Determination of Residual Stress by Neutron Diffraction Analysis

Felix Gifford
c3260374

June 6, 2017

1 Abstract

In many engineering situations it is important to control and measure residual stresses accurately and non-destructively. Residual stresses can be measured in many ways, however most of these techniques involve a destructive method. Some examples include cutting into a sample of material to observe the effects of the stresses 'relaxing', drilling holes into the material and observing the stress' effects on a core of material, and many similar methods. It is not always ideal or even possible to use a destructive technique such as the above, and one must begin to consider non-destructive methods involving magnets, sound, or in this particular case, neutron diffraction.

2 Background

2.1 Diffraction

Diffraction consists of a range of phenomena that occur when a wave interacts with physical objects. When a wave approaches the edge of an object it is bent around it into the area that geometrically should be in shadow. Once a wave diffracts around an object, various interference effects can occur. These interference effects are the result of multiple wave interacting with each other out of phase, and can have both constructive and destructive effects. Constructive effects occur when the peaks or troughs of the waves align and add together, increasing the amplitude of the wave. Destructive effects, on the other hand occur when a peak of one wave intersects with the trough of another, causing the waves to cancel out. These interference effects occur with the diffraction of waves but are most apparent when the size of the opening is close to the wavelength. Diffraction from multiple apertures also leads to a variety of interference patterns. A prominent example of this is Young's double-slit interferometer, in which Young passed a beam of monochromatic beam of light through two adjacent slits and observed an interference pattern, proving the wave nature of light.

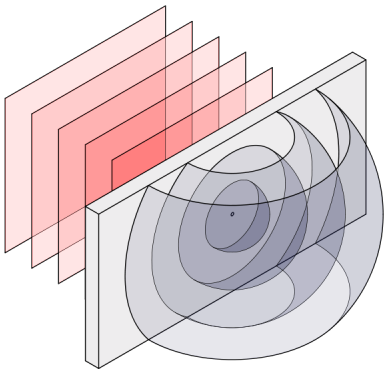


Figure 1: A visualization of a wavefront diffracting through a pinhole much smaller than it's wavelength.

(Inductiveload, 2007)

2.2 Quantum Diffraction

One of the fundamental concepts in quantum mechanics is that of wave-particle duality. This concept first originated with Einstein's idea that like, an electromagnetic wave, could also be propagated by discrete quantum particles, photons. From this de Broglie theorised the converse, if light has properties of both a wave and a particle, then all particles have properties of waves. The de Broglie wavelength of a particle was derived from rearranging the formulae for the Planck-Einstein relation,

$$E = hf$$

and momentum,

$$p = \frac{E}{c} = \frac{h}{\lambda}$$

where f and λ represent the frequency and wavelength, c the speed of light, h Planck's constant, into

$$\lambda = \frac{h}{p}$$

which relates momentum to the wavelength through Planck's constant. This relation holds true for particles, and this was experimentally proven using electrons. The Davisson-Germer experiment in 1927 involved firing electrons at a nickel target and measuring the intensity at the target. The intensity varied as the predicted diffraction pattern suggested, thus confirming de Broglie's hypothesis. Just as light diffracts when it interacts with physical objects, so can quantum particles. A crystal lattice structure exhibits characteristic diffraction patterns when it interacts with waves, and de Broglie's ideas allow one to extend this idea to quantum particles as well. Huygen's principle means that each atom that interacts with a particle, becomes the source of it's own wave and this effect compounding with a fixed wavelength leads to diffraction patterns

that can be used to infer details of the crystal lattice structure. The diffraction peaks of these patterns occur at certain angles relative to the planes in the crystal lattice and this relationship expresses itself as Bragg's Law:

$$n\lambda = 2d\sin\theta$$

. In the case of crystal diffractometry, d represents the spacing between lattice planes.(Wensrich & Jackson, n.d.)(Fultz & Howe, 2013)

