uncert_IF = uncertainty_IF(brs_w0, 0.16, brs_Δw, 0.05, brs_IF )
ste_uncert_IF = uncertainty_IF(ste_w0, 0.16, ste_Δw, 0.05, ste_IF )
```

In [103]:
```
alum_w0
```

Out[103]:
```
array([169.90499128, 169.63203568, 169.15440372, 168.64947827,
       168.07213775, 167.58249804, 166.96685441])
```

In [104]:
```
alum_Δw
```

Out[104]:
```
array([0.84969875, 0.82484387, 0.81833119, 0.81785219, 0.82396885,
       0.82446488, 0.825747  ])
```

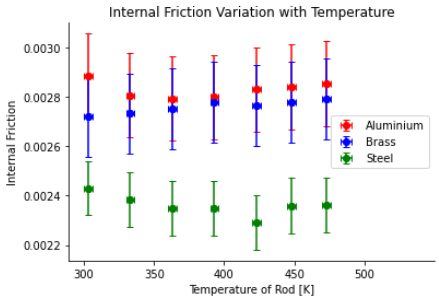
In [105]:
```
ste_IF
```

Out[105]:
```
array([0.00243002, 0.00238543, 0.00234875, 0.00235074, 0.00229213,
       0.00235955, 0.00236295])
```

In [106]:
```
ste_uncert_IF
```

Out[106]:
```
array([0.00010995, 0.00011008, 0.00011046, 0.00011083, 0.00011148,
       0.00011168, 0.00011198])
```

In [107]:
```
plt.plot(temperatures, alum_intfric, 'ro')
plt.plot(temperatures, brs_intfric, 'bo')
plt.plot(temperatures, ste_intfric, 'go')


plt.xlabel("Temperature of Rod [K]")
plt.ylabel("Internal Friction")
plt.title("Internal Friction Variation with Temperature")

# Plots the uncertainty bars on each modulus of rigidity
plt.errorbar(temperatures, alum_intfric, alum_uncert_IF, 1,capsize=3, fmt='o', color = 'red', label = "Aluminium")
plt.errorbar(temperatures, brs_intfric, brs_uncert_IF, 1, capsize=3, fmt='o', color = 'blue',  label = "Brass")
plt.errorbar(temperatures, ste_intfric, ste_uncert_IF, 1, capsize=3, fmt='o', color = 'green',  label = "Steel")

plt.xlim(289,549.9)

plt.legend(loc = 'center right')

ax = plt.subplot(111)
ax.spines["right"].set_visible(False)
ax.spines["top"].set_visible(False)
```



Calculation of Young's Modulus for each temperature

In [108]:
```
# Values of lengths
alum_l = 0.355
brs_l = 0.358
ste_l = 0.349

# Values of Densities and uncertainties
p_alum = 2710
p_brs = 8550
p_ste = 7880

p_uncert_alum = 60
p_uncert_brs = 150
p_uncert_ste = 150
```

In [109]:
```
def Young(p, w0, L):
    return p* (2**2*np.pi*w0*L)**2
# Uses equation 14 to find the value of Young's modulus
```

Uncertainty on Youngs Modulus

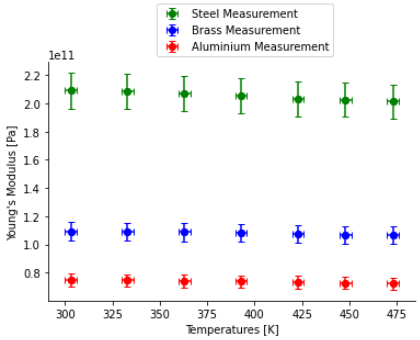
In [110]:
```
def Young_uncert(w0, Δw0, L, ΔL, p, Δp, Y):
    return Y * ( (2*(Δw0/w0))**2 + (2*(ΔL/L))**2 + (Δp/p)**2 )**(0.5)
# rearrangement of equation 20 to find the uncertainty on each Young's Modulus
```

In [111]:
```
Y_uncert_alum = Young_uncert(alum_w0, 0.16, alum_l, 0.01, p_alum , p_uncert_alum, Young(p_alum, alum_w0, alum_l))
```

In [112]:
```
Y_uncert_brs = Young_uncert(brs_w0, 0.16, brs_l, 0.01, p_brs , p_uncert_brs, Young(p_brs, brs_w0, brs_l))
```

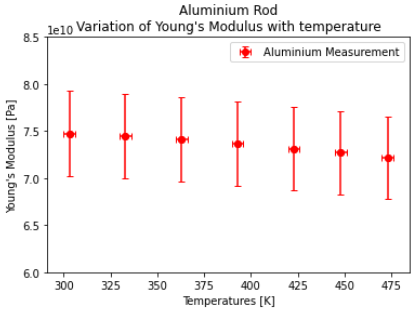
In [113]:
```
Y_uncert_ste = Young_uncert(ste_w0, 0.16, ste_l, 0.01, p_ste , p_uncert_ste, Young(p_ste, ste_w0, ste_l))
```

In [114]:
```
plt.plot(temperatures, Young(p_ste, ste_w0, ste_l), 'go')
plt.errorbar(temperatures, Young(p_ste, ste_w0, ste_l), Y_uncert_ste, 1,capsize=3,
             fmt='o', color = 'green', label = "Steel Measurement")

