3-2021

Error Reduction for the Determination of Transverse Moduli of Single-Strand Carbon Fibers via Atomic Force Microscopy

Joshua D. Frey

Error Reduction for the Determination of Transverse Moduli of Single-Strand Carbon Fibers via Atomic Force Microscopy

THESIS

Joshua D. Frey, Major, USA

AFIT-ENP-MS-21-M-115

DEPARTMENT OF THE AIR FORCE
AIR UNIVERSITY

AIR FORCE INSTITUTE OF TECHNOLOGY

Wright-Patterson Air Force Base, Ohio

AFIT-ENP-MS-21-M-115

ERROR REDUCTION FOR THE DETERMINATION OF TRANSVERSE MODULI OF SINGLE-STRAND CARBON FIBERS VIA ATOMIC FORCE MICROSCOPY

THESIS

Presented to the Faculty

Department of Engineering Physics

Graduate School of Engineering and Management

Air Force Institute of Technology

Air University

Air Education and Training Command

in Partial Fulfillment of the Requirements for the

Degree of Master of Science in Nuclear Engineering and Master of Science in Materials Sciences

Joshua D. Frey, M.S, B.A.

Major, USA

March 2021

AFIT-ENP-MS-21-M-115

ERROR REDUCTION FOR THE DETERMINATION OF TRANSVERSE MODULI OF SINGLE-STRAND CARBON FIBERS VIA ATOMIC FORCE MICROSCOPY

THESIS

Joshua D. Frey, M.S, B.A.
Major, USA

Committee Membership:

Abigail Bickley, Ph.D.
Chair

Major Nicholas Herr, Ph.D.
Member

Darren Holland, Ph.D.
Member

Abstract

PeakForce Atomic Force Microscopy (AFM) Quantitative Nanomechanical Measurement (QNM) is utilized to measure the transverse fiber modulus of single strand carbon fibers to less than 5% error for eleven types of carbon fibers, manufactured by Mitsubishi, Toray, and HEXCEL, with longitudinal moduli between 924-231 GPA, including export-controlled fibers. A positive linear correlation between the longitudinal and transverse modulus with an R^2=0.76 is found. Statistical and physical criterion for outlier removal are studied and established to improve the quality of data to exclude outlier measurement points in an image based on the peak force, adhesion force, and indentation depth. Statistical and physical criterion are also developed to exclude outlier images within the sample set.

Three alternative methods for calculating the transverse modulus using the raw instrument data were studied. The first method approximated the indentation force curve using the peak force and adhesion force values. This method calculated moduli lower than that reported by the instrument and with no correlation between the transverse and longitudinal modulus. The second method approximated the indentation force curve using the peak force and net force zero point. This method found values larger than that reported by the instrument and no correlation between the transverse and longitudinal modulus. The final method performs a linear fit to the measured indentation force curves at each indentation point. This method also found values lower than reported by the instrument.

Pitch-based fibers are found to exhibit lower measurement error than PAN-based fibers. Additionally, PAN fibers exhibited no apparent modulus correlation when the Pitch fibers are excluded. Underlying reasons for this lack of correlation are

explored, with the most likely reasons being the difference in long-range order in the fiber microstructure and aging effects due to the different sourcing and storage methods used for the PAN fibers. Low uncertainty characterization of the transverse modulus supports greater understanding of fiber mechanical behavior, and would allow fiber manufacturers to certify their fibers in both the longitudinal and transverse axes. Additionally, it would improve the confidence in engineering estimates used by industry and defense programs for transverse performance of carbon fiber-reinforced composites.

To Private First Class Frank Frey.

Once a Marine, always a Marine.

Acknowledgements

The author would like to acknowledge the assistance and advice of the members of his committee, specifically Major Nicholas Herr for his assistance in understanding the operations and theory of the Atomic Force Microscope. This research would certainly have not been completed, much less contributed new knowledge and understanding, with the support of Dr. Bickley, Maj Herr, and Dr. Holland. The author also appreciates the support of the other AFIT nuclear engineering students, especially his fellow Army officers, whose support and insights into both coursework and their own approaches to research shed new light on solving some of the problems presented in this document. Finally, the author cannot sufficiently express in words his appreciation for the support of his wife, who will soon be free of the burdens of helping with homework.

Joshua D. Frey

Table of Contents

Page

Page

List of Figures

List of Tables

ERROR REDUCTION FOR THE DETERMINATION OF TRANSVERSE MODULI OF SINGLE-STRAND CARBON FIBERS VIA ATOMIC FORCE MICROSCOPY

I. Introduction

1.1 Background

Characterizing material behavior under various types of load is an essential step in determining the suitability of that material for a specific use, whether it be within industry or defense applications. Design requirements for future generations of industrial machinery and high performance vehicles will almost certainly demand materials that are simultaneously lighter and stronger, with greater resilience under thermal or other environmental conditions, while maintaining low cost relative to other materials with similar properties.

One of these novel materials, developed between 1960-1980 in Japan, the United States, and Great Britain, is carbon fiber. Carbon fiber was developed to improve the mechanical reinforcement in composite materials. Composite materials are engineered to take advantage of the combination of multiple different material classes, such as polymers, metal alloys, and ceramics. In the case of carbon fiber, it is utilized as a reinforcement material within a resin matrix, allowing the production of a material that possesses both low weight and high specific strength and specific modulus.

Carbon fibers are produced from one of three precursors: Rayon, Polyacrylonitrile (PAN), or Pitch. Each of these precursors have slightly different production methods and microstructures leading to a respective increase in the Young modulus. The

improvement of production processes for these fibers has also led to steady increases in the Young modulus of carbon fibers over time, as shown in Figure 1. Much of this improvement has been due to research into the microstructure of the fibers and its correlation to mechanical properties [1]. Since the mid-1980s, manufacturers have continued the development of higher modulus fiber, with Pitch-based fibers having Young moduli as high as 935 GPa [2]. Additionally, manufacturers have pursued efforts to reduce unit cost and improve production efficiency in order to expand the market for carbon fibers, to the point where modern fibers are used in products from bicycle frames to ballistic missiles.

The Young modulus shown in Figure 1 and referenced in the majority of manufacturer specifications is the longitudinal fiber modulus, the stiffness parallel to the axis of the fiber. The figure compares the properties of the WYB and VYB Rayon fibers, shown by the solid lines, developed in the 1960s with the properties PAN and Pitch-based fibers, shown by the dotted lines, developed since the 1970s. This behavior is relatively simple to characterize by traditional methods of applying an increasing load on the fiber until deformation and fracture. The transverse modulus, that is the stiffness perpendicular to the fiber axis, is more difficult to characterize due to the very small fiber diameter, the convolution of fiber and composite load sharing, and the large relative errors associated with measurement in previous studies. Additionally, different measurement methods have shown significantly different results for the transverse modulus, leading to uncertainty in whether a given method is in fact accurate at all [3, 4, 5]. These factors motivate this research and demonstrate that there is a continued need for more consistent methods to determine fiber transverse modulus behavior.

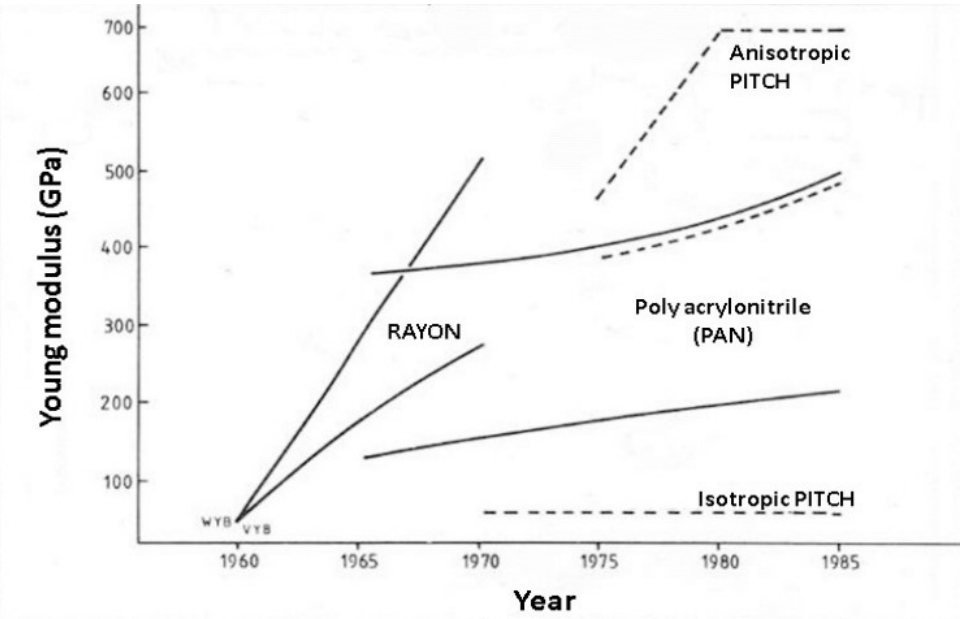

Figure 1. Evolution of the Young modulus for different carbon fibers during the 1960-1985 period [1]. Reprinted with permission from John Wiley & Sons Inc.

1.2 Motivation

As the industrial development of carbon fiber has increased its utility in many industries, and proven to be an efficient and cost-effective way of achieving a combination of strength, flexibility, and low cost, it has begun to compete with other high performance materials for defense and nuclear applications. Specifically, its strength and heat resistance make it an excellent material for use in uranium enrichment centrifuges. While low velocity centrifuges, with low separative work capacity, can be fabricated from more common metallic alloys, the highest performance centrifuges require very high rotational velocities, necessitating the use of more exotic materials like carbon fiber.

In order to prevent the use of high-performance centrifuges by states seeking to develop nuclear weapons, the International Atomic Energy Agency has added carbon fibers with specific material properties to its export control regimes. These

export controlled limits are defined for "carbon 'fibrous or filamentary materials', having all of the following: 1. A 'specific modulus' exceeding 12.7 x 10^6 m; and 2. A 'specific tensile strength' exceeding 23.5 x 10^4 m" [6]. While the centrifuges themselves are export controlled, it is also possible to fabricate them locally, and so materials such as carbon fibers must also be controlled. To verify that these export controls have not failed it is necessary to check compliance via regular inspection, which include forensics samples that could contain a variety of useful materials, such as uranium particles or carbon fiber fragments, which would arise from normal wear on the centrifuges in a facility or due to accidental release. A method of measuring these small fiber fragments to compare against the export control limits is necessary, but conventional methods of mechanical testing are limited and likely are infeasible for such small fiber fragments.

In order to address the need to support forensic verification of carbon fiber properties, an alternative method of testing must be developed. This study seeks to continue the development of the method initiated by Veigas, in which an Atomic Force Microscope is used to perform nanoindentation measurements of carbon fiber fragments [3].

1.3 Problem Statement

This research addresses the problem of how to reduce the error associated with the PeakForce Quantitative Nanomechanical Measurement (PF-QNM) procedure for determining the carbon fiber transverse modulus to demonstrate this method for the full range of commercially available carbon fibers.

1.4 Research Objectives

The objective of this work is to develop and demonstrate a method for reducing the measurement error associated with a previously developed PF-QNM AFM method for determining the transverse Young modulus for a variety of single-strand carbon fibers fragments [3]. This work will be accomplished through three research objectives that expands the applicability of previous work to a broader range of fibers.

1.4.1 Decreasing Uncertainty via Statistical Methods

A data post-processing method will be developed to allow for the direct analysis of instrument raw data and manipulation using statistical approaches to reduced measurement error associated with the currently developed technique. This post-processing method will be demonstrated and compared against the study by Veigas in which the PF-QNM method of transverse modulus measurement was developed [3]. The method will also be used to produce further modulus characterization for high and very high modulus fibers.

1.4.2 Measure Fiber Transverse Modulus

The transverse modulus of a variety of carbon fiber samples will be determined via PF-QNM AFM measurement. These samples will be taken from those used for previous method development in order to compare results and validate the post-processing procedure, as well as from a broad range of other carbon fibers to test the application of the method to high and very high modulus carbon fibers that are export controlled.

1.4.3 Determine the Influence of Fiber Microstructure on Fiber Transverse Modulus

PAN and Pitch-based carbon fiber transverse moduli will be measured and compared to the theoretical prediction of longitudinal-transverse modulus behavior. If the Pitch-based fibers deviate from the predicted behavior, a review of the literature on fiber microstructure will be presented in order to explain this behavior phenomenologically.

II. Theory

2.1 Overview

This chapter provides an overview of the relevant background material to understand the state of the fields of science and engineering necessary for this study. It describes the theory of operation and engineering applications of AFM and the PF-QNM method. Additionally, this chapter addresses the various sources of error in AFM measurement and the approaches used in the field for limiting measurement error. It also develops the mathematical models and simplifying assumption of surface contact mechanics on which the PF-QNM method relies. Finally, the details of carbon fiber production and fiber microstructure are described in order to differentiate the Pitch and PAN fiber types and demonstrate the underlying physical phenomena that lead to their different mechanical properties.

2.2 Atomic Force Microscopy and PeakForce Quantitative Nanomechanical Measurement

2.2.1 Theory of Instrument Operation

Atomic Force Microscopy is a relatively young instrumental technique, having been invented in 1985 to overcome the primary limitation of Scanning Tunnelling Microscopy, which could only image a conductive sample. The mechanism for imaging is the interaction between the sample and the AFM tip via near-field forces, such as interatomic short-range forces, Van der Waals dipole interactions, electrostatic forces, and capillary forces [7]. The first three of these forces serve as the means of interaction by either attraction or repulsion between the tip and sample as a function of distance, while for dry samples capillary forces are a source of error in the true

tip-sample interaction, and are minimized by operation in a controlled atmosphere glovebox.

The combination of these forces produces a deflection of the AFM cantilever, which must be balanced by the cantilever spring force, as shown in Figure 2. For a known cantilever spring constant k_N and a measured z-axis deflection, the tip-sample interaction force can be measured precisely. Detection of the cantilever deflection and production of an image is done via laser reflection from the cantilever onto a photodetector, as shown in Figure 3. The laser is reflected onto a four quadrant photo detector, allowing for determination of z-axis deflection.

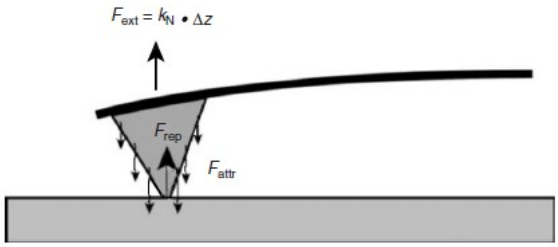

Figure 2. Tip-Sample-Cantilever Force Balance [8]. Reprinted with permission from Wiley-VCH Verlag GmbH & Co. The external force on the tip F_{ext} is the product of the cantilever spring constant k_N and the change in cantilever height Δz, while the internal forces (to the tip-sample interaction) are those of repulsion and attraction, F_{rep} & F_{attr}.

PeakForce Quantitative Nanomechanical Measurement is a proprietary instrumental method developed by the Bruker Corporation as an additional imaging mode for their suite of AFM instruments. It utilizes the PeakForce Tapping method of imaging, which utilizes a piezoelectric material to oscillate a cantilever in the z-axis at its resonant frequency and "tap" the sample surface with a very small tip mounted on the cantilever as it is translated across the x and y axes. The maximum tip-sample interaction force, or "PeakForce", is used as a feedback loop parameter to maintain the oscillation and properly track over any surface roughness features or material

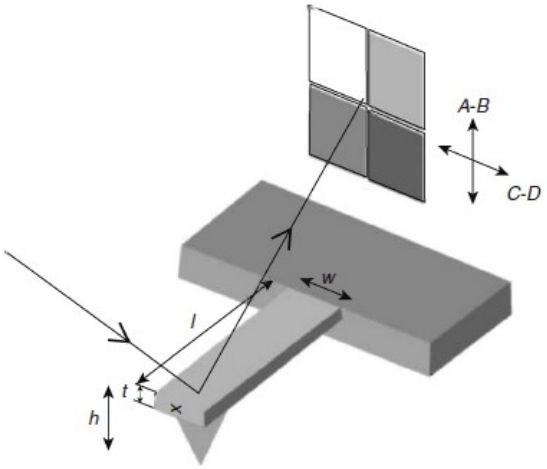

Figure 3. Laser Deflection by AFM Cantilever onto Photodetector [8]. Reprinted with permission from Wiley-VCH Verlag GmbH & Co. The variables t, l, & w are the dimensions of the cantilever, h is the height from the tip vertex to the top of the cantilever and A, B, C, & D define the quadrants of the photodetector.

changes, which would result in a deviation from the expected constant PeakForce value.

Figure 4 shows the typical behavior of the force interaction between the tip and sample as a function of time at one image pixel. At each imaging point the tip must be brought into contact with the surface, which is the behavior from 0-200μs. There is then a small "jump-to-contact" force that occurs when the tip is close enough to the surface for attractive near-field forces to dominate. Once the tip is in contact with the surface, beginning at $\sim$225μs, the applied tip force results in a large increase to the peak force value at $\sim$500nN. After reaching this value, the tip is retracted. The negative force peak at $\sim$350μs is due to adhesion forces between the tip and sample. Finally, the tip is drawn sufficiently far from the surface that the interaction force returns to zero.

A more useful method for displaying the tip-sample interaction is in terms of the force as a function of separation distance from the sample, as shown in Figure 5. This

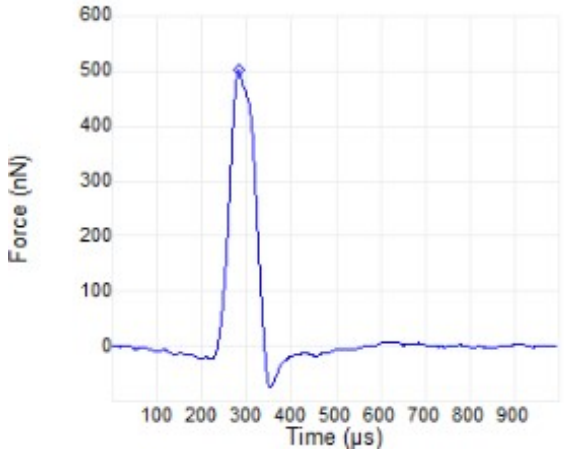

Figure 4. Tip-Sample Force vs. Time [9]. The left of the peak force is the downward movement to engage the sample, and the right of the peak force is the retraction from the sample. The peak force is much larger than the adhesion force. Reprinted with permission from Bruker Corporation.

method describes the interaction in a manner similar to the mechanical stress-strain curve, where the applied stress and resulting length change strain for a sample is plotted. For the force curve, the retract curve in red is used for the measurement of the sample modulus, with the slope of the curve between the absolute maximum and minimum force being the fit region. The use of the force curve for modulus measurement is described in more detail in later sections.