2.3 Residual Stress

When a material experiences a loading and subsequent unloading, there are some stresses left in the material. These stresses are known as residual stresses. Residual stresses are commonly left behind after various manufacturing techniques, sometimes they are a desired result, and sometimes they are a detrimental by-product. They can improve the strength and performance of materials, or weaken a material leading to undesired failure. A very common example of residual stresses being utilized to improve the performance of a material is tempered or toughened glass. Tempered glass is annealed glass (all residual stresses are relaxed prior to the tempering process) which undergoes either a chemical or thermal toughening procedure (Ford, n.d.). The thermal toughening procedure involves taking the annealed glass and heating it well above its transition temperature, approximately 564 °C, before rapidly quenching. The glass is not completely cooled as the desired result is a solidified outer layer and an inner volume that is still above the transition temperature. Once the glass has fully cooled, the outer layers which experienced the quench are under a great deal of residual compressive stress and the inner volume, which was allowed to slowly cool, experiences a residual tensile stress. These residual stresses greatly improve the strength of the glass by allowing the outer surfaces to experience a much higher tensile load before failure.

2.4 Strain Measurement

A strain diffractometer is a device that uses the principles of quantum mechanics and diffraction to provide insight into the behaviour of of material at the nano-scale. A material behaving elastically under an applied stress will have a corresponding strain in accordance with Hooke's Law. This strain manifests itself physically as an elongation but on the nano-scale it can be measured as a change in the atomic spacing of the material. This change in atomic spacing can be observed by comparing the positions of a diffraction peak, caused by the lattice structure in the material, before and after a stress is applied. This measurement technique is both non-destructive and non-contact, making it ideal for use in many engineering situations. The output of the strain diffractometer will be two angles, those of the diffraction peak before, and while, the strain is applied. These angles can be converted into spacings using Bragg's Law, and further converted into strain (ϵ) using the following formula where d is the control spacing and d' is the strained spacing.

$$\epsilon = \frac{d' - d}{d}$$

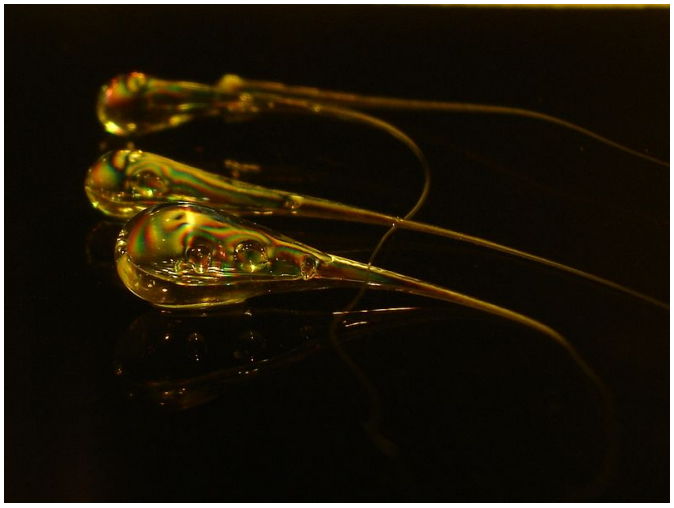


Figure 2: Prince Rupert's drops are an extreme example of residual stresses in toughened glass

(Grogan, 2006)

The region of material that is tested is known as the "Gauge Volume", and the measured strain represents the average value over this region. The gauge volume is formed at the intersection of the incoming beam and the diffracted beam. The direction that the strain is measured in is also the direction that bisects the two beams, thus is is important to take two (or three) perpendicular measurements in order to get a complete strain state. It is important to not that this measurement technique will not measure any plastic strain in the sample. Elastic strain is caused by the change in lattice spacings due to applied stress, whereas plastic strain is the result of the movement of defects and displacements in the lattice of the material. Plastic strain does not affect the lattice spacing, only the location of atoms in the lattice.