plt.plot(temperatures, Young(p_brs, brs_w0, brs_l), 'bo')
plt.errorbar(temperatures, Young(p_brs, brs_w0, brs_l), Y_uncert_brs, 1,capsize=3,
             fmt='o', color = 'blue', label = "Brass Measurement")

plt.plot(temperatures, Young(p_alum, alum_w0, alum_l), 'ro')
plt.errorbar(temperatures, Young(p_alum, alum_w0, alum_l), Y_uncert_alum, 1,capsize=3,
             fmt='o', color = 'red', label = "Aluminium Measurement")


plt.legend(bbox_to_anchor =(0.28, 1), ncol = 1)

plt.xlabel("Temperatures [K]")
plt.ylabel("Young's Modulus [Pa]")
# plt.title("\nVariation of Young's Modulus with temperature")

ax = plt.subplot(111)
ax.spines["right"].set_visible(False)
ax.spines["top"].set_visible(False)
```



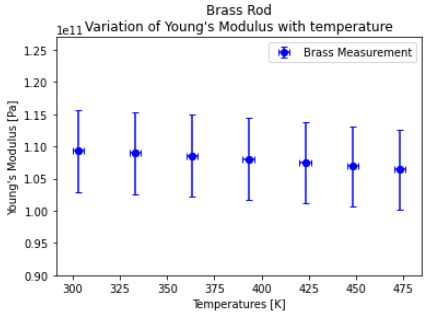
In [115]:
```
plt.plot(temperatures, Young(p_alum, alum_w0, alum_l), 'ro')
plt.errorbar(temperatures, Young(p_alum, alum_w0, alum_l)=, Y_uncert_alum, 1,capsize=3,
             fmt='o', color = 'red', label = "Aluminium Measurement")

plt.legend()

plt.xlabel("Temperatures [K]")
plt.ylabel("Young's Modulus [Pa]")
plt.title("Aluminium Rod\nVariation of Young's Modulus with temperature")

plt.ylim(6*10**10, 8.5*10**10)
```

Out[115]: (60000000000.0, 85000000000.0)



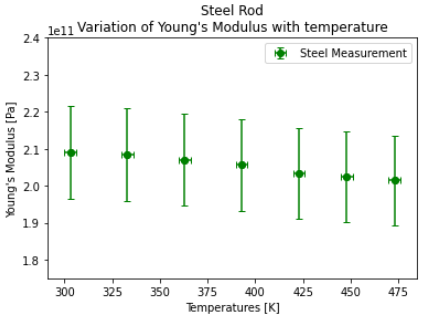
In [116]:
```
plt.plot(temperatures, Young(p_brs, brs_w0, brs_l), 'bo')
plt.errorbar(temperatures, Young(p_brs, brs_w0, brs_l), Y_uncert_brs, 1,capsize=3,
             fmt='o', color = 'blue', label = "Brass Measurement")

plt.legend()

plt.xlabel("Temperatures [K]")
plt.ylabel("Young's Modulus [Pa]")
plt.title("Brass Rod\nVariation of Young's Modulus with temperature")

plt.ylim(0.9*10**11, 1.27*10**11)
```

Out[116]: (90000000000.0, 127000000000.0)



In [117]:
```
plt.plot(temperatures, Young(p_ste, ste_w0, ste_l), 'go')
plt.errorbar(temperatures, Young(p_ste, ste_w0, ste_l), Y_uncert_ste, 1,capsize=3,
             fmt='o', color = 'green', label = "Steel Measurement")


plt.legend()

plt.xlabel("Temperatures [K]")
plt.ylabel("Young's Modulus [Pa]")
plt.title("Steel Rod\nVariation of Young's Modulus with temperature")

plt.ylim(1.75*10**11, 2.4*10**11)
```

Out[117]: (175000000000.0, 240000000000.0)



Young's Modulus at room temperature for each rod

In [118]:
```
# Aluminium Rod
print(f"Young's Modulus for Aluminium at 303K = {Young(p_alum, alum_w0, alum_l)[0]*48:.2g} ± {Y_uncert_alum[0]*48:.1g}")
```

```
Young's Modulus for Aluminium at 303K = 7.5e+10 ± 5e+09
```

In [119]:
```
# Brass Rod
print(f"Young's Modulus for Brass at 303K = {Young(p_brs, brs_w0, brs_l)[0]*20:.2g} ± {Y_uncert_brs[0]*20:.1g}")
```

```
Young's Modulus for Brass at 303K = 1.1e+11 ± 6e+09
```

In [120]:
```
# Steel Rod
print(f"Young's Modulus for Steel at 303K = {Young(p_ste, ste_w0, ste_l)[0]*20:.2g} ± {Y_uncert_ste[0]*20:.1g}")
```

```
Young's Modulus for Steel at 303K = 2.1e+11 ± 1e+10
```