2.2.2 Analysis of AFM Force Curves

The force-distance curves produced by the interaction between the tip and sample is the fundamental measurement produced by the AFM in the PF-QNM method, and so it is insightful to understand the actual behavior of the force curves underlying the AFM image. Figures 6 and 7 show this behavior for a 128x128 image consisting of 16,384 indentations, with each vertical line along the Height Sensor axis showing the range of tip force, while the force curve line is plotted along the average force at each value of tip height. At 200nm, the tip has achieved the PeakForce setpoint and

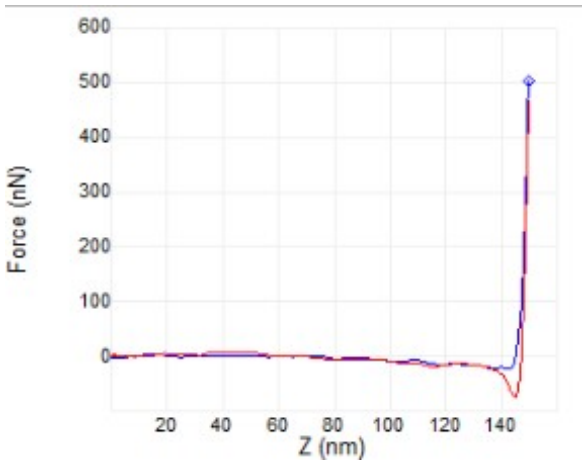

Figure 5. Separation Distance vs. Tip-Sample Force [9]. This curve is essentially Figure 4 folded across the peak force value and plotted against z-axis height. The blue curve is the engage curve, while the red curve is the retract curve. Reprinted with permission from Bruker Corporation.

begins retracting. From 400-0nN, the force curve is characterized by generally linear behavior, and is analogous to the elastic region of a stress-strain curve. The linear region also shows discontinuous behavior around the peak force maximum, which is excluded from the force curve for calculations of the transverse modulus. As the tip force becomes negative, indicating the tip has begun retracting from the sample, the force curve is characterized by a near-field adhesion force, with the minimum force value measured as the adhesion force. Finally, the tip fully separates from the sample and retracts, where it is no longer of interest.

It can be clearly seen that the region of tip-sample interaction is characterized by large relative variation in the net tip force. The region of largest absolute variation is at the maximum force, in addition to showing discontinuous behavior as the tip retracts from the maximum force. Additionally, the z-axis distance where the adhesion force minimum occurs, and the width of the region, is also seen to vary from indentation to indentation, though this cannot be demonstrated on Figure 7. This variation in adhesion may be due to a number of factors, including sample topography

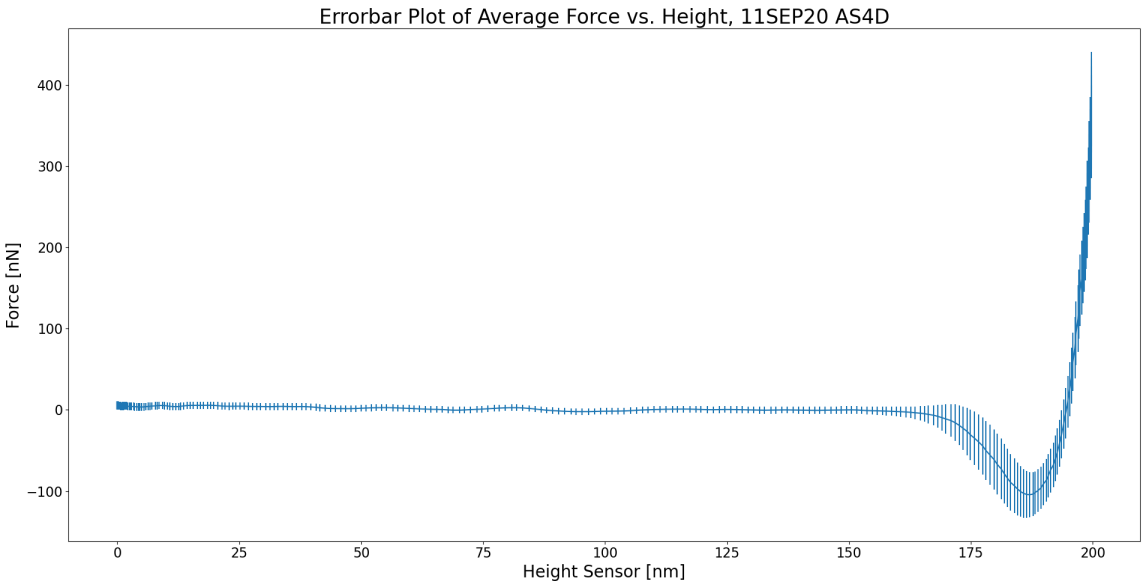

Figure 6. Average Force Curve.

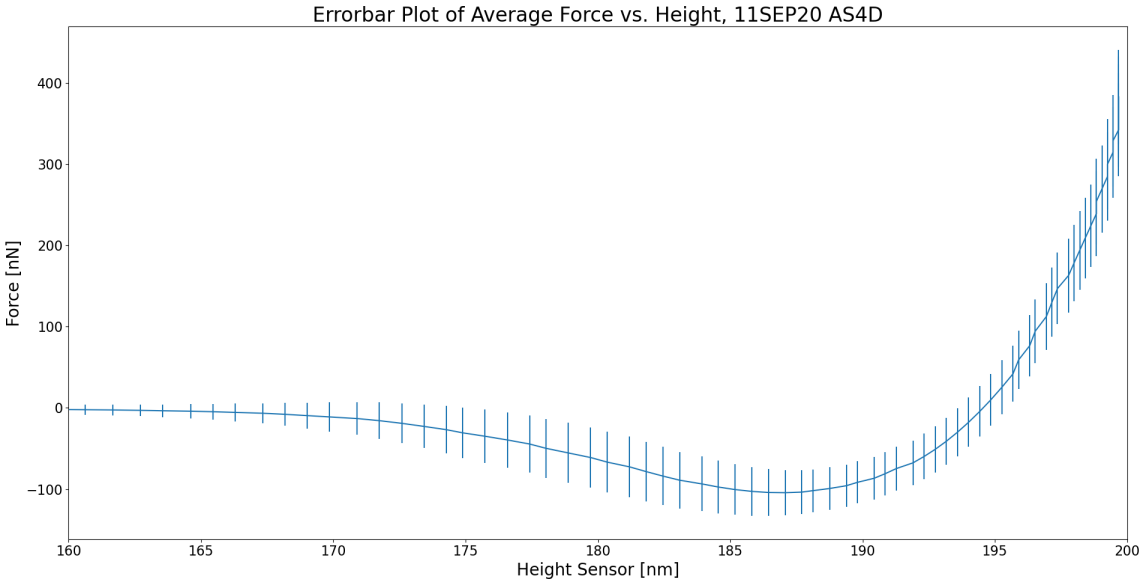

Figure 7. Average Force Curve, Final 40nm.

and deviations from the indentation direction from the ideal surface normal.

For the application of the PeakForce Tapping method to the determination of mechanical properties, the relation between applied force and separation distance, shown in Figures 6 & 7, becomes useful since it contains information analogous to a stress-strain curve used in macroscopic materials analysis. In the case of PF-QNM, the slope of the force-distance curve is combined with the geometric properties of the AFM tip to determine the sample modulus using an analytical contact mechanics model, which will be described later. By performing this analysis at all points of a PF-QNM image, the modulus can be used as a means of generating image contrast for differentiating materials. This method can also be used to determine the average modulus of a sample of a known material by performing many nanoindentation measurements. An advantage of PF-QNM is that it allows for the measurement of microscopic samples and the performance of many more indentations due to the small indentor size.

The range of force values at each point in the AFM image, however, indicates that the method of fitting the slope of the force curve is sensitive to the variations in the tip force across the measured surface. Since the peak force, minimum force, and indentation depth will all vary at every point, the material modulus determined from each force curve will vary. Decreasing the variation associated with these different portions of the force curve is critical to minimizing the measurement error of the PF-QNM method.

2.3 Sources of Error in Atomic Force Microscopy

As in any instrumental technique, AFM measurements are influenced by sources of error that can be intrinsic to the specific instrument, the AFM technique itself, or deviations induced by the operator or environment. Marinello, et. al. proposed a

taxonomy for the classification of error sources in AFM measurement and a model for how these sources can be characterized and measured. The four sources of error in AFM measurement are the scanning system, the tip-surface interaction, the environment, and the data processing technique, each of which can present multiple different distortions of the true sample image [10].

Distortions due to the scanning system are caused by variation in the photodiode signal produced by lateral motion of the piezoelectric scanner. The three modes which can cause these distortions are scaling of the scanned topography, piezoelectric hysteresis, and scanner cross-talk. Topography scaling error arises from a poorly calibrated scaling coefficient to correct for the difference between the real tip position and the measured tip position. The linear model of the correction coefficient is shown below, where x' is the measured coordinate and x is the correct coordinate.

$$c_{xx'} = \frac{\partial x}{\partial x'} \tag{1}$$

While non-linear scaling error is possible for some AFMs, the Bruker Dimension Icon AFM used in this study does not present this contribution due to its closed feedback loop. Closed feedback loop AFMs have been characterized as presenting position distortions <1% relative to the measured range in position for the entire image. Piezoelectric hysteresis is due to different material properties and dimensions of the materials used to in the scanner system. This variation is described as the sensitivity, which is the ratio between the piezo movement and piezo voltage. The scanner exhibits more sensitivity at the extreme range of the scanner, and so voltage instabilities will result in larger movement deviations. This sensitivity can be minimized by accounting for instrument warm-up time, which allows the piezoelectric material to condition to the applied voltage. Piezoelectric creep also causes deviation due to drift when an offset voltage is applied, such as for a large change in position. This can be

minimized by accounting for where the creep effect will be greatest in the image and removing that section of the image during post-analysis. Finally, scanner cross-talk occurs due to correlation between coordinate motion in 3D, which results in drift in one axis when the scanner moves in another. It is of greatest concern in open feedback loop AFMs, but can be present in closed loop systems, and presents most significantly when imaging curved surfaces, so that minimizing this source of error is limited both by the instrument and the topography of the sample being measured [10].

Tip-surface interaction distortions are caused by overshoot, mode switching, and convolution. Edge overshoots arise from improper hysteresis and creep compensation in the z-axis piezoelectric, and can be reduced through variation of the feedback loop gain. Mode switching arises in intermittent mode (tapping) due to jumps between repulsive and attractive behavior. Convolution arises from the convolution of the tip geometry with the surface topography to produce an image that combines both. This becomes more significant as the tip dimension approaches the measured feature dimension, and can vary over time due to tip degradation and contamination [10]. This convolution can be minimized by ensuring a sufficiently sharp tip relative to the surface roughness and periodic recharacterization of the tip shape to prevent overuse.

Environmental errors arise from complex instrument-operator-room interactions, exemplified by drift and noise. Drift is the gradual uncontrolled movement of the system over time and can be recognized by a varying misalignment producing shifts in the z and x directions. It is most prevalent during room temperature changes. Drift velocities are usually between 30-100 nm/hr (or $\sim$1 nm/min), with 10 nm/hr being the effective minimum limit. Drift is of concern in closed-loop scanners, since the tip position is determined based on a reference frame. Initial drift generally decays over time, and so a "warm-up" period is employed. Noise effects are caused by many interactions, including mechanical, vibrational, acoustical, and electrical.

Some can be minimized via insulation and dampening, but generally are unavoidable. Filtering of noise can be performed, but also leads to loss of information from the measurement [10].

Data processing distortions arise from the digital image processing performed by the instrument, and can result from the filtering and leveling methods due to their application to all data points in an image. Filtering is performed in three ways. Statistical filtering is used for noise removal via averaging, median filtering, or conservative reduction. Fourier filtering is used for frequency dependent noise. Outlier filtering allows for elimination of artifacts and unwanted features due to anomalies such as dust particles, but can be misleading since the true image data is replaced by a software generated value. Data leveling is performed by instrument software to remove the instrument slope from image data. Routines compute the best fit plane for the measured data and subtract it from each data point. This produces two errors. Cosine error results in an angle dependent shrinking of the total topography (1-cosβ). This distortion is not constant over the image plane, but is proportional to both the x-position and the local height. Surface topography deformation results when each scan line is adjusted by a given parameter with respect to adjacent lines (in the fast scan direction), leading to a flattening of the image in the slow scan direction. This is due to the applied flattening not being based on the real measurement value [10].

All of these sources of measurement deviation can be controlled or adjusted to some degree, however simultaneous characterization and control is challenging, requiring a complete characterization of the instrument with a known standard. For the PF-QNM method used in this study, it is important to understand the scale on which scanning system and tip-surface interaction errors present themselves in the z-axis, since an inaccurate z-axis measurement results in both an inaccurate indentation depth and an inaccurate tip radius. Tip-surface convolution is an important

source of error in tip shape characterization, which will be described later. For the study of carbon fibers, which have both a macroscopic curvature and microscale surface roughness, the influence of tip-surface interaction errors from mode switching and of scanner deviations due to axis cross-talk may also contribute to uncertainty in a measurement. Statistical measurement error will also arise from the hundreds of individual nanoindentation measurements performed during the PF-QNM method, and so an appropriate distribution must be used to determine the standard deviation for these measurements. Finally, this study proposes to develop a method of image data processing that is performed outside of the instrumental analysis software, and so should avoid errors arising from filtering and leveling methods applied by that software, which are not clearly described in the instrument software guide. Other error sources are expected to arise during this alternative data processing, however, and will be appropriately characterized.

2.4 Contact Mechanics Modeling

While the interaction between the AFM tip and the sample prior to contact can be modeled by the short-range force interactions described earlier, the contact forces after the tip begins indenting the sample surface must be determined using the models of contact mechanics developed for macroscopic objects. In the case of the Bruker PF-QNM method, the two models utilized are the Derjaguin, Muller, and Toporov (DMT) model and the Sneddon model. The Sneddon model is based on a conical indentor with no adhesion force, and is not applicable for this study [11]. The DMT model, on the other hand, is an extension of the Hertz model of contact between two hard spheres, which was the earliest analytical solution for contact forces [12]. The DMT model extends the Hertz model by accounting for adhesion forces, though this is limited to frictionless adhesion. The DMT model has also been extended to

many other indentor shapes, including the paraboloid indentor that is assumed in the Bruker DMT model [13].

2.4.1 Contact Mechanics for Nanomechanical Measurements

The physical mechanism used by the PF-QNM method is effectively the same as that used for macro-scale mechanical measurements and microindentor instrumental methods, in which a compressive force is applied over some surface area of the sample and the deformation distance is measured, as shown in Figure 8. From the applied values and the measured deformation, the compressive modulus can be determined as

$$E = \frac{\sigma}{\epsilon} = \frac{F/A_0}{\Delta L/L_0} = \frac{FL_0}{\Delta L A_0} \tag{2}$$

where σ is the mechanical stress, ϵ is the mechanical strain, F is the force applied by the indentor, ΔL is the indentation distance, L_0 and A_0 are the sample thickness and indented surface area, respectively.

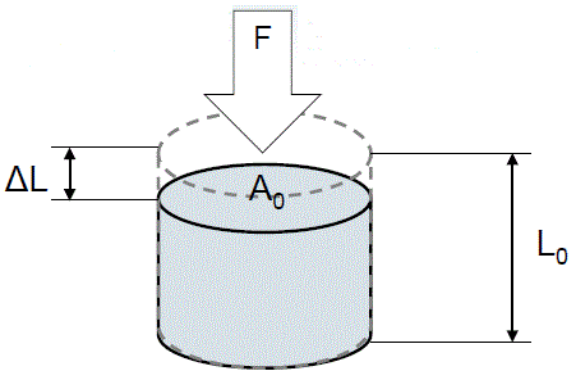

Figure 8. Diagram of Stress-Strain Deformation Mechanics [3].

This simple model is sufficient for macroscopic samples, such as those used in mechanical testing of engineering alloys, which can be shaped to precise thicknesses,

and for indentors with precisely known indentation area. The PF-QNM method of nanoindentation using the AFM, however, is not able to apply this model due to the geometry of the indentor tip and the imprecision in the sample thickness. The DMT Model of surface contact mechanics, introduced earlier, provides the mechanism for determining the mechanical interaction between the AFM tip and the sample.

The DMT model of contact mechanics was originally developed in 1975 as a model for determining the surface interaction mechanics between an elastic spherical particle and a rigid surface. It extends the Hertz model developed in 1881 by accounting for adhesion between the particle and the surface [12]. For the two interacting objects, there are two balanced force components, the repulsive and attractive forces. The DMT model gives the attractive force as

$$F_e = \frac{4R^{1/2}E}{3(1-\nu^2)}\alpha^{3/2} = \frac{4E^*}{3}\sqrt{R\alpha^3} \tag{3}$$

where F_e is the elastic repulsive force applied by the surface to the sphere, R is the radius of the sphere, E is the elastic modulus of the sphere, ν is the Poisson's ratio of the sphere, E^* is the sample reduced modulus, and α is the indentation depth. The repulsive force is given by

$$F_g = \pi R\phi(\epsilon) \tag{4}$$

where F_g is the molecular attraction force applied by the surface to the sphere, and $\phi(\epsilon)$ is a material dependent repulsive interaction potential that varies with α, but at large α becomes constant [12]. Since the repulsive/attractive labeling of these forces is merely a convention for conceptually balancing forces, the labels could be reversed if the surface is assumed to be elastic and the tip rigid, as shown in Figure 9, without affecting the mathematical particulars of the derivation.

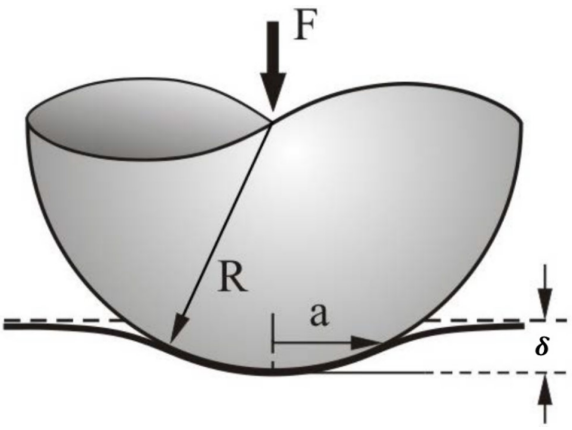

Figure 9. Diagram of Rigid Paraboloid Tip Indenting an Elastic Surface [9]. F is the applied tip force, R is the radius of curvature of the tip, δ is the indentation depth, a is the tip radius at δ.

The assumption underlying the early development of the DMT model required an elastic spherical indentor and a rigid surface, such that the forces described above could be used to determine the interaction surface area and the pressure incident on the rigid surface. The DMT model has been extended over time to allow for the use of other indentor shapes, including any indentor shape with a power law profile of the form,

$$f(r) = cr^n \tag{5}$$

where r is the indentor tip radius, c is an arbitrary constant and n is any positive real number. This study utilizes the power law indentor profile formulation to determine a numerical approximation of the AFM tip radius at any indentation depth.

As described earlier, the PF-QNM method applied using the AFM instrument uses a peak applied tip force value as a constant feedback loop parameter, while measuring the z-axis tip movement as the tip indents the sample. The DMT model allows for the calculation of the sample surface elastic modulus if the AFM tip is

assumed to be rigid relative to the sample. This necessitates correct tip selection for a sufficiently hard tip relative to the sample so that this approximation can hold. Finally, the PF-QNM method is also capable of measuring the adhesion force as the tip withdraws from the surface, which avoids the difficulty of determining an accurate analytical form of the repulsive interaction potential between the tip and sample.