3 Experimental Details

3.1 Preparation of Samples

The samples to undergo residual stress testing were cut from sheets of aluminium from a supplier based in Sydney. The stress was applied using a four point bending device and displacement was measured with a steel ruler. The four point bending device applied a pure bending moment load on the central 100 mm region of the sample. The material outside this region was removed and discarded. The remaining material was analysed for residual stresses in the strain diffractometer. The four point

3.2 Testing of Mechanical Properties

In order to determine the residual stress in a material by the neutron diffraction method, it is critical to know some of the important mechanical properties of the material. Namely: the Young's Modulus (E), and Poisson's Ratio (ν). The Young's Modulus (or Modulus of Elasticity) is the relationship between stress (σ) and strain (ϵ) in accordance with Hooke's Law.

$$\sigma = E\epsilon$$

Poisson's Ratio is used for stress and strain in three dimensions:

The mechanical properties of the materials were determined by the use of the University's tensile testing facilities on small samples of each material. The samples used were small 'dog-bone' coupons, cut from the original stock plates of aluminium. A schematic of these 'dog-bones' is shown below.

These coupons were put through a standard tensile test, involving the application of an increasing tensile load until failure while simultaneously recording measurements with a load cell and a strain gauge. Once the sample had passed passed it's yield stress and begun plastic deformation the strain gauge was removed the and load was increased to the point of failure. The final load (F_F), and ultimate strain (σ_U) are recorded in Table 1.

Table 1: Final load and ultimate strain measured for the tensile test coupons

Sample	F_F (MPa)	σ_U
6061-T6	18.1	0.2
7075-T6	34.7	0.18

3.3 Strain Measurement

The strain diffractometry measurements were carried out at the KOWARI diffractometer at Sydney's Bragg Institute. A neutron wavelength of 1.73 Å was used to obtain 90° measurement geometry and a gauge volume of $2\,\text{mm} \times 2\,\text{mm} \times 20\,\text{mm}$ was used. Reference spacings (d_0) were obtained from an un-deformed sample of each material. (Wensrich & Jackson, n.d.)

Table 2: Reference (d_0) spacings

Sample	d_0 (Å)
6061-T6	$1.2181161 \pm 8 \times 10^{-6}$
7075-T6	$1.2188861 \pm 5 \times 10^{-6}$

3.4 Data Processing

3.4.1 Software

All data was processed using Python3, as part of the Anaconda Data Science distribution (Analytics, n.d.). The data was taken in as a Numpy array, which is a numerical array computing library included in the Anaconda distribution, before being processed. The plots were created in Matplotlib, a plotting library that mirrors MATLAB's plotting functionality. Uncertainties were handled by the python Uncertainties library (Lebigot, n.d.). All the libraries used are open source, and the data processing code is included in the appendices.

3.4.2 Determination of Elastic Properties

Once the raw data had been gathered from the experiments it needed to be processed and used in calculations to determine the residual stress. The first data-set to be processed was the stress/strain data as that would allow for the determination of the elastic properties of the two metals, allowing further calculations. The data, which consisted of two columns, gauge length and applied force, was loaded from text files to a Numpy array, then sorted by displacement values before being transformed into stress and strain values. In order to find an elastic modulus (E) from this sorted data, a linear regression fit was carried out on the first 20% of the data. This 20% was chosen as it contained a linear region of the elastic behaviour of the metal, while being minimally affected by the non-linearities caused by plastic deformation. The results of this fit are recorded in Table 3 and Figure 3 show plots of the stress/strain curve and the calculated elastic modulus.

When the elastic modulus was determined, the 0.2% proof stress was determined by

Table 3: Linear Regression results for the stress/strain datasets

Sample	E (GPa)	R value
6061-T6	71.73	0.9995
7075-T6	71.69	0.9997

finding the intersection of the line

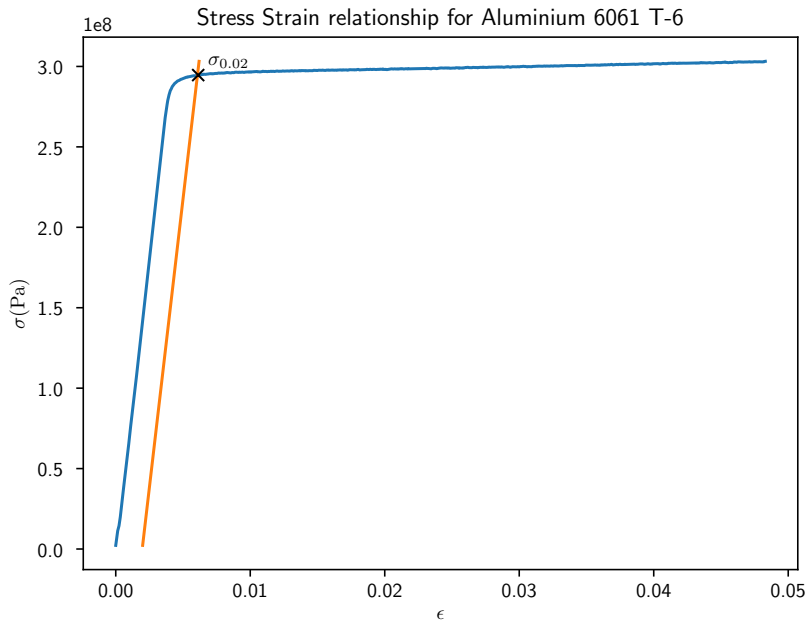
$$\sigma = E(\epsilon - 0.002)$$

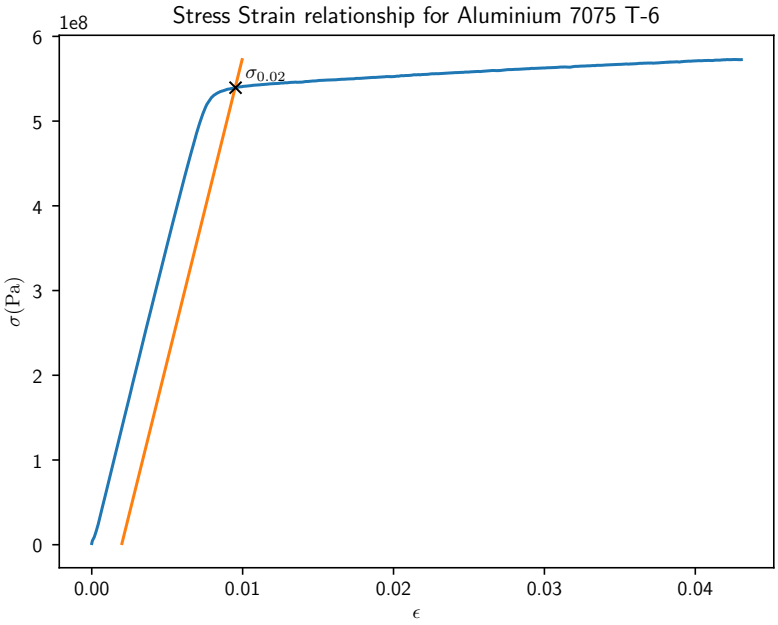
with the curve given by the stress-strain relationship. Table 4 contains the results of the data analysis.