2.4.2 Error in the DMT Modulus Calculation

The DMT model described by Equation 3 can be solved for the Young's modulus to give the following:

$$E = \frac{3(F_{peak} - F_{adh})(1 - \nu^2)}{4\sqrt{R\alpha^3}} \tag{6}$$

where F_{peak} is the maximum applied tip force and F_{adh} is the adhesion force. These force values define the point on the AFM indentation force curve between which the Young's modulus is the slope. The Peak Force AFM is capable of imaging using measurements of the peak tip force, adhesion force, and indentation depth, each of which will have their own variation about some mean value. In the case of the peak force, the mean value will be the peak force setpoint, and the instrument will attempt to maintain a constant peak force through a feedback loop. The indentation depth and the adhesion force will themselves vary based on the feedback loop. Additionally, the tip radius is a function of indentation depth. By characterizing the variation in each of these secondary image measurements, and using error propagation on the DMT modulus equation, a theoretical minimum variance bound can be derived.

$$\frac{\partial E}{\partial F_{peak}} = \frac{3(1 - \nu^2)}{4\sqrt{R\alpha^3}} \qquad \frac{\partial E}{\partial F_{adh}} = -\frac{3(1 - \nu^2)}{4\sqrt{R\alpha^3}}$$

$$\frac{\partial E}{\partial \nu} = \frac{3(\nu - 1)(F_{adh} - F_{peak})}{2\sqrt{R\alpha^3}}$$

$$\frac{\partial E}{\partial R} = -\frac{3(1-\nu^2)(F_{adh} - F_{peak})\left(\alpha^3 + 3R\alpha^2 \frac{d\alpha}{dR}\right)}{8(R\alpha^3)^{3/2}} \quad \frac{\partial E}{\partial \alpha} = -\frac{3(1-\nu^2)(F_{adh} - F_{peak})\left(3R\alpha^2 + \alpha^3 \frac{dR}{d\alpha}\right)}{8(R\alpha^3)^{3/2}}$$

$$\sigma_{DMT}^2 = \left(\frac{\partial E}{\partial F_{peak}}\sigma_{F_{peak}}\right)^2 + \left(\frac{\partial E}{\partial F_{adh}}\sigma_{F_{adh}}\right)^2 + \left(\frac{\partial E}{\partial R}\sigma_R\right)^2 + \left(\frac{\partial E}{\partial \alpha}\sigma_\alpha\right)^2 + \left(\frac{\partial E}{\partial \nu}\sigma_\nu\right)^2 \tag{7}$$

The theoretical minimum variance, Equation 7, for any DMT modulus measurement can be compared to the actual variance to test the measurement precision against the absolute limit based on the tip-sample interaction. Two important terms in this theoretical error are the uncertainty in the Poisson ratio and the interdependence of α and R. While the Poisson ratio is generally assumed to be a fixed value, this value is either determined via experiments that are inherently uncertain to some degree, or is merely asserted to be a certain value. Both of these assumption involve uncertainty, and this will influence the expected uncertainty in the modulus. Regarding the interdependence of α and R, each of these values has their own measurement uncertainty while directly causing the value of the other. This will add a dependence on the rates of change of each of these measurements. This theoretical formulation of the error can be used to assess the contributions of each source of variance and compared to determine if any source is more strongly correlated with the variance in the modulus, which would provide a specific focus for error reduction when manipulating AFM imaging parameters.

2.4.3 AFM Tip Characterization

Understanding the AFM tip geometry is necessary for calculating the sample modulus via the DMT model since most AFM tips are not fabricated with a controlled geometry. Additionally, the mechanical interactions during indentation will erode the tip over time, degrading the fine tip point to a more rounded shape. An accurate determination of the tip shape is a critical part of the PF-QNM method calibration since the instrument relies on the operator to input a single tip radius value and control the indentation of the tip to depths near the height of the tip where that radius occurs. Determination of tip geometry via an analytical solution is generally not feasible due to the manufacturing tolerances of AFM tips and the unconventional geometric shape of most tips. For example, the Bruker RTESPA-525 tip shown in Figure 10 used in this study has a trigonal planar shape with the faces defined by the crystal plane angles of 15, 25, & 17.5° +/- 2° and a nominal tip height of 10-15μm. This imprecision in the nominal tip geometry parameters necessitates the use of indirect methods to characterize the tip.

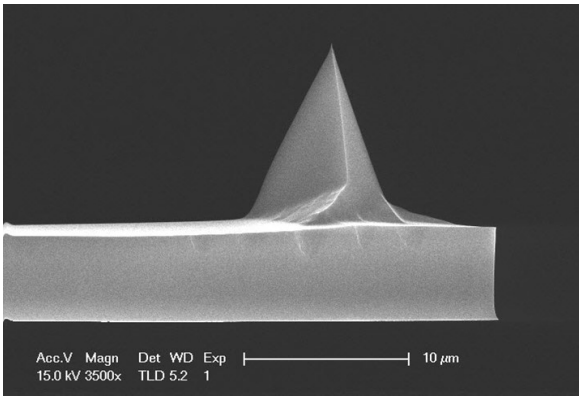

Figure 10. Scanning Electron Microscope Image of Bruker RTESPA-525 AFM Tip [14]. Reproduced with permission from Bruker Corporation.

The indirect method of tip qualification involves the imaging of a surface roughness

standard, such as the Bruker titanium standard shown in Figure 11, with many sharp surface features which can then be used to reconstruct the tip. The sharpness of the surface roughness features is the primary determinant of the accuracy of this method. When many features are imaged and the result combined, an inverse image of the tip shape is produced. While a tip characterizer with infinite sharpness would produce a perfect tip image, this impossible limit means that any tip reconstruction is a superposition of the tip shape and surface feature. By imaging many surface features, the characterizer can provide an upper bound limit on the tip shape [15].

As shown in Figure 12, a characterization standard sharper than the AFM tip will most accurately reproduce the tip shape, a smooth standard will interpret a rounded tip, and a standard of random surface roughness will produce some combination. Another disadvantage of this method is that, for AFM tips of very large size or with very stiff cantilevers, the tip may not be able to travel sufficiently deep into the roughness sample, or may damage the sample if the tip is much harder than the sample. Finally, it should be noted that while this method provides an accurate means to determine the tip shape, it cannot provide a direct measurement of the tip shape and thus contributes an indeterminate source of error in the calculation of precise values from PF-QNM images.

For the AFM tip used in this study, the Bruker RTESPA-525, nominal geometry parameters are provided that allow for a comparison of the nominal true tip geometry and the tip geometry necessary for use in the DMT model. The RTESPA-525 AFM tip specification is shown below in Table 1 and Figure 13. From the angle parameters, the surface area at any height from the tip can be calculated and compared to the theoretical volume or surface area of a spherical indentor, as required by the DMT model. These can then be compared and an upper indentation depth bound can be specified for use in AFM imaging.

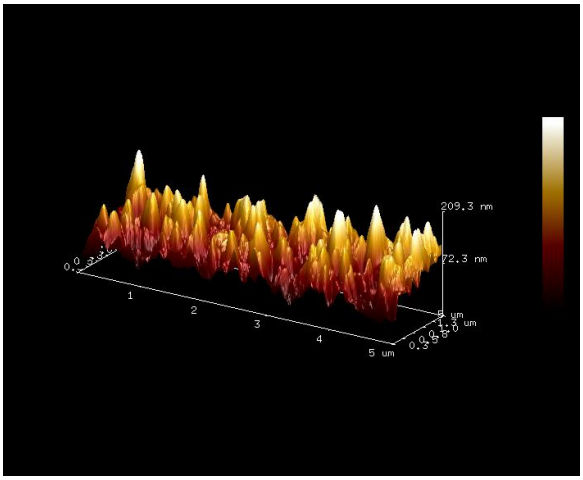

Figure 11. Image of Ti Surface Roughness Tip Characterization Standard.

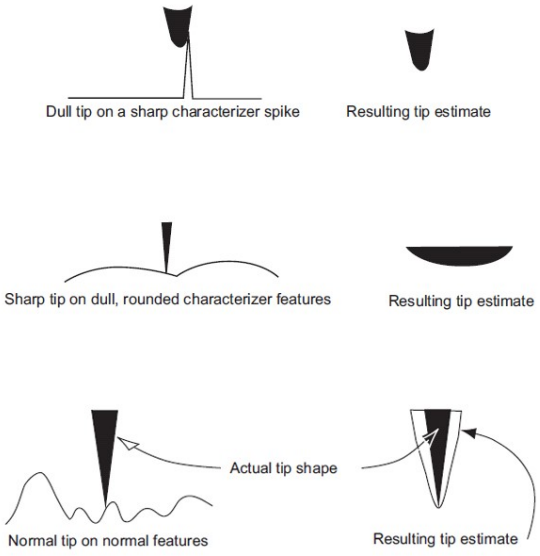

Figure 12. Tip Estimates Given by Different Surface Features [16]. Reproduced with permission from Bruker Corporation.

Table 1. RTESPA-525 Nominal Tip Geometry [14].

	Nominal Value	Variation
Tip Height (h)	10-15 μm	
Front Angle (FA)	15^o	2^o
Back Angle (BA)	25^o	2^o
Side Angle (SA)	17.5^o	2^o

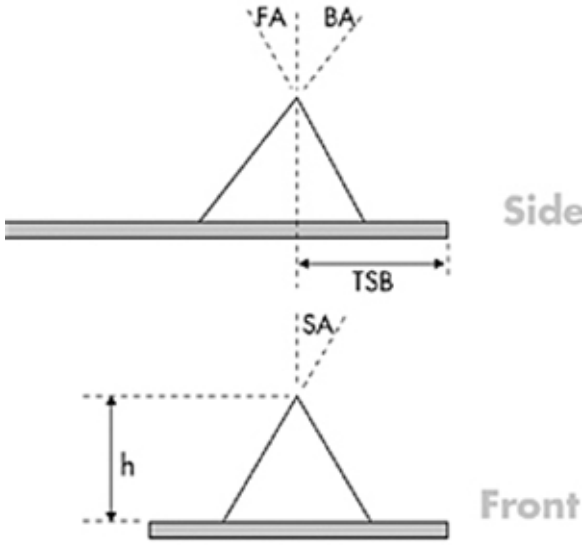

Figure 13. Bruker AFM Probe Geometry Specification [14]. Reprinted with permission from Bruker Corporation. TSB is the tip setback distance, the distance from the end of the cantilver to the vertical axis of the tip. FA, BA, SA, and h are as defined in Table 1.

The tip surface area is described by Equation 8, which is derived using trigonometric relations between the various triangular faces of the AFM tip.

$$SA_{tip} = 2\sqrt{p_1(p_1-f)(p_1-s)(p_1-a)} + 2\sqrt{p_2(p_2-f)(p_2-s)(p_2-b)} \tag{8}$$

$$p_1 = \frac{f+s+a}{2} \qquad p_2 = \frac{f+s+b}{2} \qquad f = \frac{h}{sin(90-FA)}$$
$$s = \frac{h}{sin(90-SA)} \qquad a = \sqrt{L^2+w^2} \qquad b = \sqrt{l^2+w^2}$$
$$L = h\frac{sin(FA)}{sin(90-FA)} \qquad l = h\frac{sin(BA)}{sin(90-BA)} \qquad w = h\frac{sin(SA)}{sin(90-SA)}$$

This relationship can be applied for any trigonal pyramid AFM tip with similarly defined geometry specifications

2.5 Carbon Fiber Production and Microstructure

The production of carbon fibers for use in reinforced composites is a well established industry with manufacturing and distribution across the globe, though the largest operations by weight occur in Japan and the United States, with the three largest producers by manufacturing capacity being Toray, HEXCEL, and Mitsubishi Chemical Carbon Fiber & Composites, with Toray producing slightly less than the next four largest producers combined [17]. Manufacturing firms produce a wide array of fiber types to support broad material requirements for tensile modulus and tensile strength, with fibers distributed as stranded tows of many single fibers or as pre-formed sheets or weaves. Finally, fibers can be provided "sized", to increase the surface energy of the fiber to improve adherence to a specific composite matrix,

or unsized without this treatment. Of interest to this study is the effect on fiber modulus and strength due to the different precursor materials and manufacturing processes, while the actual fibers used in this study are both sized and unsized based on availability from the manufacturer or existing materials on hand.

2.5.1 Carbon Fiber Microstructure

Carbon fibers, unlike the more highly ordered forms of carbon such as diamond or graphene, consist of many graphite crystallites organized into layered planes that are then formed into a fiber macrostructure [18]. These crystallites are generally oriented along the fiber axis with some deviation from parallel known as the misorientation angle, which has an average value of 30° for most fibers. Higher longitudinal modulus fibers are produced primarily by decreasing this misorientation angle, as shown in Figure 14. Increasing the axial alignment of the crystallites is normally done by increased heat treatment temperature and stretching. Increased heat treatment temperature also encourages the release of crystalline impurity atoms, further improving performance [19]. Due to the differences between the precursor materials, Pitch-based fibers require a lower heat treatment temperature to achieve the same orientation angle, as shown in Figure 15. This lower heat treatment temperature leads to lower production costs, and so the Pitch precursor is the prevalent form of ultra high modulus fibers. Perturbation in this heat treatment during production may result in the observed variation of the modulus from nominal values, as seen in the Mitsubishi fibers measured in this study, which had certified longitudinal modulus values between 3.28% and 1.08% greater than the nominal values.

PAN and Pitch-based fibers possess slightly different microstructures that lead to this variation in the amount of heating required to change the crystallite alignment angle. Additionally, high strength and high modulus fibers each possess slightly

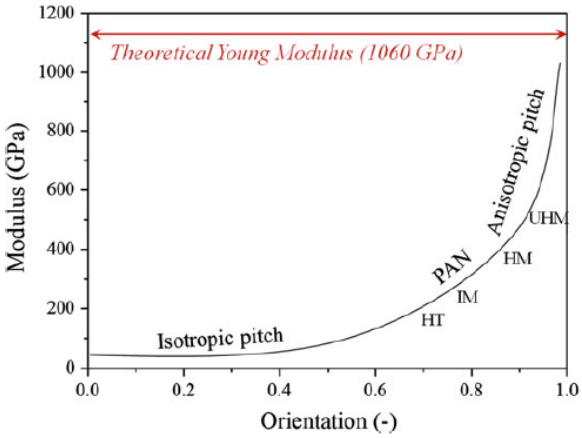

Figure 14. Correlation of Fiber Orientation Angle to Longitudinal Modulus. The Theoretical Young Modulus is for a perfectly oriented, defect free ideal fiber [20]. Reproduced with permission from Springer Nature.

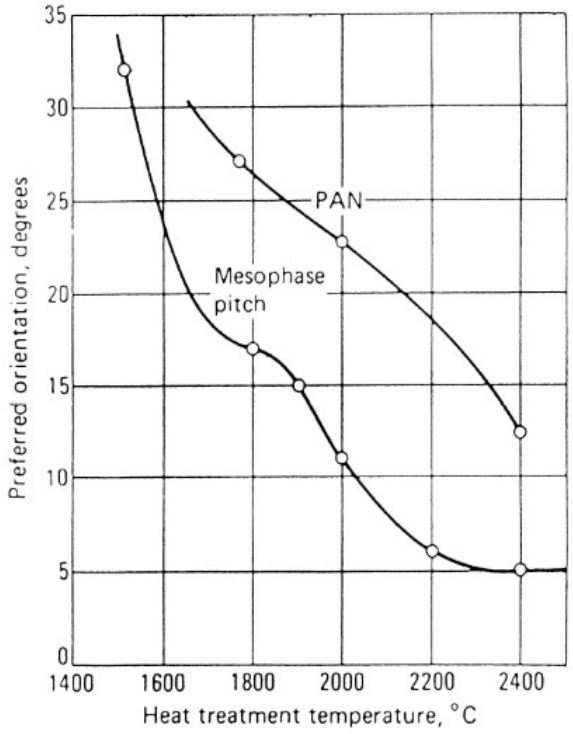

Figure 15. Pitch and PAN Fiber Orientation Angle with Heat Treatment °C [19]. Reproduced with permission from Springer Nature. As the heat treatment temperature increases the crystallites become more longitudinally aligned, resulting in an increased longitudinal modulus as shown in Figure 14.

different microstructures that lead to the difference in their characteristic mechanical property. PAN fibers possess a random crystallite orientation about the fiber axis, while Pitch fibers possess a more ordered radial folding microstructure, as shown in Figure 16.

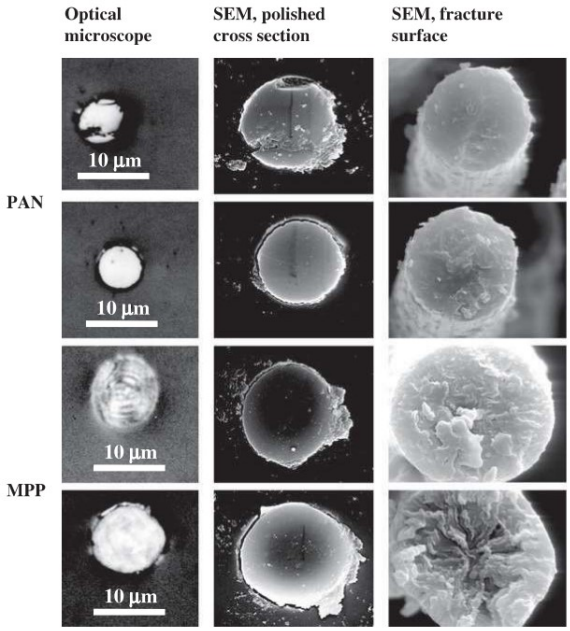

Figure 16. Cross-sectional Optical & SEM Images of PAN and Pitch-Based (MPP) Fibers [21]. Reproduced with permission from Wiley-VCH Verlag GmbH & Co.

The randomly oriented microstructure of PAN fibers leads to a larger average interplanar spacing between the graphite layers and a tendency for fibers to have many small, folded crystallites. This lack of orientation and the effect of many small crystallites failing and relieving the applied stress leads to the high tensile strength of these fibers, while this same misorientation and low degree of graphitization leads to weaker interplanar bonds resulting in the relatively lower tensile modulus of most PAN fibers [18].

Pitch-based fibers, on the other hand, show a large degree of order along the fiber axis while possessing varying levels of order perpendicular to it. This higher longi-

tudinal order allows for smaller interplanar distances and thus stronger interplaner bonding between the graphite crystallites. The overall 3D order, however, does not strongly determine Pitch precursor fiber properties. Pitch fibers in general possess a much greater tensile modulus due to the liquid-crystalline nature of the Pitch precursor, which allows for greater strain relief [18].