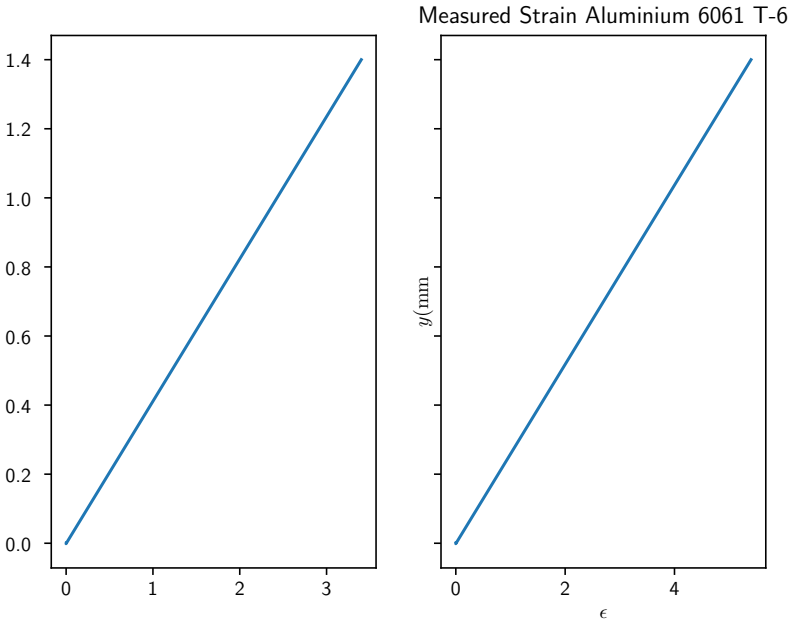
3.4.3 Strain Diffractometry Data

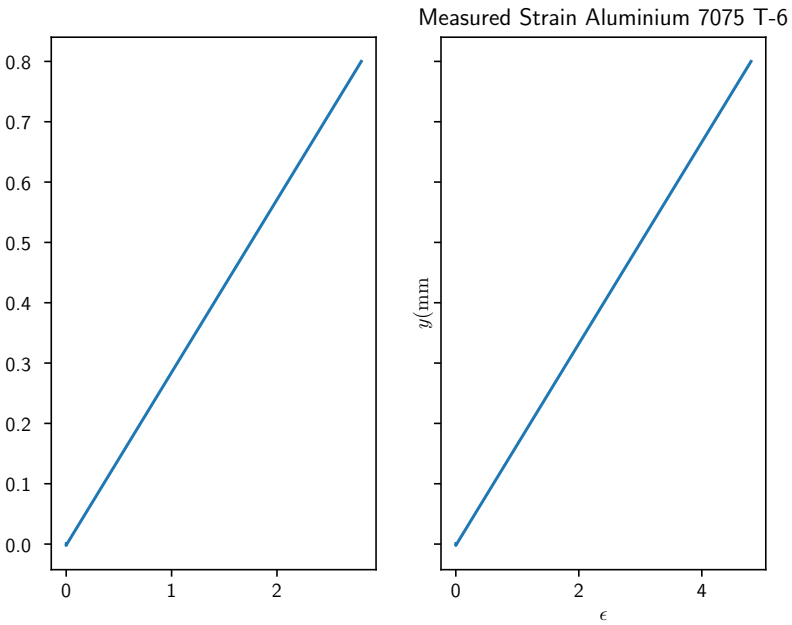
The data from the diffractometry measurements was also stored in a plain-text format. This data consisted of 5 columns: y-position along the sample, axial lattice spacing (d_a), axial error, transverse lattice spacing (d_t), and transverse error. All data was measured in ngstroms (Å), except the y-position which was measured in mm. The processing of

Figure 3: Stress/Strain curves for 6061-T6 and 7075-T6 Aluminium samples






Measured Strain Aluminium 6061 T-6
1.4
1.2
1.0
0.8
0.6
0.4
0.2
0.0
0
1
2
3
y(mm
0
2
4
ϵ


Measured Strain Aluminium 7075 T-6
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0.0
0
1
2
y(mm
0
2
4
ϵ

this data was simple as it involved one operation performed on the columns of the data array.

3.5 Results

The results of the data analysis are included in tabular form below. Table 4 contains the elastic properties that were obtained through the analysis, and Table 5 contains the axial and transverse strain values.

Table 4: Elastic property results from data analysis

Sample	Elastic Modulus (E) (GPa)	Proof Stress ($\sigma_{0.2}$) (MPa)
6061-T6	71.73	294.63
7075-T6	71.69	539.42

Table 5: Strain diffractometry results

6061 T6 y-value	Axial Strain	Transverse Strain	7075 T6 y-value	Axial Strain	Transverse Strain
1.4	-0.001420291546	0.0001698179672	0.8	-0.002306120317	0.0007393406159
3.4	-0.001092727532	0.0003629292807	2.8	-0.001656815185	0.0005740552788
5.4	-0.000789461694	0.0001668108647	4.8	-0.000759490160	0.0002885765946
7.4	-0.000135464919	8.2350114246127	6.8	0.0001730842611	-3.921531306335
9.4	0.0003284613018	-0.000188470540	8.8	0.0011415340613	-0.000412717808
11.4	0.0006615075525	-0.000215692904	10.8	0.0020506813557	-0.000749782937
13.4	0.0013152941661	-0.000717828128	12.8	0.0019620701228	-0.000683342766
15.4	0.0022257435067	-0.000498756235	14.8	0.0012867871739	-0.000412089365
17.4	0.0027324809186	-0.000483877522	16.8	0.0004229410771	-0.000189509914
19.4	0.0023129084329	-0.000706408034	18.8	-0.000533231940	3.8428529129992
21.4	0.0011895343965	-0.000523492793	20.8	-0.001173392657	0.0003211883374
23.4	-0.000492923457	0.0002548246427	22.8	-0.001761837303	0.0006461054892
25.4	-0.002551468616	0.0009106537545	24.8	-0.002344967261	0.0009160191423
27.4	-0.002203688958	0.0005845362359	26.8	-0.002115738295	0.0010095159834
29.4	-0.002301479308	0.0008785935921	28.8	-0.001206846152	0.0007523369082
31.4	-0.001292880867	0.0006289408702	30.8	-0.000181070241	0.0004585990438
33.4	-0.000624315695	0.0007182451656	32.8	0.0006884720401	0.0001592896990
35.4	-0.000351230067	-9.740040378756	34.8	0.0015098014490	-0.000230179013
37.4	0.0001733127080	-0.000270417573	36.8	0.0022143332342	-0.000597834366
39.4	0.0008594295732	-0.000301208562			
41.4	0.0012029912419	-0.000498725039			

3.6 Theoretical Results

The behaviour of a theoretical elasto-plastic beam under pure bending load gives rise to a stress distribution with regions of plastic stress on the outer surfaces of the beam, normal to the radius of curvature, as they experience the highest stress. The interior of the beam, surrounding the neutral plane remains under purely elastic load. Calculations based on elasto-plastic beam theory are found in the appendices.

4 Conclusion

Due to time constraints full theoretical models were not completed and a comparison could not fully be made. If full theoretical models were made based on elasto-plastic beam theory a comparison between the distribution of elastic stresses in reality and theory could be made. There would definitely be some variance between the two due to the assumptions made in determining the elastic properties

5 Discussion

The analysis of residual stress using neutron diffraction is clearly an effective method (Fultz & Howe, 2013) (*NATO Advanced Research Workshop on Measurement of Residual and Applied Stress Using Neutron Diffraction*, n.d.), however this could not be fully verified in this report. It is critical to create an effective (and complete) theoretical model and understanding to hold as a comparison to any experimental technique. The methods of elasto-plastic beam theory are sufficient to develop such a model (Wensrich, 2017), but more time is needed to gain a complete understanding. The analysis of the given data is sufficient, but further data about the samples is needed to process the data to a higher degree of accuracy. If test data to determine the Poisson ratio was sampled, then it could be a more precise value than the $\nu = 0.34$ (Wensrich & Jackson, n.d.) was assumed to be. The data analysis code is sufficiently modular to be able to process more data and can easily be extended. Visualizing the data does require additional effort, using 3D models to display the distribution of stress and strain within a sample would allow for a much more intuitive representation of the state of the samples.

References

Analytics, C. (n.d.). *Anaconda software distribution.* `continuum.io`.

Ford, M. (n.d.). *How is tempered glass made?* [blog]. `www.scientificamerican.com/article/how-is-tempered-glass-mad/`.

Fultz, B., & Howe, J. (2013). *Diffraction and the x-ray powder diffractometer.* Springer Berlin Heidelberg.