The different microstructural properties of the various fiber types also has an influence on other mechanical properties. Of interest to this study, as the fiber tensile modulus increases, it is expected that the transverse compressive modulus will decrease. As the crystallites become more longitudinally aligned they also become less transversely aligned, and so the same effects of crystallite alignment will apply. It has been shown that the structure/microstructure of carbon fibers are "clearly correlated with transverse compressive properties of PAN-based and Pitch-based single carbon fibers and especially with the transverse compressive modulus/strength and the Weibull modulus of transverse compressive strength," with the Weibull modulus being a statistical parameter that describes variability in mechanical properties of brittle materials [22]. Additionally, it has been demonstrated for Pitch-based fibers that both the transverse modulus and transverse compressive strength decrease with increased crystallite size, and that these differences cannot be attributed to variations in cross-sectional structure [23]. Instead, the transverse modulus is strongly determined by interplanar crystallite bonding, while the transverse compressive strength is determined by the strength of the inner fiber core, indicating that fibers should not be treated as homogeneous media for transverse compressive testing.

The exponential correlation shown in Figure 14 between the fiber longitudinal modulus and the crystallite orientation angle gives an indication of how the transverse modulus and longitudinal modulus are likely correlated. The transverse modulus is expected to decrease as the longitudinal orientation angle increases, and so the

transverse modulus must decrease in the same manner that the longitudinal modulus increases. Prior work by Veigas found a negative linear correlation between the transverse and longitudinal modulus for fibers between 231-377 GPa, generally within the high strength (HT) or intermediate modulus (IM) regime on Figure 14 [3]. For this study, which will also measure the transverse modulus of fibers between 434-924 GPA, the correlation is predicted to resemble that of the high modulus (HM) and ultra-high modulus (UHM) regimes, which begins to resemble a power law correlation.

2.6 Summary

This chapter has presented the theory of AFM instrument operations and the principles applied to produce AFM images via PF-QNM. It further describes the contact mechanics forces involved in a PF-QNM measurement, and how those forces and other AFM imaging data channels can be used to calculate the elastic modulus of a sample. Additionally, the variety of error sources in the AFM imaging method were described along with the accepted practices for minimizing their influence on an measured image, as well as how the error in the DMT modulus is dependent on the error of the physically measured variable of indentation depth, interaction forces, and the tip radius. It also described the importance of accurately characterizing the AFM tip, and the inherent uncertainty involved in tip characterization. Finally, the process of carbon fiber manufacturing was briefly described and the precursor-dependent variation in fiber structure between PAN and Pitch-based fibers were characterized, along with the expected impact of this microstructure on fiber modulus measurements.

III. Methodology

The methodology employed to achieve the research objectives was undertaken in two experimental areas. The primary area of experimentation is the development of a post-processing technique applied to the raw image data produced by the AFM, with the goal of reducing measurement error relative to that provided by built-in instrument analysis software. This post-processing technique was developed with the DMT modulus channel serving as the baseline data set, with both statistical techniques and numerical methods employed to determine whether other parameters that were theoretically similar to the DMT modulus could be derived from the image data that could be measured with better precision than the directly reported DMT modulus. The second area of experimentation, which supports the primary area, is the experimental measurement of carbon fibers with high and ultra-high longitudinal modulus using the PF-QNM AFM method, and the improvement of this method by modification of instrument settings and measurement technique. This area of experimentation both expands the available data set and to provides new measurements for previously characterized fibers to assess the relative improvement to the measurement precision and ensure consistency of measurement parameters within the data set.

For the purposes of this study, while the "true" transverse modulus value is the ideal measurement value, it is difficult to declare that the AFM reported values are the true transverse modulus since the reported values of modulus by a force-controlled indentation technique will be coupled to the selected indentation force setpoint. This study does not seek to improve the established PF-QNM method to definitively measure the "true" transverse modulus, but instead seeks to optimize the instrument settings and technique to achieve high levels of measurement precision. This improved precision allows for instrument-specific trends in the fiber transverse moduli to be determined, but are not necessarily the "true" transverse modulus of the fiber. Since

the method by which the instrument calculates the DMT modulus is not a direct measurement, but instead is based on a software calculation using other measurement data, such as peak force, adhesion force, and indentation depth, this study will describe the measured transverse modulus as the "relative transverse modulus", to clearly define that the values measured are relative to a consistent set of measurement parameters.

3.1 Methodology for Measurement of Fiber Transverse Moduli

Eleven different types of carbon fibers from three manufacturers were measured in this study. The first group of five fibers, possessing a tensile modulus ranging from 377-231 GPa, were measured in order to reproduce previous measurements by Veigas to provide a baseline set to compare the measurement precision. The second group of six fibers, ranging from 924-241 GPa, were measured to both increase the number of fibers in the available measurement set and to expand the set to demonstrate the application of the method to high and ultra-high modulus fibers, ensuring the method is valid for all commercially available carbon fiber types. Table 2 details the full set of fibers measured in this study. It should be noted that the Mitsubishi fibers were provided with a certified specification document that give the actual longitudinal tensile strength and longitudinal tensile modulus of the fibers, whereas all other fibers are listed with the manufacturer nominal strength and modulus. Fibers with certified properties are denoted with an asterisk in Table 2. Additionally, IM9/G-12K fibers are no longer listed by HEXCEL as available for purchase, and so the tensile modulus reported by Veigas is used, while tensile strength is unavailable.

Table 2. Fibers of Interest.

Fiber	Precursor	Manufacturer	Longitudinal Modulus [GPa]	Longitudinal Strength [GPa]
K13C2U [2]	Pitch	Mitsubishi	924*	3.94*
K63A12	Pitch	Mitsubishi	800*	3.14*
K1352U	Pitch	Mitsubishi	647*	3.67*
K63712	Pitch	Mitsubishi	642*	2.85*
TRH50 [24]	PAN	Mitsubishi	255	5.6
34-700WD	PAN	Mitsubishi	234	4.83
HM63 [25]	PAN	HEXCEL	434	4.826
IM9/G-12k [3]	PAN	HEXCEL	304	
AS4D	PAN	HEXCEL	241	4.723
AS4-GP-12K	PAN	HEXCEL	231	4.413
M40JB [26]	PAN	Toray	377	4.40

3.1.1 Reproduction of Low Modulus Fiber Measurements

Measurements of the fibers characterized by Veigas are conducted in order to provide direct comparison between methods, specifically to assess how varying imaging parameters such as the PeakForce Setpoint change the measured modulus values and/or the measurement error. Prior measurements performed by Veigas will be discussed in the context of comparison to determine if and how variation of instrument settings led to any reduction in precision or in the variation of the fiber transverse modulus

3.1.2 Measurement of High and Very High Modulus Fibers

The purpose of expanding the measured sample set is to account for a larger variety of carbon fiber classifications and precursor types. Research by Veigas utilized high strength fibers with tensile strengths ranging from 4.41-6.5 GPa and longitudinal modulus ranging between 231-377 GPa, exclusively using PAN-based fibers.[3] Additional fibers in the high and very high modulus ranges from three manufacturers will be measured to demonstrate the application of the PF-QNM measurement method-

ology to all commercially available carbon fibers classification types. The new fibers span tensile strengths from 2.6-4.826 GPa, a similar span to Veigas' sample set, but with tensile moduli spanning from 241-935 GPa, a range about 4.75 times broader than Veigas' sample set. In addition to conducting modulus measurements across a broader modulus range, the addition of Pitch-based carbon fibers will account for fibers possessing a different precursor material, manufacturing process, and internal microstructure. The fiber classification by strength/modulus properties are shown in Figure 17.

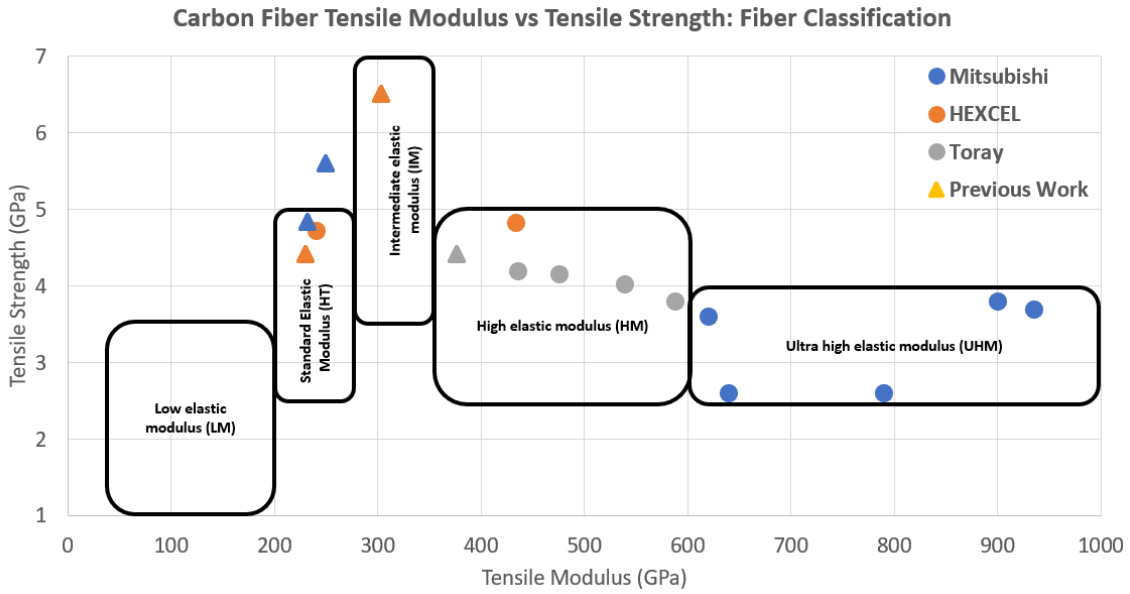

Figure 17. Modulus Classification of Analyzed Fibers. [27]

3.1.3 Sample Preparation Procedure

Fiber sample preparation was performed in accordance with the method developed by Veigas. A ~1cm section of carbon fiber tow was cleaved from the bulk using a knife, and small bundles of fibers were teased from this section to progressively separate out single fibers. These small fiber bundles were then mounted on a 12mm sample disc

using less than 1ml of cyanoacrylate resin adhesive, which was allowed to cure for about one hour. The fibers were then swabbed lightly with a cotton swab soaked in methanol to dissolve any adhesive that may have wicked onto the fiber surface. The samples were then allowed to cure a further hour and swabbed a second time. The fibers were then passed into the AFM glovebox and arranged on the sample tray, where they cured overnight. Table 3 shows the number of samples prepared and measured for each fiber type.

Table 3. Fiber Sample Preparation.

Fiber	Precursor	Manufacturer	# of Samples Measured
K13C2U	Pitch	Mitsubishi	5
K63A12	Pitch	Mitsubishi	6
K1352U	Pitch	Mitsubishi	10
K63712	Pitch	Mitsubishi	15
TRH50	PAN	Mitsubishi	11
34-700WD	PAN	Mitsubishi	9
HM63 [25]	PAN	HEXCEL	22
IM9/G-12k [3]	PAN	HEXCEL	16
AS4D	PAN	HEXCEL	28
AS4-GP-12K	PAN	HEXCEL	19
M40JB [26]	PAN	Toray	7

In order to characterize the properties of the cyanoacrylate adhesive after treatment with methanol, samples were also prepared without any fiber and measured using the AFM to determine the cured adhesive modulus value, which would serve as a boundary value for the accurate measurement of the fiber transverse modulus.

3.2 Methodology for Error Reduction

The objective of this work is to develop and demonstrate a method for reducing the measurement error associated with a previously developed PF-QNM AFM method for determining the carbon fiber transverse Young modulus. This improvement is

achieved by a combination of instrument parameter optimization and measurement data post-processing using statistical analysis. Instrument parameter optimization is focused on the parameters governing direct AFM tip-fiber interaction and the means by which the various errors in the measured interactions governing the AFM force curve, such as the peak force, adhesion force, indentation depth, and tip radius, can have their individual errors reduced. It was assumed that by reducing these errors, the error in the DMT modulus measured by the instrument would be reduced, though it was not known a priori which of these errors was most strongly correlated with the modulus error, nor was it known whether any of these individual errors were themselves correlated or coupled. This was investigated separately.

3.2.1 Instrument Parameter Approaches to Error Reduction

The PF-QNM AFM method uses a large number of instrument settings to allow the researcher fine control over the imaging of a wide array of materials. For the purposes of this study, only the instrument settings that were expected to strongly influence the tip-sample interaction mechanics were examined. The PeakForce Setpoint controls the maximum applied tip force, and so is the most direct means of varying the tip-sample interaction. Additionally, an optimal spatial dimension for the measurement is used to ensure the fiber surface height change is small relative to the image size. Finally, the image resolution was held constant for all images, while the scan rate was allowed to vary under instrument control based on image size. The setting ranges used for imaging each fiber, and the adhesive, throughout this study are shown in Table 4. Finally, to prevent measurement error associated with piezoelectric drift after instrument startup, an instrument warmup period of 20 minutes was used.

Variation of the PeakForce Setpoint was performed for four of the fibers studied

Table 4. Instrument Imaging Parameters.

Fiber	Peak Force Setpoint [nN]	Peak Force Amplitude [nm]	Image Area [μm^2]	Resolution	Scan Rate [$\mu m/s$]
K13C2U	569.9	100	2.73-11.02	128x128	0.25
K63A12	456.4-569.9	100	0.098-0.705	128x128	0.25-0.50
K1352U	499.7	100	1.03-25	128x128	0.25-0.50
K63712	473.4-500.4	100	6.25-25	128x128	0.50
HM63	456.4-569.9	100	0.021-3.99	128x128	0.25-0.50
M40JB	500	100	0.165-4.08	128x128	0.50
IM9	456.4-3499	100	0.250.67	128x128	0.50
TRH50	149.9-3031	100	0.001-0.56	128x128	0.50
AS4D	399.4-1299.8	100	0.027-0.71	128x128	0.50
34-700WD	3000-3185.4	100	0.002-0.56	128x128	0.25-0.50
AS4-GP	468.8-3000.3	100	0.67-4.09	128x128	0.50
Adhesive	400-436.2	100	0.39	128x128	0.25-0.50

by Veigas for which prior raw data was available (IM9, TRH50, 34-700WD, & AS4-GP-12K). The study was limited to these fibers in order to have a directly comparable set of measurements at different setpoints. The setpoint was decreased from the value used by Veigas, $\sim 3\mu$N, by about one order of magnitude to $\sim$400nN.

Decreasing the PeakForce Setpoint is expected to decrease the magnitude of variation in the PeakForce, described as PeakForce Error by Bruker. A decreased PeakForce Error magnitude should lead to a more consistent indentation depth, tip radius at this depth, and adhesion force. For a perfectly flat sample and an invariant PeakForce, these values should also be invariant. Since the carbon fibers are curved, however, the only operator-controllable means of decreasing the variation in the indentation, tip radius, and adhesion force is to decrease the variation in the applied tip force. Since the instrument operates using a force feedback loop, force variation is an intrinsic measurement error that itself can only be reduced in absolute magnitude. For example, a 5000nN PeakForce with 10% PeakForce Error will have a 500nN PeakForce Error magnitude, while a 500nN PeakForce with the same 10% error will only have a 50nN error magnitude. Since the PeakForce must be above some thresh-

old value to actually indent the fiber, this PeakForce error can only be minimized in magnitude and not eliminated, while it is expected that the variation in the adhesion force and indentation depth would decrease relative to the smaller PeakForce error amplitude.

The optimal measurement size is expected to be smaller than the actual fiber width of about 5μm. While the variation in height observed with the AFM is useful for differentiating fiber from adhesive, and for locating the ridges and valleys characteristic of the fiber microstructure, this variation can be as large as 500nm over a 5μm image. This large difference in height is expected to increase the measurement error due to surface roughness effects, where the tip does not indent normal to the fiber surface. This order of magnitude difference is observed to be consistent as the image area is decreased. A measurement area of approximately 500nm^2 produces a height variation less than 50nm, and the image appears flat.

3.2.2 Statistical Approaches to Error Reduction

The statistical post-processing of the data involves two approaches that can be utilized simultaneously or separately. The first approach focuses on improving individual fiber image data quality by excluding extreme values in each of the data channels. This entails determining an appropriate cut-off value for each channel that allows for exclusion of very small and very large values that are not characteristic of the measurement and would otherwise bias a modulus calculation. For example, the AFM will occasionally measure negative indentation depths, resulting in points with moduli on the order of TPa. Though these points are usually few in number, the points will shift the mean of the modulus measurement and must be excluded. Another example are points with large indentation depth. In these cases, the tip may indent by one or two orders of magnitude greater than the rest of the measurement.

In this case, the indentation is likely the result of some contaminant on the fiber surface, such as cyanoacrylate adhesive, which can also be identified by inspection of the adhesion force data. These points result in moduli on the order of MPa, and must be excluded for the same reason as the very small indentation depth.

The second statistical method for data processing in this study focuses on improving the quality of the overall fiber measurement data set by excluding whole images based on specific criteria. The first criteria is based on individual measurement error, and excludes any images regardless of reported modulus that has measurement error greater than a defined exclusion value. The second criteria excludes measurements that are indistinguishable from the cyanoacrylate adhesive used to mount the fiber samples. While it is possible that the fibers could have the same modulus as the adhesive, it is expected that this result would be more likely to occur from poor sample preparation, and so these images are excluded.

Finally, the third criteria excludes high modulus outlier images. Earlier exclusion criteria remove low modulus outliers, and so a similar criteria is established in order to prevent high modulus outliers from influencing the sample set. It is known from prior study that the PF-QNM method can report occasional high modulus values due to improper installation of the tip, or due to poor sample mounting. These high modulus outliers are identified by analysis of the complete fiber sample results and removing any measurements with a modulus value larger than 150% of the mean for that fiber.

3.3 Post-Processing Methodology

The methodology described above will be performed through the direct manipulation of raw AFM data files output from the instrument. Since the instrument is designed for imaging a variety of materials and using many different imaging modes,

the optimal output file types and analysis software procedure must be determined to ensure that the desired measurement data are exported for further statistical analysis. This study assumes that a simple method with low computational cost is preferred in order to conduct a greater number of measurements in the same time frame. Additionally, this study develops a method requiring minimal access to proprietary software or actual instrumentation, and would be usable on any computer workstation. This study uses the Bruker NanoScope Analysis software, which can be attained at no cost directly from the Bruker Corporation website, for viewing and exporting AFM image data.[28] This data is then imported into and manipulated with a Python script, included in Appendix A.

In developing the post-processing method, this study assumes that it is feasible to replicate the instrument software technique wherein the DMT modulus is calculated by slope fitting a linear section of the AFM force curve. Using a scripted routine that allows for more selective exclusion of certain portions of the force curve, this study use multiple approaches to determine if an alternative calculation method can achieve the same or better precision as the instrument software.

The first calculation method used in this study is to directly apply the DMT modulus formula, Equation 6, using the measured indentation depth, interaction forces, and tip radius. This method assumes that the measured values in these data channels do not differ significantly from those used for the instrument's modulus fitting routine. This method is further expanded by adding additional known material data, such as the AFM tip modulus and Poisson Ratio, in order to deconvolve the AFM tip material properties from the sample, of which the former is otherwise assumed to be much, much harder than the sample. Additionally, since the Poisson Ratio is poorly characterized for carbon fibers, this method allows variation of the Poisson Ratio to provide an approximate error bound based on inherent imprecision in this parameter.

This method also allows the exploration of variation due to the use of different tip fit functional formulations.

A second calculation method is also proposed and described wherein the AFM force curves are directly analyzed and a DMT modulus fitting routine is employed. This method is not utilized in the characterization of the complete fiber sample set, due to its significantly greater computational cost and NanoScope Analysis software limitations, however the method is described and an exemplar measurement is presented for completeness and to demonstrate its feasibility for future research.

Each of the curves in the ensemble derived from a single AFM measurement, shown in Figure 18, is used to calculate the DMT modulus using the slope fitting technique described earlier as used by the Bruker software.

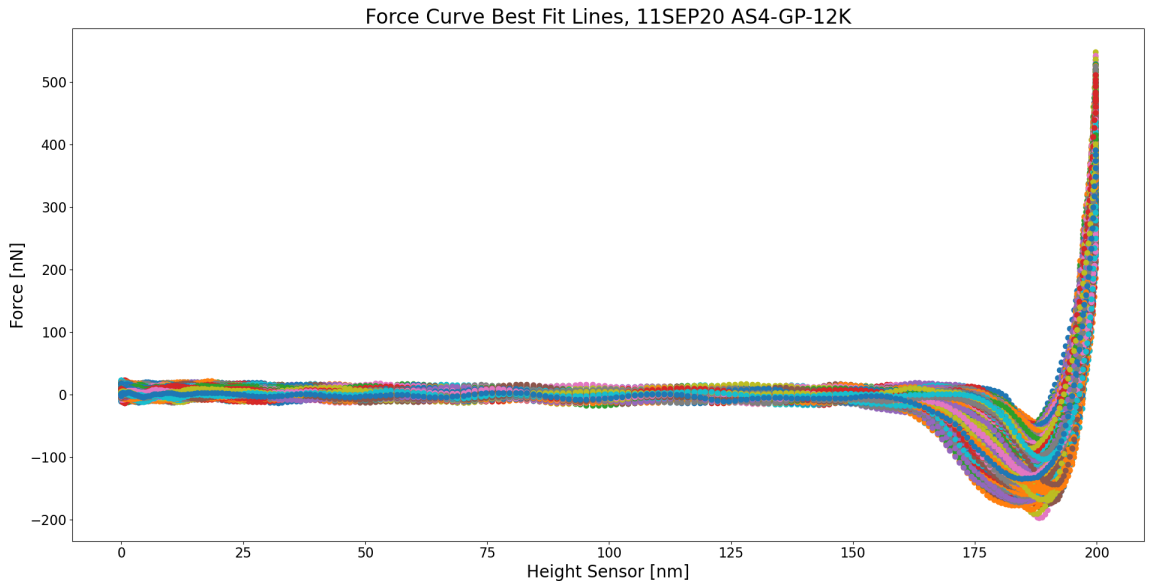

Figure 18. Ensemble of Force Curves for a Single AFM Image.

The modulus for each force curve is determined by selectively fitting the slope of the force curve between the PeakForce and the Adhesion Force, in the same manner as the instrument software described earlier. The fit region was defined between

80% and 40% of the total force amplitude, that is the sum of the Peak Force and Adhesion Force. The region was then fit to Equation 9, where F and α are the force and indentation depth respectively, a is the fitting parameter containing the Young's modulus, and b is a fitting parameter to account for bias in the force curve baseline.

$$F_{net} = a\alpha^{2/3} + b \tag{9}$$

The Young modulus is then calculated from the calculated fit parameter a, as shown in Equation 10.

$$E = \frac{a}{4/3\sqrt{r_{tip}}} \tag{10}$$

The mean modulus and standard deviation are then determined from the collected ensemble of values.

3.3.1 Use of Instrument-Specific Analysis Software

The Bruker Dimension Icon used in this research outputs multiple different file types while capturing an image, none of which can be directly interpreted by a user and require analysis software to interpret and reproduce a sample image. Two file types provide the specific data for the complete image, the first being the .spm file, and the second is the .pfc file. The .spm file is a raw data file containing image parameters and image information, while the .pfc file is a similar file that also contains the required information to recreate all force curves generated during imaging. The Bruker Dimension Icon AFM is furnished with the image analysis software NanoScope Analysis, which is capable of interpreting these raw instrument data files and outputting a data channel text file that can be used for calculating point-wise image data or full image recreation.

The third file type produced by the AFM is output when the user enables "high speed data capture" during imaging, and generates a .hsdc file. The .hsdc file is an alternative means of capturing the complete force curve data from the image, as in the .pfc file, but has the advantage of including a larger number of force curves and outputting one file for each imaging line. This file type, however, also does not provide the raw data of the image and must be processed by the NanoScope Analysis software.

The chosen file type used for analysis in the post-processing script is imported into NanoScope Analysis, which is used to export a .txt file containing the actual measured data, in the case of the .spm file type. For the .pfc and .hsdc files, however, the software exports multiple files containing force curve data in a proprietary format again, which must be imported into NanoScope Analysis again and the .txt file containing the actual force curve data exported. During the course of this study the process of exporting the true raw force curve data from the .hsdc or .pfc files was found to be exceptionally laborious and time consuming relative to the .spm, with the time to export raw data from the .spm on the order of seconds, while exporting a full image worth of force curves could take as long as one hour.

Due to the ease of use, this research utilized only the .spm image files captured by the instrument and data text files exported from NanoScope Analysis, which contained all the information required to recreate an image and produce the data for calculation of the fiber modulus. The .pfc and .hsdc files were also captured for future study, but were not utilized to produce the results of this study.

The NanoScope Analysis software is also used to determine the AFM tip shape, which is a necessary requirement to determine the tip radius at the indentation depth where the modulus is calculated. A study was conducted of all new AFM tips used to characterize the variation in tip radius with height. Both a power law fit and

polynomial fit were used to determine the optimal functional representation of the tip over a 20nm indentation depth interval. This functional representation of the tip was then incorporated into the data analysis script and used for comparison between the AFM measured modulus and the script calculated modulus.

3.4 Determination of Image Exclusion Criteria

In order to improve the quality of the measurement data prior to performing statistical analysis, a number of studies were performed to determine appropriate criteria for excluding certain indentation measurements or entire images. The first study conducted was of the AFM tip. First, a comparison between a power law and multiple polynomials was performed to determine the best method for approximating the tip radius, which could then be used in the DMT model formula to calculate the modulus, rather than using the instrument reported value. The second study was performed to measured the modulus of the cyanoacrylate adhesive. Finally, a set of statistical criteria were established to exclude images characterized by larger than expected measurement error.

3.4.1 Exclusion of Outliers Within a Single Image

Exclusion of extreme values in each of the PeakForce, adhesion force, and indentation depth data channels was performed in order to improve the image quality by removing measurement points that are characterized by extreme values in the measurement distribution. The first cut-off that is applied removes all measurements with indentation depths larger than 20nm, chosen due to the applicable range for the DMT model and the tip radius fit model. A second cutoff is then applied to remove measurements where any single data channel value is more than two standard deviations from the mean value in that channel, with the assumption that the overall

distribution of each data channel was Gaussian.

3.4.2 AFM Tip Shape Modeling

As described earlier, the Bruker AFM PF-QNM method applies the DMT contact mechanics model to determine the sample modulus. The instrument requires a single tip radius input in order to develop a sample image using the DMT modulus, which assumes a constant indentation depth over the entire image area. Since the AFM produces measurements of all necessary data to calculate the DMT modulus, a more precise calculation of the DMT modulus requires determination of the tip radius at each individual indentation depth. Since the AFM can only receive a single tip radius as an input image parameter, and the NanoScope Analysis software can only characterize a tip in 0.5nm intervals, a functional approximation is used to determine the tip radius using the tip characterization data. This indentor radius is the approximate AFM tip radius, with the assumption that the AFM tip can be treated as a 3D parabola.

Tips are provided with either nominal tip geometry parameters or with a certified calibration. The necessity of obtaining an accurate tip characterization is demonstrated in Figure 19, which shows how the characterized tip radius differs from the tip radius calculated from both the nominal tip geometry, based on a trigonal planar tip shape, and a tip radius that assumes a perfectly spherical tip. While the difference is not large, about 5 times larger than the nominal tip geometry and twice as large as an ideally spherical tip, an accurate tip shape characterization will improve the accuracy of the AFM measurement of the DMT modulus.

The fabrication of the Bruker RTESPA AFM tips used in this study entails etching the tip from a crystal substrate, resulting in a shape that is a trigonal pyramid, rather than the ideal parabola required by the DMT model. The AFM tip was characterized

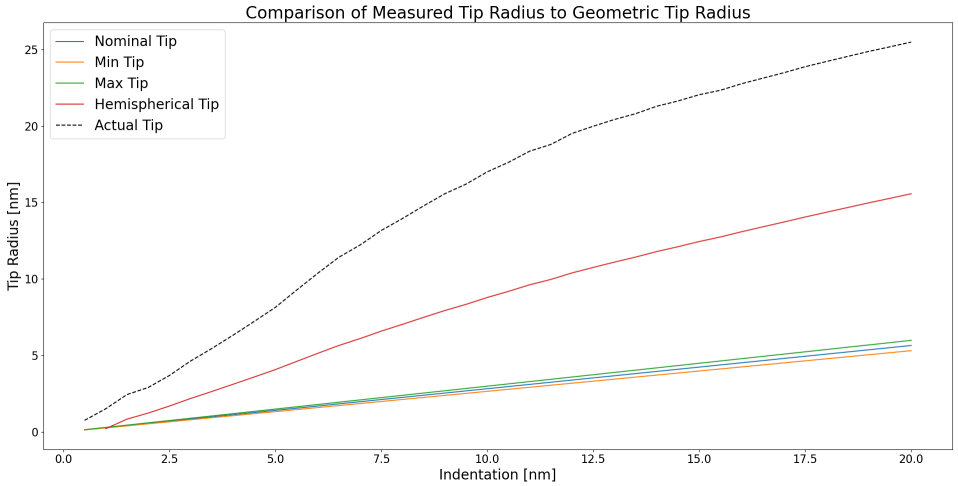

Figure 19. Tip Shape Model Fit for New AFM Tip

using the Bruker NanoScope Analysis tip characterization method using a titanium roughness standard furnished with the instrument. Tip radii were measured at 41 height intervals from 0.5-20nm and numerically fit as a function of the indentation depth z to both a power law function, Equation 11, using SciPy curve_fit method and a 3rd order polynomial, Equation 12, using the Numpy polyfit method, with the quality of fit quantified via the R^2 value. In these equations, r_{tip} is the calculated radius of curvature at distance z from the vertex of the tip, while parameters a-f are the fit constants. Each unique tip will possess a unique set of parameters. The results shown below are presented for a single new exemplar tip, which were consistent for all other new tips used in this study. Tips worn over time diverged from these results, however, and are presented later.

$$r_{tip} = f(z) = az^b \tag{11}$$

$$r_{tip} = f(z) = cz^3 + dz^2 + ez + f \tag{12}$$

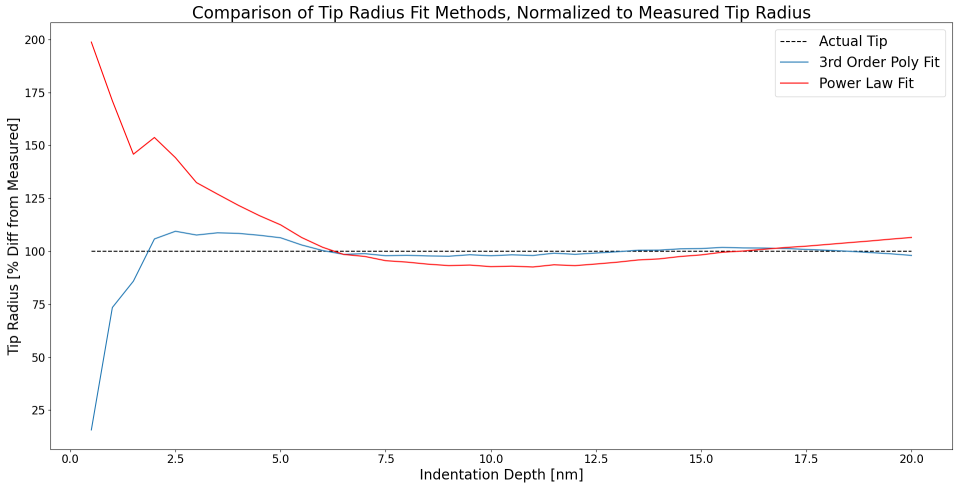

Figure 20. Tip Shape Model Comparison for Power Law and Polynomial fits.

A new AFM tip possesses a good fit to a power law function with $R^2 = 0.98221$ and a slightly better fit to the 3rd order polynomial with $R^2 = 0.99822$. When directly compared, as shown in Figure 20, with tip radius values normalized to the measured values in order to exaggerate small differences, it is clear that the polynomial method more closely matches the measured tip radius across a broader range in indentation depth. Both fit methods show extreme behavior as indentation approaches 0nm, which is expected since the tip radius at zero height must also be zero. This divergence from the measured radius at low indentation depth is also not expected to significantly influence characterization of the fibers in this study, since the instrument method uses a target indentation depth between 4-10nm. Variation in the actual indentation depth, or a very low AFM Peak Force Setpoint, could however lead to a larger number of shallow indentations, and so the behavior of the fit function must be kept in mind while optimizing the instrument settings for use with these models. A further study of fourth and fifth order polynomials was performed to determine if a better fit could be achieved.

For the comparison of polynomial fits, the Numpy polyfit method was applied

to polynomial functions of a similar form to Equation 12, but expanded to fourth and fifth orders as well. The correlation between the polynomial tip models and the measured data, for this exemplar tip, are shown in Figure 21. Both the fourth and fifth order polynomials more accurately replicate the measured tip radius at values above 2.5nm than does the third order polynomial. All polynomials tested demonstrate extreme behavior as indentation depth approaches zero, which is expected due both to the nature of fitting a truncated range with a polynomial, and because the tip radius at zero indentation is forced to equal zero. Based on these results, a fifth order polynomial was selected as the optimum fit method, while an indentation depth maximum of 20nm was selected to ensure a valid tip fit range.

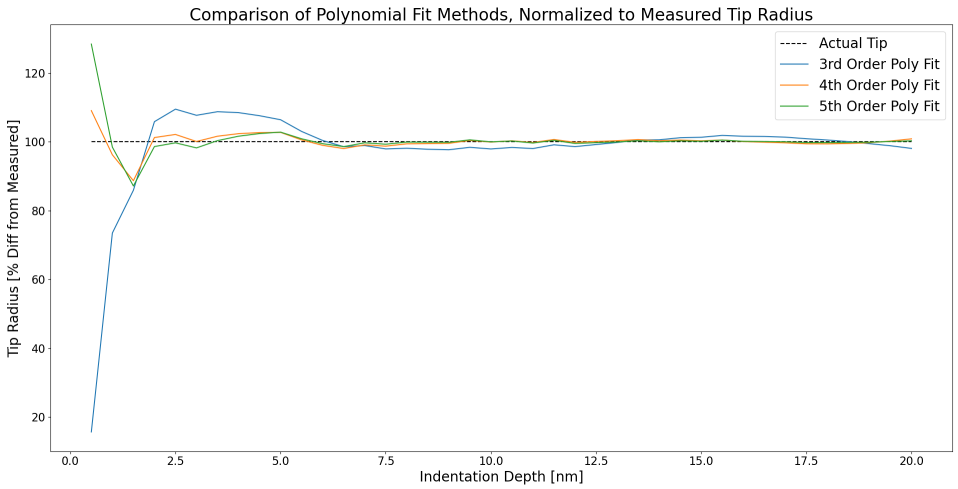

Figure 21. Tip Shape Model Comparison for Different Polynomials.

This study of tip shape modeling for a new tip is important, it should be noted that the tip will become worn over time and change shape due to mechanical interaction during indentation, and during scanning if a tip crash occurs. The change in tip shape due to indentation is relatively uniform since it is assumed the tip indents normal to the sample surface. Variations in tip wear due to surface roughness or sample orientation, which results in indentations off axis from the normal, were not

studied but could lead to asymmetric tip wear. A tip crash, in which tip motion in the scan direction is more rapid than the tip height can be adjusted for a rapid height change, can result in more drastic tip wear better defined as breaking. A tip crash often results in a large section of the tip being shorn off, leading to more than an order of magnitude difference in the tip radius. For these reasons, criteria for tip exchange were explored to clearly define when a tip was too worn for further use.

A study of the change in tip shape due to sustained usage was explored for three levels of wear. A new AFM tip was compared to an AFM tip that was worn solely by imaging and an AFM tip that had experienced a tip crash. The measured tip radii are shown in Figure 22. The most pronounced feature of the worn tips is the divergence in shape beginning at 2.5nm for the crashed tip. Since a tip crash is likely to shear a section of the tip, it is likely that this divergence is actually the true tip, while the sharper section is an artifact of the image characterization algorithm. The worn tip also exhibits divergence from the new tip, however it occurs between 4-6nm, a range that is similar to the range of indentations produced by the instrumental method on carbon fibers. This comparison shows the qualitative difference between tip crash and imaging use wear for the actual tip radius, and also identifies a criteria for immediate tip exchange after a tip crash.

For a worn tip, the tip exchange criteria must be based on a combination of the required image resolution and the number of images required for a fiber. For the PF-QNM method, increased image resolution will increase the number of indentation measurements performed on the fiber, allowing for more data to be collected and improved measurement statistics. This study used a 128x128 image resolution, resulting in 16,384 measurements per image, which is the largest resolution that allows export of all data file formats. The number of images taken for a single fiber is determined by the level of precision required. For this study, the target precision level was chosen

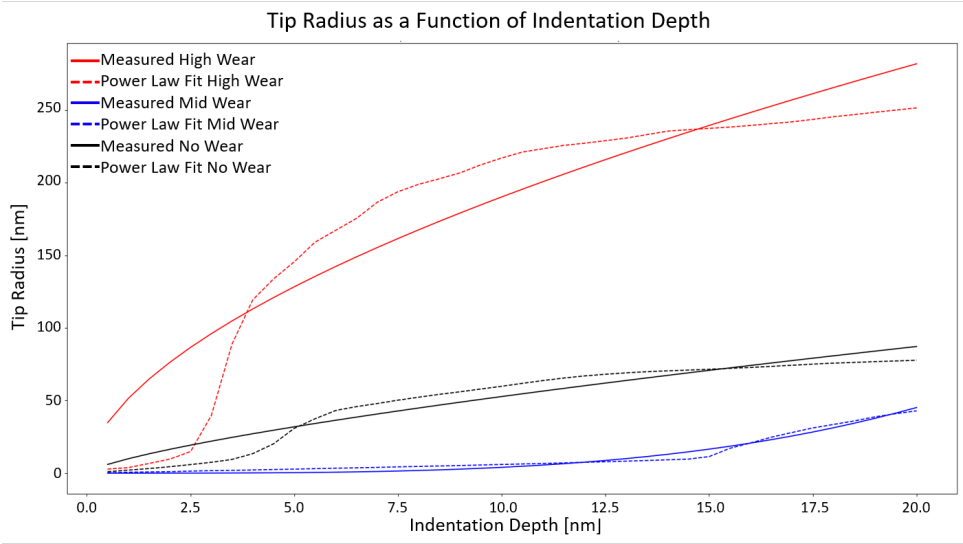

Figure 22. Tip Shape Model Fit for Various States of Tip Wear.

as less than 5% measurement error, which required between five and twenty images.