Grogan, M. (2006). *Prince rupert's drops.* `commons.wikimedia.org/wiki/File:Prince`$_{R}uperts_{d}rops.jpg$.

Inductiveload. (2007). *Diffraction through pinhole.* `commons.wikimedia.org/wiki/File:Diffraction`$_{t}hrough_{P}inhole.svg$.

Lebigot, E. O. (n.d.). *Uncertainties: a python package for calculations with uncertainties.* `pythonhosted.org/uncertainties/`.

Nato advanced research workshop on measurement of residual and applied stress using neutron diffraction. (n.d.). Springer-Verlag.

Wensrich, C. (2017). *Mech2420 lectures.*

Wensrich, C., & Jackson, R. (n.d.). Mech2420: Residual stress measurement using neutron diffraction report handout.

A Analysis Code

```
#!/Users/felixgifford/anaconda/bin/python3
print("Importing Modules")
import sys, os
print("System libraries")
import configparser
print("Config file Parser")
import uncertainties
print("Uncertainties")
from uncertainties import unumpy
print("Uncertainties unumpy")
import numpy as np
print("Numpy")
import scipy.stats, scipy.interpolate, scipy.optimize
print("Scipy")
import matplotlib as mpl

mpl.use('pgf')
params = {
        "pgf.preamble": [
        r'\usepackage{siunitx}'
        ]

}
mpl.rcParams.update(params)
print("Matplotlib")
import matplotlib.pyplot as plt
print("Pyplot")



class Sample(object):
        def __init__(self, header):
                self.sample = configparser.ConfigParser()

                self.sample.read(header)

                self.name = self.sample.get('Sample', 'name')
                self.id = self.sample.get('Sample', 'id')

                self.dimensions = (self.sample.getfloat('Values', 'width'), s
                self.area = self.dimensions[0]*self.dimensions[1]
```

```
        self.poisson = self.sample.getfloat('Values', 'poisson')

        self.d_0 = uncertainties.ufloat(self.sample.getfloat('Values'
        print(self.d_0)

        self.ss_path = self.sample.get('Data', 'ss_data')
        self.diff_path = self.sample.get('Data', 'diff_data')

    def createElastoPlastic_Model(self):
        self.ss_data = np.loadtxt(self.ss_path, skiprows=1)

        self.ss_data = self.ss_data[self.ss_data[:,0].argsort()]
        #convert the force column into stress
        #force is given in kN so e3 converts that to N
        self.ss_data *=  [1,(1e3)/self.area]
        #convert the length column into strain
        self.ss_data[:,0] = ((self.ss_data[:,0]-self.ss_data[0,0])/se

        self.ElastoPlastic_Model = ElastoPlastic_Model(self.ss_data)

    def create_SS_plot(self):
        self.ElastoPlastic_Model.create_plots()

        plt.title("Stress Strain relationship for {}".format(self.nam

        plt.savefig('pgf/{}_StressStrain.pgf'.format(self.id))

        plt.close()

    def createDiffractionModel(self):
        self.diff_data = np.loadtxt(self.diff_path, skiprows=1)

        self.DiffractionModel = DiffractionModel(self.diff_data, self

        self.DiffractionModel.dump_to_disk(self.id)

    def create_diff_plot(self):
        self.DiffractionModel.create_plots()

        plt.title("Measured Strain {}".format(self.name))
        plt.ylabel(r"$y (\si{\milli\meter}$")
        plt.xlabel(r"$\epsilon$")
```

```
                plt.savefig('pgf/{}_strain.pgf'.format(self.id))

                plt.close()



class ElastoPlastic_Model(object):
        '''Contains all the data and functions to generate an ElastoPlastic m
        def __init__(self, ss_data):
                '''Initialize'''
                self.stress_strain_data = ss_data

                self.data_n = len(self.stress_strain_data)
                self.percent_x = lambda x: int(self.data_n*(x/100))

                self.create_model()

        def create_model(self):
                '''Creates the model functions from data'''
                self.stress = scipy.interpolate.interp1d(self.stress_strain_da

                self.elastic_modulus = scipy.stats.linregress(self.stress_stra

                stress_range = (self.stress_strain_data[0,1], self.stress_stra
                strain_range = (0.002, (stress_range[1]-stress_range[0])/self

                self.E_02 = scipy.interpolate.interp1d(strain_range, stress_ra

                x1 = max(strain_range[0], self.stress_strain_data[0,0])
                x2 = min(strain_range[1], self.stress_strain_data[-1,0])

                diff = lambda x: self.stress(x)-self.E_02(x)

                self.proof_strain = scipy.optimize.bisect(diff, x1, x2, xtol=
                self.proof_stress = self.stress(self.proof_strain)

                #self.ultimate_stress = self.stress(self.percent_x(100))

        def create_plots(self):

                x=np.linspace(self.stress_strain_data[0,0], self.stress_strain
                stress_range = (self.stress_strain_data[0,1], self.stress_stra
```

```
                x1=np.linspace(0.002, (stress_range[1]-stress_range[0])/self.
                plt.plot(x, self.stress(x))
                plt.plot(x1, self.E_02(x1))
                plt.plot(self.proof_strain, self.proof_stress, 'kx')
                plt.annotate(r'$\sigma_{0.02}$', xy=(self.proof_strain, self.

                plt.xlabel(r'$\epsilon$')
                plt.ylabel(r'$\sigma (\si{\pascal})$')

        def __str__(self):
                return "E: {:>5.2f} GPa, Proof Stress: {:>5.2f} MPa".format(s

class DiffractionModel(object):
        def __init__(self, diff_data, d_0, E):
                '''Initialize'''
                self.diff_data = diff_data
                self.d_0 = d_0
                self.data_n = len(self.diff_data)
                self.E = E

                self.transformData()

        def transformData(self):
                self.axial = uncertainties.unumpy.uarray(self.diff_data[:,1],
                self.transverse = uncertainties.unumpy.uarray(self.diff_data[

                self.strain_measured = np.column_stack( (self.diff_data[:,0],

                print(self.E)

                self.stress_measured = self.strain_measured*self.E

        def create_plots(self):

                fig, (ax0, ax1) = plt.subplots(ncols=2, sharey=True)
                ax0.plot(uncertainties.unumpy.nominal_values(self.strain_measu
                                        uncertainties.unumpy.nominal_values(
                                        )#xerr=uncertainties.unumpy.std_devs
                ax1.plot(uncertainties.unumpy.nominal_values(self.strain_measu
                                        uncertainties.unumpy.nominal_values(
                                        )#xerr=uncertainties.unumpy.std_devs

        def dump_to_disk(self, id):
```

```
                np.savetxt("data/{}_strain.txt".format(id), self.strain_measur
                np.savetxt("data/{}_stress.txt".format(id), self.strain_measur
        #def __str__(self):
        #       return self.strain_measured

for sample in ["data/6061.cfg", "data/7075.cfg"]:
        s = Sample(sample)
        s.createElastoPlastic_Model()
        s.createDiffractionModel()
        print(s.ElastoPlastic_Model)
        s.create_diff_plot()
```

B Sample Headers

B.1 6061

```
[Sample]
name = Aluminium 6061 T-6
id = 6061

[Values]
width = 12.5e-3
height = 5e-3
poisson = 0.34

d_zero = 1.2181161
d_zero_error = 8e-6

[Data]
ss_data = data/stress_strain_6061.txt
diff_data = data/diffraction_data_6061.txt
```

B.2 7075

```
[Sample]
name = Aluminium 7075 T-6
id = 7075

[Values]
width = 12.5e-3
height = 5e-3
poisson = 0.34

d_zero = 1.2188861
d_zero_error = 5e-6
```

```
[Data]
ss_data = data/stress_strain_7075.txt
diff_data = data/diffraction_data_7075.txt
```