A study was made of the tip shape change over a large number of indentations to determine the degree of wear sustained by the AFM tip. This study was made for twelve measurements, about 200,000 indentations of the Hexcel HM63 fiber with a longitudinal modulus of 434 GPa, at a 128x128 resolution and PeakForce Setpoint of 569.9 nN. Since there are also uncountable indentations made in the course of selecting an area of fiber for measurement, the number of total indentations performed is approximate. Twelve images is also the maximum number of images reasonably achieveable in a single 8 hour day, assuming thirty minutes per image. Figure 23 shows the absolute increase in tip radius after the twelve images, while Figure 24 shows the relative increase.

The images performed with the tip used for this study had mean indentation depths of about 4nm +/- 10%. The largest relative wear occurs near the mean indentation depth, while above 7nm the relative change decreases at a constant rate. The very large relative change at 0.5nm is due to the large relative change between the

very small radius at the tip point. If the peak force, adhesion force, and indentation depth values were the same for both the first and last image, a ∼50% increase in the tip radius would result in a ∼11% increase in the calculated DMT modulus. This result, however, is less than observed in practice, with the weighted modulus error of the twelve measurements being 16.72%. This is expected to be due to the other contributions to the modulus variance as described earlier, which are observed to change from image to image.

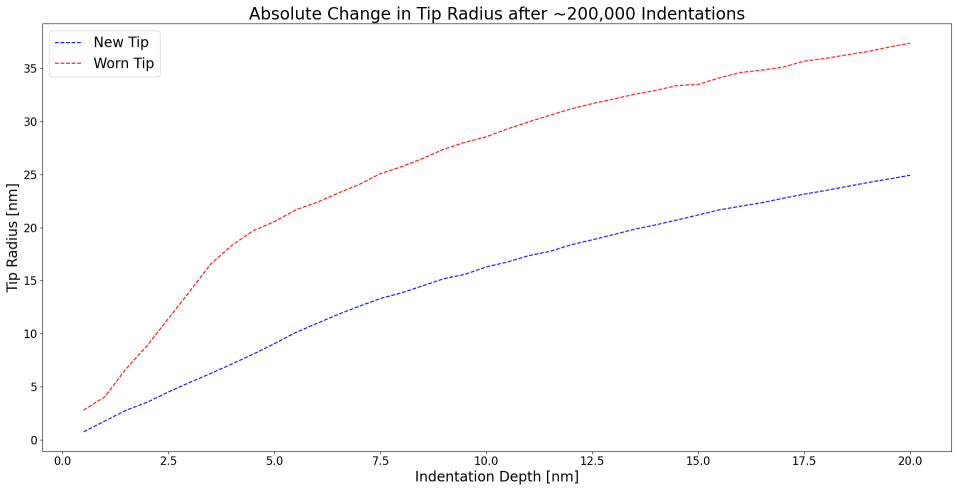

Figure 23. New and Worn AFM Tip Radius after ∼200,000 Indentations.

Since tip wear due to short-term use does not appear to cause an increase in the measured modulus, a reasonable tip exchange criterion was selected to be either exchange after ten images of the same fiber type or every time a new type of fiber was measured, whichever was most frequent. This approach ensures that each fiber type was measured with a new tip, while the process of taking many fiber measurements is not disrupted by frequent time consuming tip exchanges. This approach is also easily integrated into an eight hour daily measurement program, beginning with one hour of tip exchange and calibration, followed by six hours of imaging.

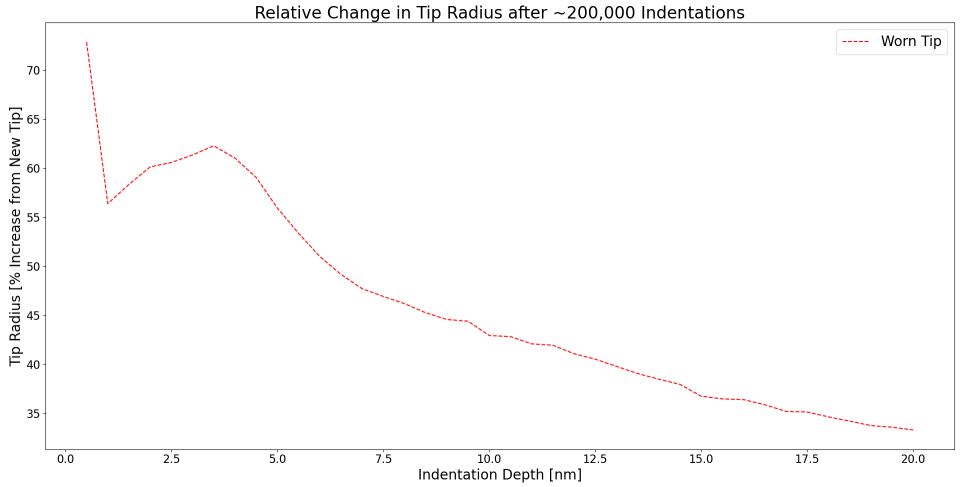

Figure 24. Relative Increase in Tip Radius after ~200,000 Indentations.

3.4.3 Measurement of Cyanoacrylate Adhesive Young Modulus

The Young modulus of the cyanoacrylate adhesive used to mount the carbon fibers was measured using the PF-QNM method to establish a precise value that could be used to better differentiate uncontaminated fibers from fibers that had wicked adhesive onto their surface. Additionally, a precise determination of the adhesive modulus would support the establishment of an upper and lower modulus bound where the fiber could not be differentiated from the adhesive. Since the adhesive is an amorphous liquid prior to setting into a solid, its material properties are assumed to be isotropic. Two sample discs were prepared with adhesive and swabbed with methanol, using the same method as a fiber sample, but without mounting the fibers. Eleven measurements were performed using the PF-QNM method, with a PeakForce Setpoint between 400-436.233nN and a resolution of 128x32 for 4,096 indentations per image. The error weighted mean for the adhesive measurements was determined to be 10.57 +/- 0.113 GPa, an error of 1.07%. The Young's modulus of the cyanoacrylate adhesive was used as a threshold criterion for excluding fiber measurements whose

mean values were less than the measured adhesive modulus.

3.5 Summary

This chapter has presented the methodology used to measure the transverse modulus of single strand carbon fibers, as well as the statistical and physical criterion to be used for improving the quality of both single image data and fiber type sub-sets of the eleven fibers to be measured. It also describes two alternative methods that will be tested for use in calculating the transverse modulus from the measured peak force, adhesion force, and indentation depth directly, in lieu of relying solely on the instrument reported value.

IV. Results

4.1 Overview

This section describes the results attained with regard to improving the precision of the transverse modulus measurement of single strand carbon fiber fragments. The transverse moduli determined after applying statistical exclusion criteria are presented, followed by transverse moduli calculated via three different post-processing methods. Finally, potential error effects due to material assumptions are also presented, as well as the correlations between the measurement errors for other instrument data channels.

4.2 Measurement of Fiber Transverse Modulus

The transverse modulus of all fibers were measured using PF-QNM and the data set was treated with the exclusion criteria described earlier. The correlation between the fiber transverse and longitudinal moduli was then determined for each precursor type separately, and for all eleven fibers together.

4.2.1 Exclusion of Extreme Values

The result of removing outlier measurements within single AFM images is found by analyzing the change in the distributions in the Peak Force, adhesion force, and indentation depth data channels before and after extreme value removal for a single fiber measurement. Exemplar results are shown for a K13C2U fiber image in Figures 25-32.

The PeakForce distribution, Figure 25, transforms to a distribution that more clearly demonstrates the non-continuous change due to the nature of the feedback loop variation from the PeakForce Setpoint, shown in Figure 26. Alternative binning

methods for the histogram do not remove this jagged appearance. The tails of the distribution are truncated as well, decreasing the absolute range of the PeakForce variation from 94.37 nN to 26.96 nN, a 71% decrease in this range.

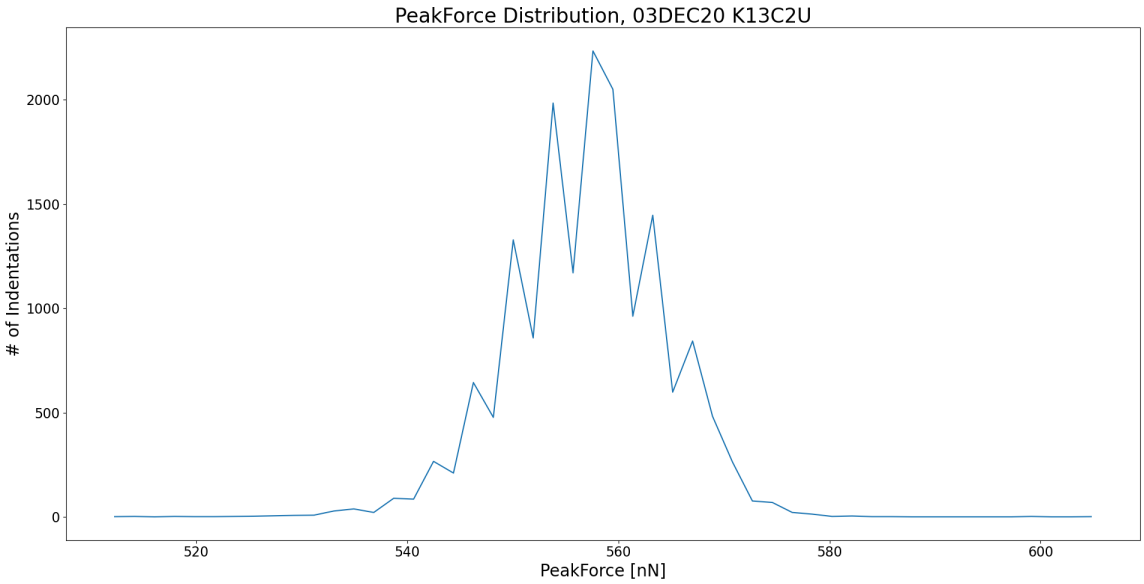

Figure 25. PeakForce Distribution, Excluding Indentation Depths ≥ 20nm.

Unlike the PeakForce distribution, the adhesion force distribution appears largely unchanged from Figure 27 to Figure 28. However, if compared with the data prior to removal of ≥20 nm indentations, the effect of removing outliers results in a reduction in the adhesion force range from 41.87 nN to 20.82 nN, a 50% reduction.

Finally, the indentation depth and tip radius distributions, Figures 29 & 31 respectively, transform in the same way, as expected since tip radius is solely a function of the indentation depth. The most extreme change is the removal of the high value tails resulting primarily from removing indentations ≥20 nm. The indentation depth range is decreased from 96.85 nm to 1.12 nm, a 98% decrease, while the tip radius range decreases from 7823 nm to 1.71 nm, a 99.97% decrease. These very large relative changes are due the presence of a small number of outlier indentation depths

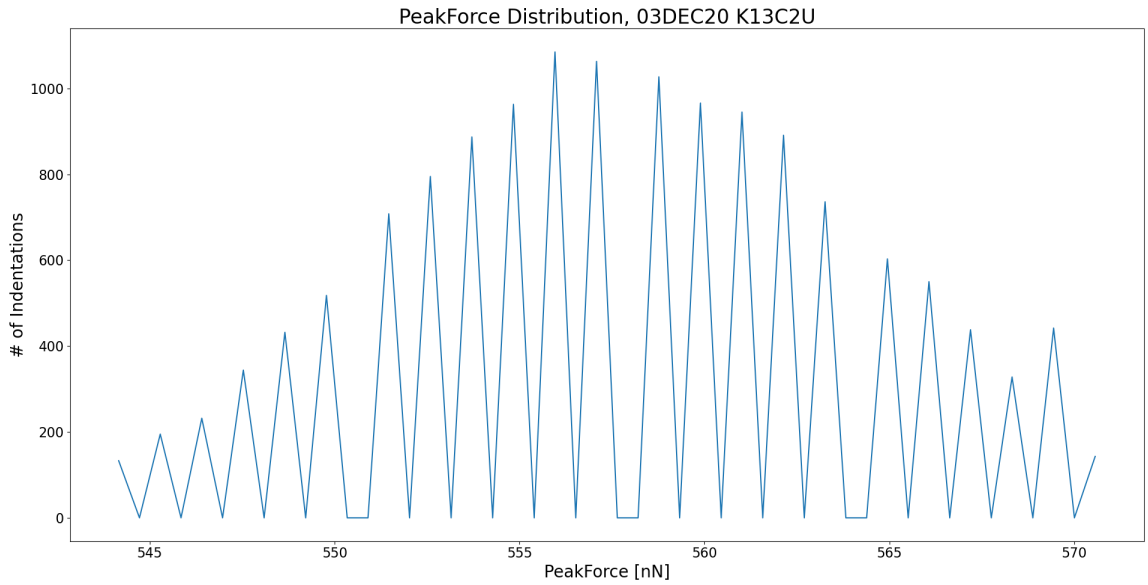

Figure 26. PeakForce Distribution Excluding Peak Force Values $\geq 2\sigma$ from the Mean.

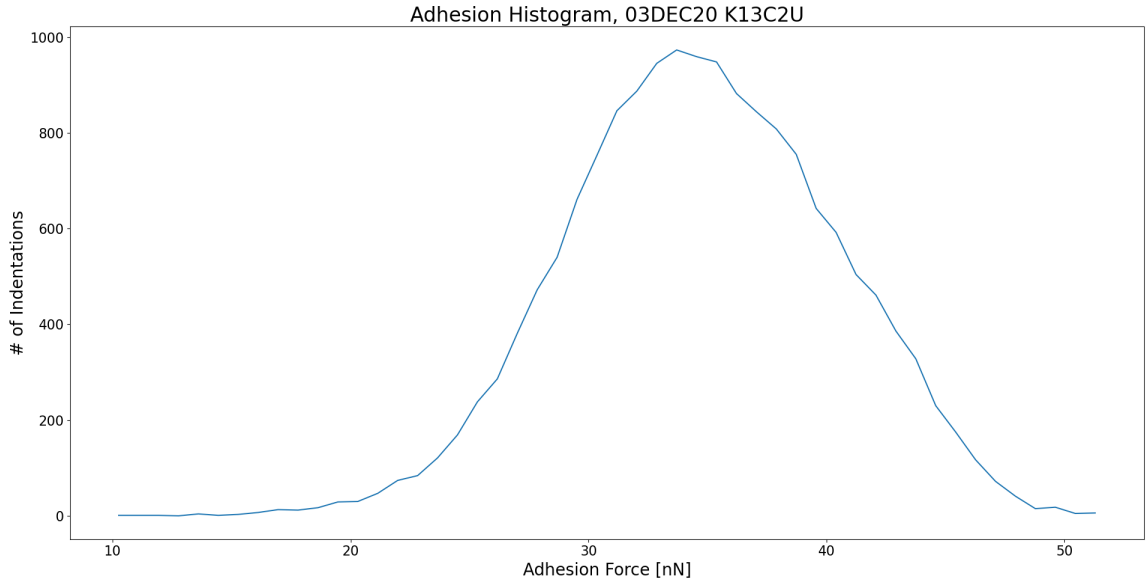

Figure 27. Adhesion Force Distribution, Excluding Indentation Depths $\geq$ 20nm.

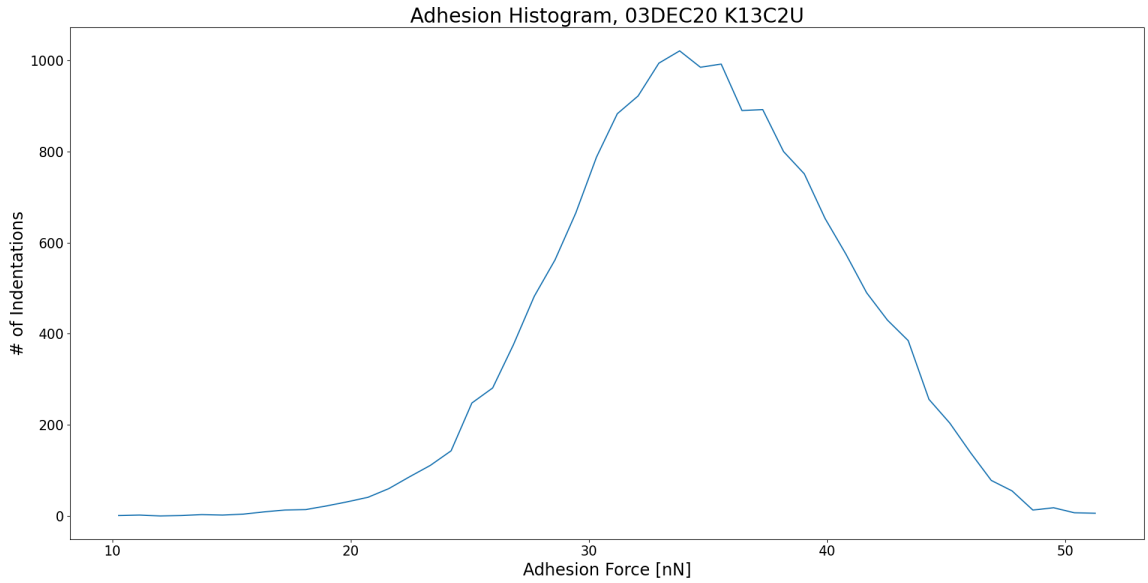

Figure 28. Adhesion Force Distribution Excluding Peak Force Values $\geq 2\sigma$ from the Mean.

which are removed when the 20nm indentation depth cutoff is applied.

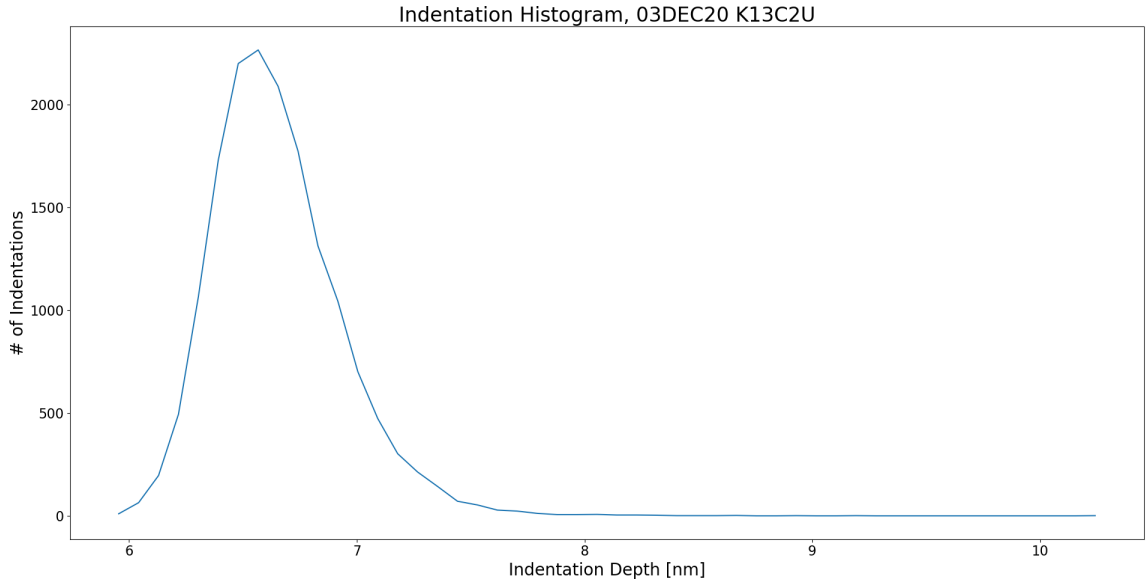

Figure 29. Indentation Depth Distribution, Excluding Indentation Depths $\geq$ 20nm.

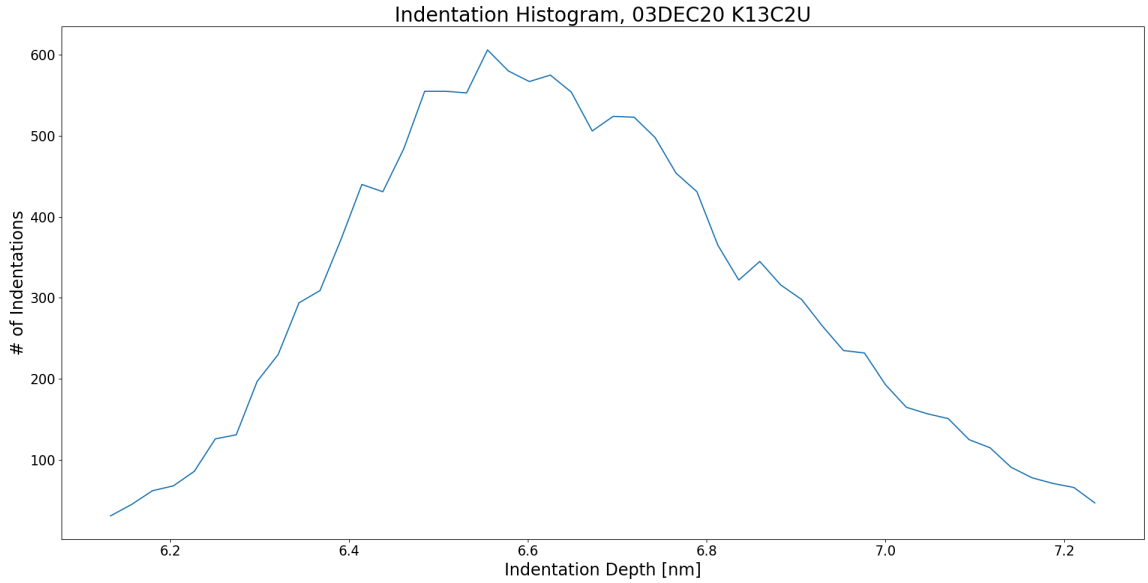

Figure 30. Indentation Depth Distribution Excluding Peak Force Values $\geq 2\sigma$ from the Mean.

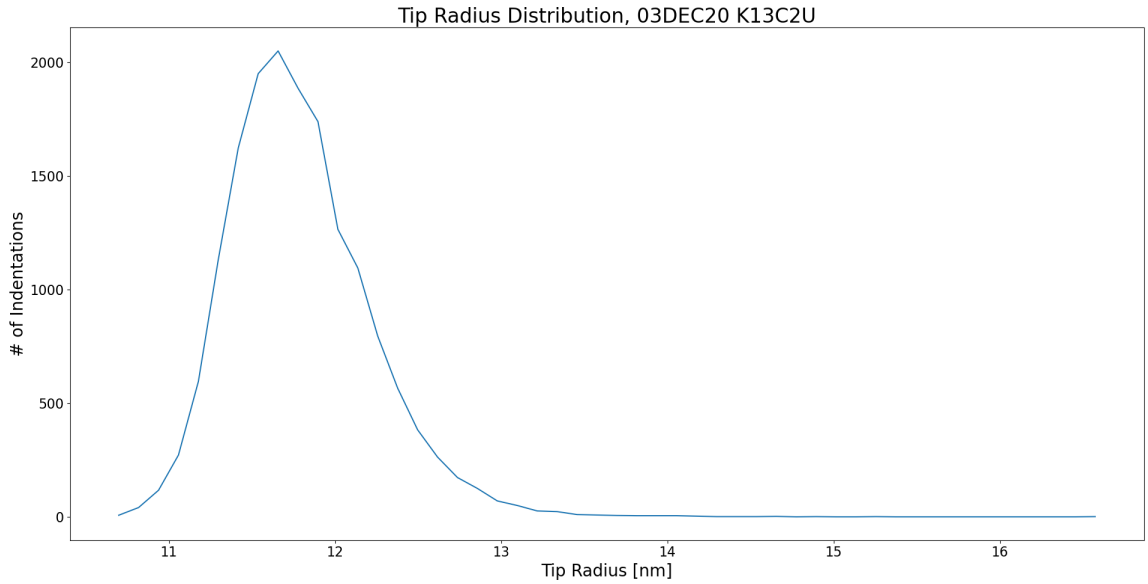

Figure 31. Tip Radius Distribution, Excluding Indentation Depths $\geq$ 20nm.

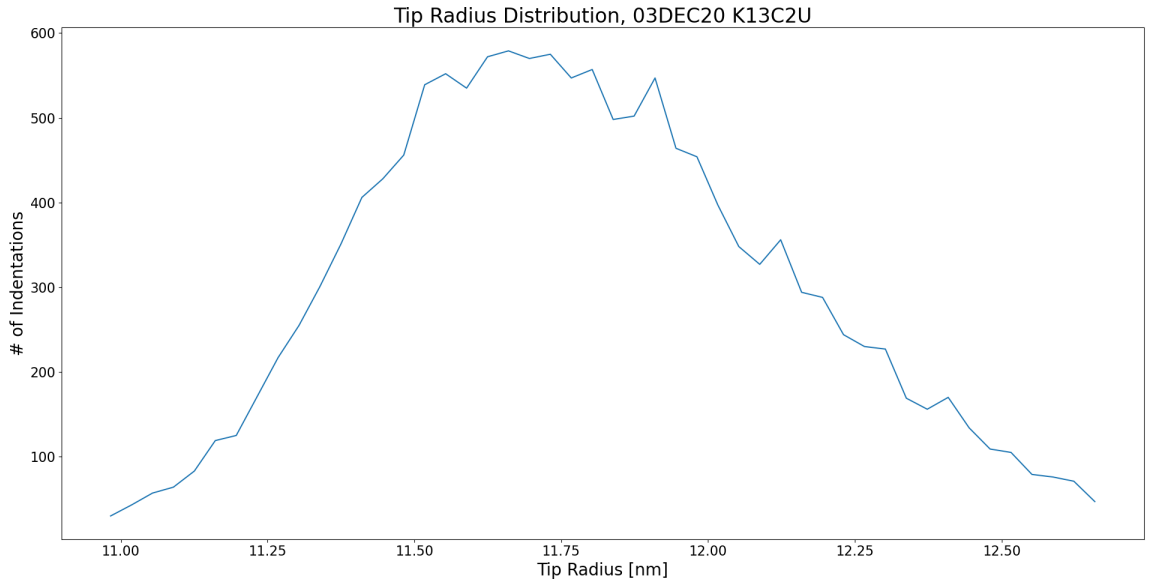

Figure 32. Tip Radius Distribution Excluding Peak Force Values $\geq 2\sigma$ from the Mean.

As far as how excluding these extreme values changes the fiber modulus measurement values, the modulus distribution for all data points is compared with those points where an extreme value in one of the other data channels occurs. Table 5 shows the number of measurements outside two standard deviations from the mean value in each data channel, and the percentage relative to the total number of measurements. As expected, the number of values outside of two standard deviations is relatively small, with no measurement exceeding the ideal Gaussian 5% value. Variation from this ideal value is expected, since none of the distributions are perfectly Gaussian. The location of the extreme values shown in Figure 33 in terms of the index location of the measurement point, shows that these values are distributed relatively uniformly across the entire image area. The adhesion force, however, shows a large increase in the number of outliers between indices 6000-7000, roughly the middle of the scan area. This increase of about 300 outliers corresponds with a scan area of about 17.5nm^2. Examining the actual adhesion force image produced by the AFM,

a dark spot is seen in the middle of the scan area, likely the source of this difference in the adhesion force outlier count.

Table 5. Number of Measurements Outside 2 Standard Deviations from Mean.

Data Channel	Number of Measure-ments	% of Total
Indentation	686	4.21
PeakForce	595	3.65
Adhesion	651	3.99

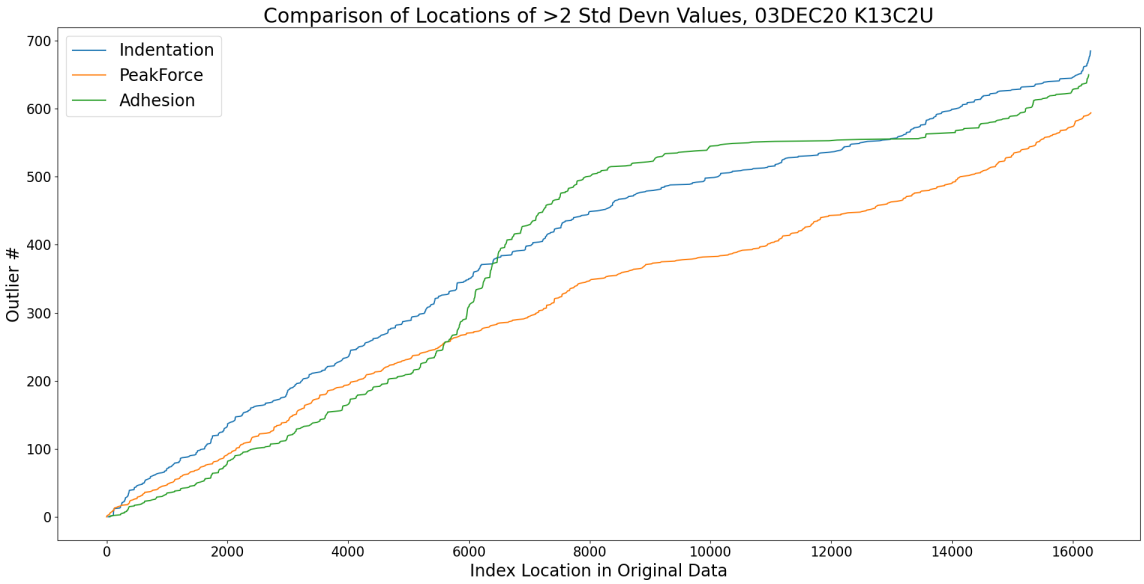

Figure 33. Location of Extreme Values by Measurement Index.

The modulus distribution for all measurements in comparison to the distributions due only to extreme values is shown in Figure 35. The modulus distribution for measurements due to extreme values in any data channels are all spread broadly across the distribution of all modulus measurements, indicating that any individual extreme value does not lead to deviation from the measured mean value. Further study was made for coincident extreme values, in which multiple extremes occur at

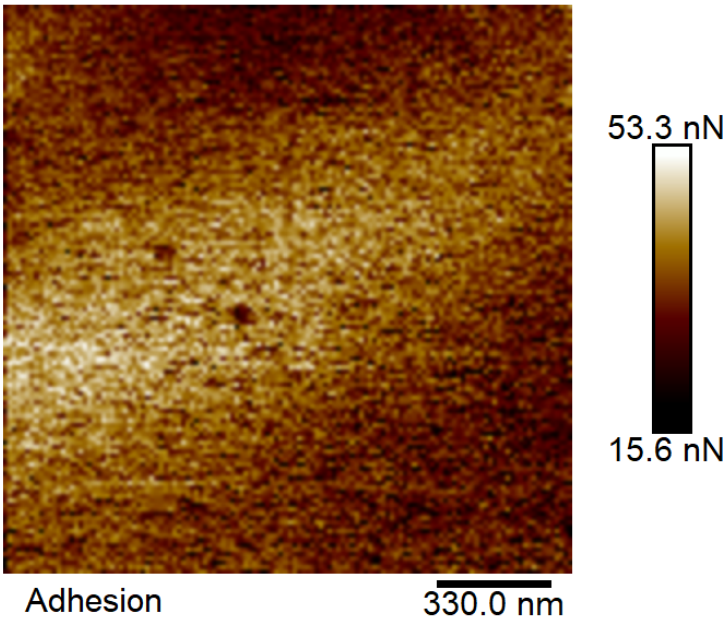

Figure 34. Adhesion Force AFM Image.

the same measurement location. Similar results were found, with the distributions centered near the mean for all measurements. Additionally, the number of coincident extreme values was very low, usually between 0.1-1%. The result of this study of the measurements at extreme values of indentation, adhesion force, and PeakForce demonstrates that these values do not contribute to shifting the mean, and so can be excluded from the image without changing the resulting modulus value. This method of extreme value exclusion was used in determination of the measured mean value of all images used to determine the sample means reported in Table 6.

4.2.2 Correlation Between Transverse and Longitudinal Fiber Modulus

The transverse modulus of each carbon fiber type was imaged using the PF-QNM method using a PeakForce Setpoint between 456.4-569.9 nN and the mean transverse

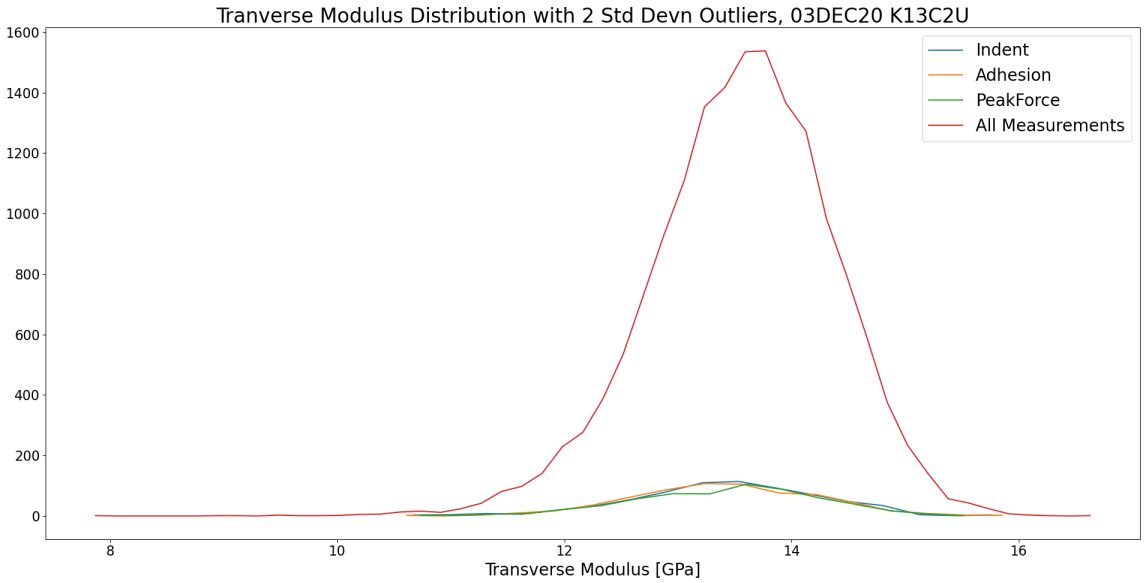

Figure 35. Transverse Modulus Measurements Due to Extreme Values.

modulus value of each was determined assuming a Gaussian distribution. The error weighted means and error weighted standard deviations were then calculated, after excluding any measurements that met the exclusion criterion described earlier. Figure 36 shows the error weighted mean transverse modulus for the 11 measured fibers plotted with their respective longitudinal modulus. The transverse modulus error is also shown for all fibers, calculated from the error weighted standard deviation. The Mitsubishi fibers, with longitudinal moduli between 641-924 GPa, also show the longitudinal modulus error calculated from the percent deviation of the certified modulus value from the nominal value. All other fibers were not provided with a certified modulus, and so the deviation of the longitudinal modulus from the nominal value is unknown. When linear fitting is performed, the Pitch-based fibers show a linear correlation with R^2 of 0.6334, while the PAN-based fibers show a linear correlation with R^2 of 0.0001. When all points are fit as a single data set, however, the overall linear correlation is R^2 of 0.7577. Table 6 shows the numerical results, as

well as the absolute and relative error.

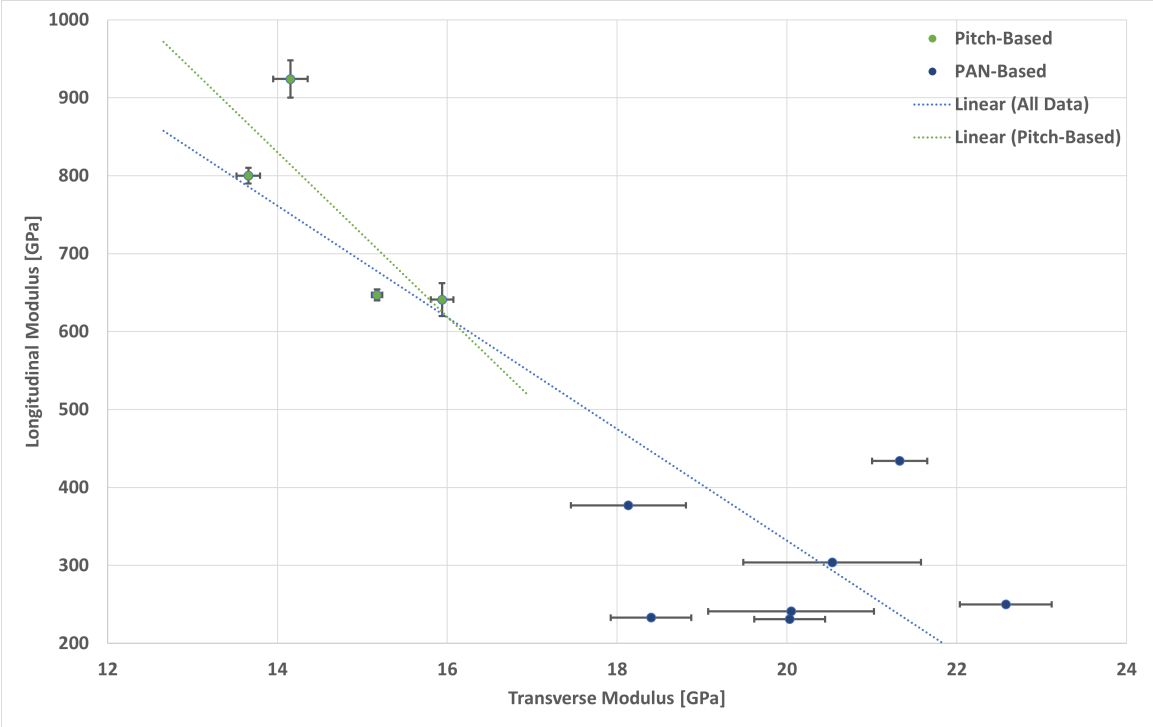

Figure 36. Measured Carbon Fiber Transverse Modulus.

Table 6. Carbon Fiber Transverse Moduli, Error Weighted.

Fiber	Manufacturer	Transverse Modulus [GPa]	Absolute Error [GPa]	Relative Error [%]
K13C2U	Mitsubishi	14.15	0.2031	1.44
K63A12	Mitsubishi	13.66	0.1385	1.01
K1352U	Mitsubishi	15.17	0.0621	0.41
K63712	Mitsubishi	15.94	0.1329	0.83
HM63	HEXCEL	21.32	0.3237	1.52
M40JB	Toray	18.13	0.6777	3.74
IM9/G-12k	HEXCEL	20.53	1.047	5.10
TRH50	Mitsubishi	22.58	0.541	2.40
AS4D	HEXCEL	20.05	0.9759	4.87
34-700WD	Mitsubishi	18.40	0.4740	2.58
AS4-GP-12K	HEXCEL	20.03	0.4168	2.08

4.2.3 Variation between PeakForce Modulus and Alternative Modulus

As stated earlier, the modulus values resulting from calculation are not expected to be the same as those reported by the AFM. This variation is due to the the points on the force curve selected for calculation. The Bruker AFM software automatically selects cutoff points for maximum and minimum fit regions based on user-defined setpoints, which cannot be easily replicated with the exported image data. Certain points on the force curve, however, are defined in terms of the measured tip force. Specifically, the PeakForce is the absolute maximum, the adhesion force is the absolute minimum, and the indentation depth is where force equals zero between the PeakForce and adhesion force values. Figure 37 shows how fitting between different points on the force curve will change the calculated value. For this fiber, the reduced modulus calculated between the PeakForce and adhesion force points gives a value 50.1% less than the AFM measured modulus, while the reduced modulus calculated between the PeakForce and indentation depth points give a value 33.5% larger than the measured modulus. From this analysis it can be concluded that the instrument DMT fitting routine includes the contribution of both the linear indentation and non-linear adhesion region.

For most images, the modulus calculation excluding the adhesion region produces a narrower distribution and thus a more precise value, while the calculation including the adhesion region is usually less precise. Both methods were found to be less useful as a characteristic measure for a specific fiber type, as can be seen in Figures 38 and 39. For either method, the trend found for the instrument-reported modulus no longer holds, with calculated mean modulus values for low longitudinal modulus fibers showing lower transverse modulus than that of very high modulus fibers, a result that is unlikely given the physical understanding of fiber microstructure and its contribution to the longitudinal and transverse mechanical properties. Based on

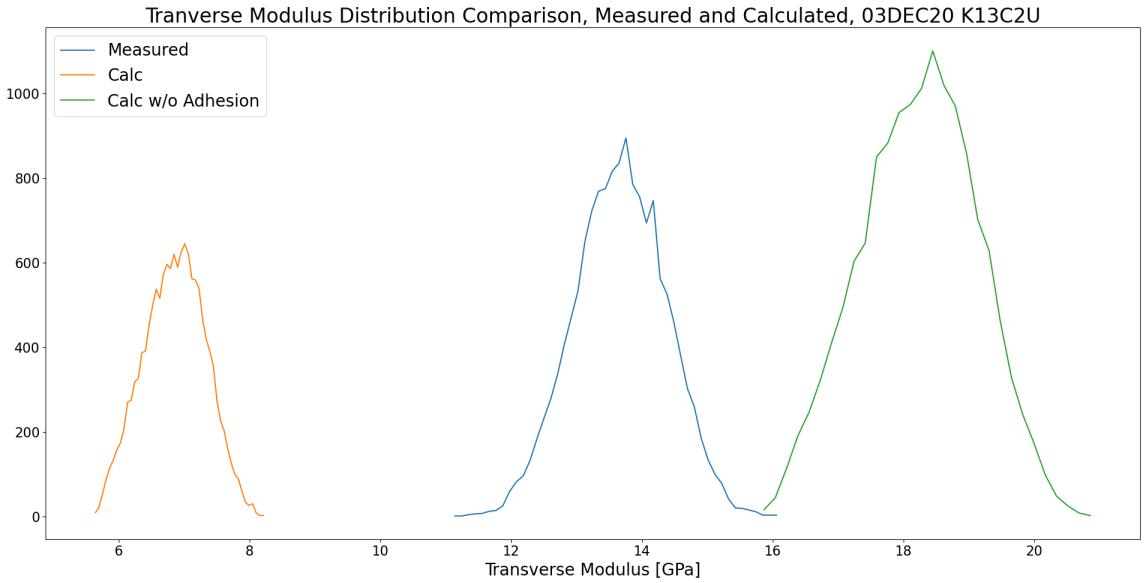

Figure 37. Comparison of Calculated and Measured Fiber Transverse Modulus.

this result, post-processing methods of modulus calculation were not found to be useful for comparing fiber types, though these methods were used to conduct further studies of the influence of other material parameters such as the Poisson Ratio and tip hardness assumption in the DMT model.

4.3 Error Contributions from Material Assumptions

The results shown in Figures 36, 39, and 40 are determined directly from statistical analysis of the instrument raw data, without attempting to deconvolve the AFM tip and fiber sample material properties. These methods can be applied through calculation with the AFM measurement data using the post-processing script described earlier using the DMT model in Equation 6. This modulus calculation assumes that the AFM tip's Young modulus is infinitely hard, or at least much harder than the sample. This assumption allows the reduced fiber Young modulus to be treated as the true fiber modulus. In the case of the AFM tip and the carbon fiber, the differ-

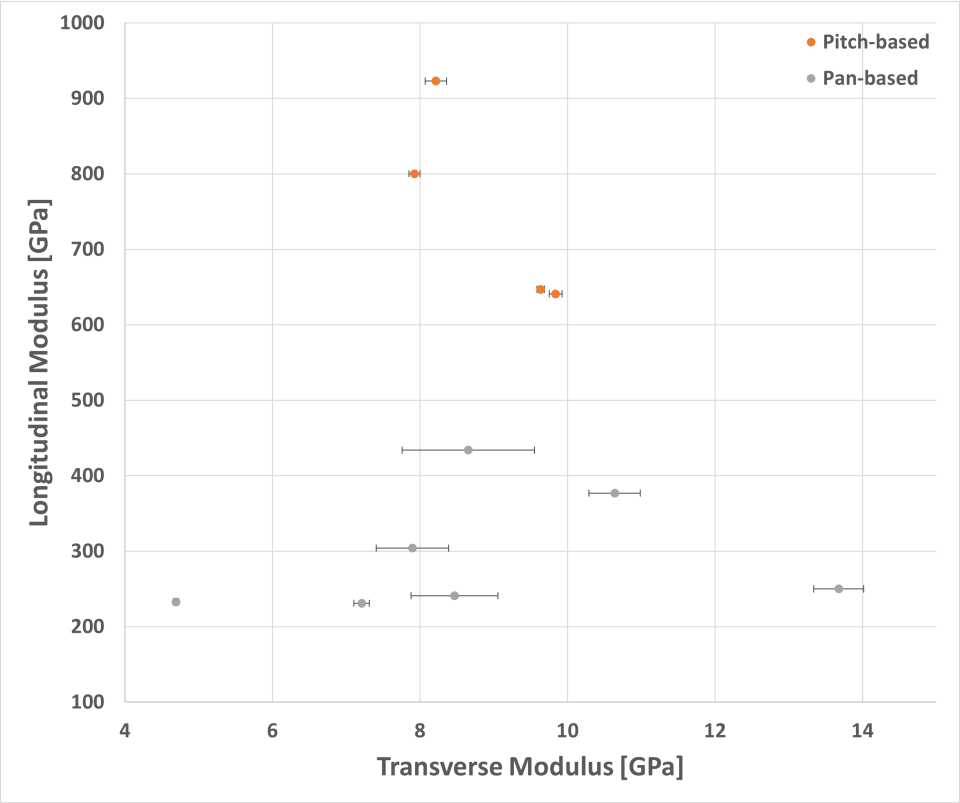

Figure 38. Mean Calculated Relative Transverse Modulus, Eleven Fibers.

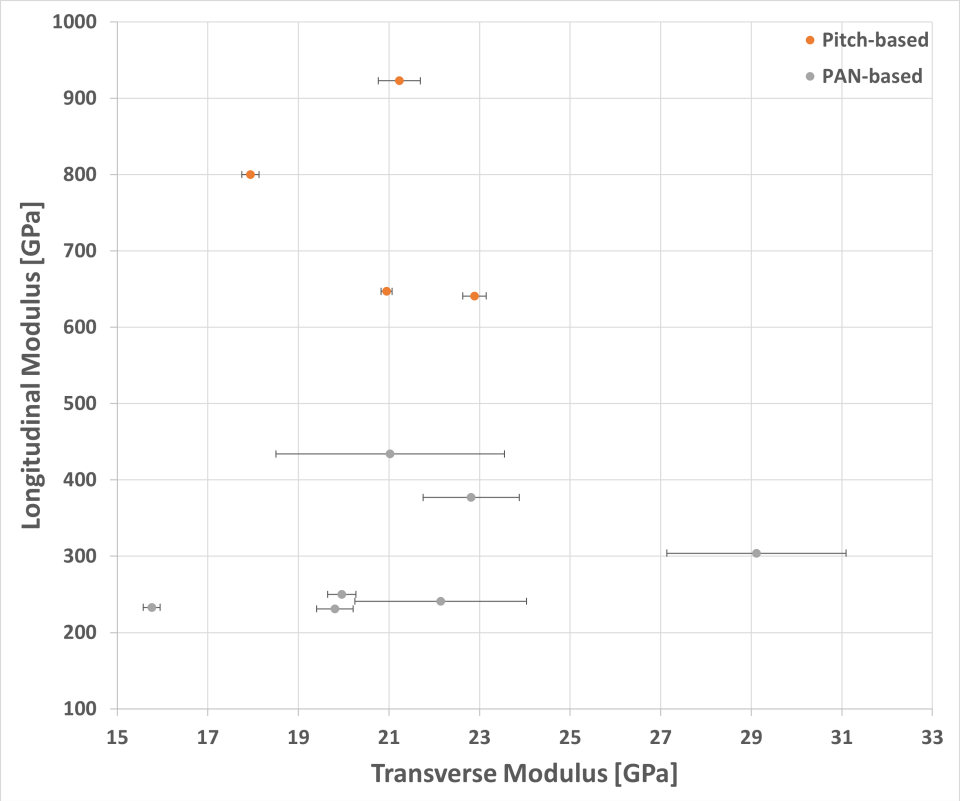

Figure 39. Mean Calculated Relative Transverse Modulus without Adhesion, Eleven Fibers.

ence in modulus is only about one order of magnitude, meaning the reduced Young modulus is likely not appropriate. Additionally, since the maximum and minimum forces associated with the force curve are characterized by large variations over a single image, it is expected that excluding these high error regions will improve the overall modulus precision.

The modulus values resulting from this direct calculation are not expected to be the same as those determined by the AFM. This is due to the design of the DMT modulus fitting routine, which specifically excludes portions of the force curve around the force maximum and minimum regions. By using the data measured directly from the tip-sample interaction, however, trends can be determined for how uncertainty in different material properties, such as the fiber Poisson Ratio, and uncertainty due to the assumptions required by the DMT model leads to variation from an ideal "true" modulus measurement.

4.3.1 Variation Due to Tip Hardness Assumption

A study to compare the modulus to the modulus using the assumption that the tip is not "infinitely hard". This study, conducted using the calculated modulus fit between the PeakForce and adhesion force, provides a error bound for using the assumption, with the error being the deviation from the "true" fiber modulus. Figure 40 shows a small variation if the AFM tip is not assumed to be infinitely hard. For this measurement, this variation was a 3.78% increase in the calculated modulus. For other fibers the variation is similar, ranging from 1.32%-12.52%, with most calculated values varying by less than 5% and within one standard deviation of the calculated modulus mean. This result supports the conclusion that, while the tip hardness assumption does result in a different modulus value, the difference is usually within normal statistical variation for the measurement method.

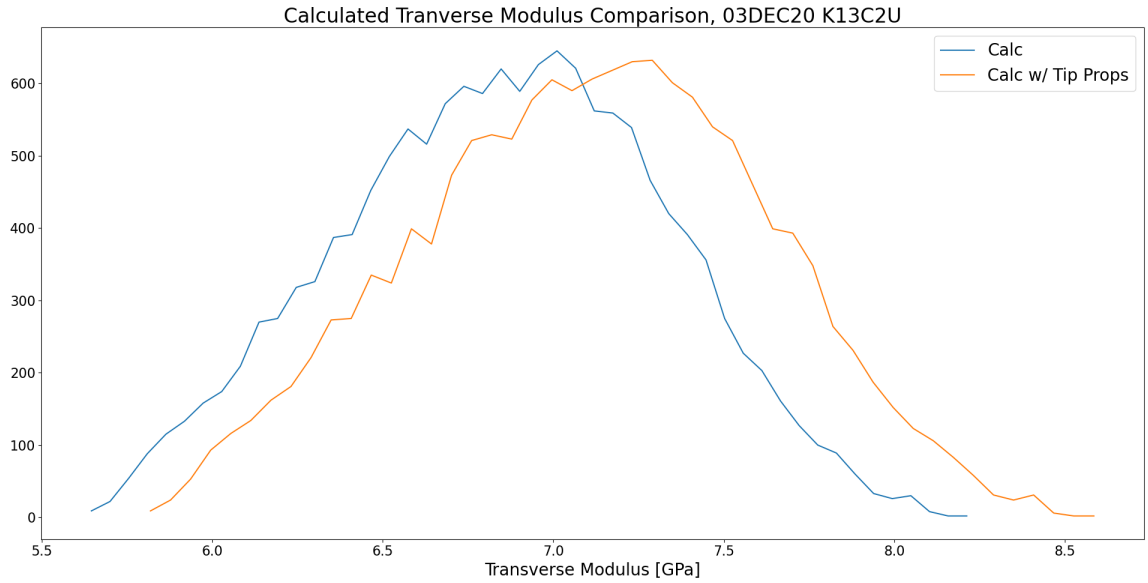

Figure 40. Comparison of Calculated Fiber Transverse Modulus.

4.3.2 Modulus Variation due to Uncertainty in Poisson Ratio

The second study compares the effect of varying the fiber Poisson Ratio. This study, again conducted using the calculated modulus fit between the PeakForce and adhesion force, is necessary since the Poisson Ratio for carbon fibers is not well characterized, and is generally assumed to be 0.3. For this study, the Poisson Ratio was varied between 0.2 and 0.4 to determine the relative change in the calculated modulus. Figure 41 shows this variation, with ν=0.3 selected as the nominal value to be varied about. For this measurement, the variation in modulus relative to the nominal case mean is 13.19%, slightly larger than twice the measurement standard deviation. For ν=0.2 the relative variation is -5.49%, while ν=0.4 gives a 7.69% variation. This variation trend holds across all fibers and measurements in this study, indicating that this variation is solely due to the numerical value of the Poisson ratio, and that uncertainty in this value does not influence the precision of the measured modulus as long as the ratio is held constant. The accuracy of the modulus measurement to the

"true" modulus, however, requires precise knowledge of the Poisson ratio.

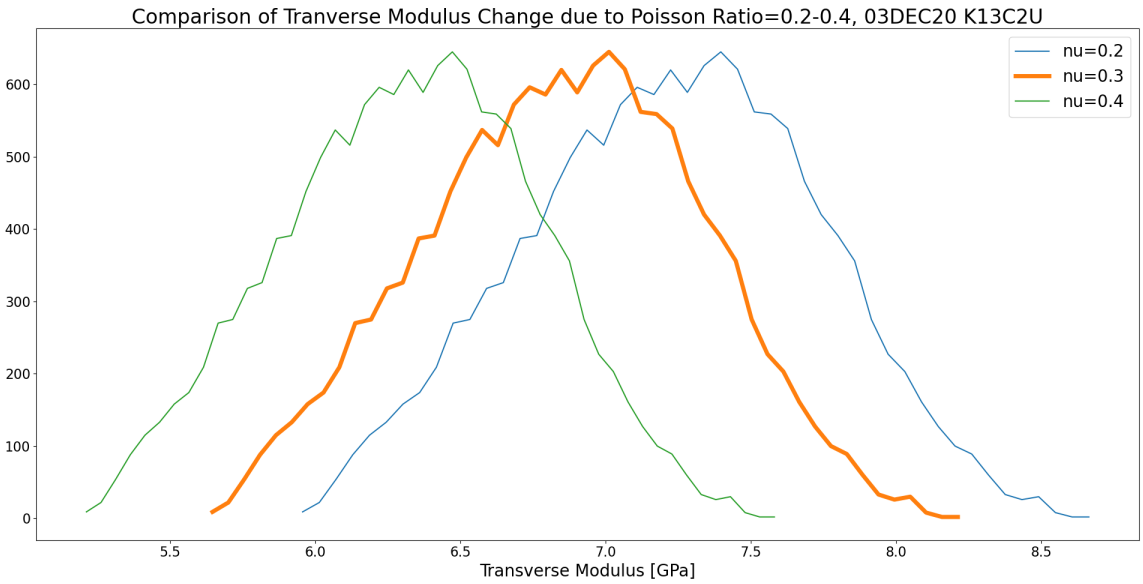

Figure 41. Change in Fiber Transverse Modulus due to Poisson Ratio 0.2-0.4.

4.4 Modulus Calculation Directly from Force Curves

The Bruker AFM software calculates the DMT modulus of the sample by applying the directly measured force curve at each indentation and applying a linear fit algorithm to some region between the PeakForce maximum and adhesion minimum. The specific method used is proprietary, however the data necessary to perform the same calculations can be exported via the NanoScope Analysis software, and so it should be possible to perform the same calculation via a post-processing script.

4.4.1 Preparation of Instrument Output Files

The method of directly accessing and manipulating the raw indentation force curves requires a multi-step procedure conducted within the Bruker NanoScope Analysis software. This method is limited by the capabilities of the software itself to han-

dle a large number of files and the ability for the user to manipulate them. While potentially a powerful source of more data, the limitations of the software proved challenging in developing a feasible implementation of the method. The software is limited to viewing and batch exporting 2000 files at once. While this number appears relatively large, a typical 128x128 image produces 128 .hsdc High-Speed Data Capture files, one for each scan line in the image. Each of these .hsdc files contains 3000 force curves, providing potentially 384,000 force curves for analysis. In practice the number of force curves exported per line was limited to 50, since the Bruker software limits the number of exported curves to 50. Finally, each of these exported force curve files must be reimported into NanoScope Analysis and the force curve exported as a .txt file that is human readable for use by the post-processing script. It was found that the method of file preparation to produce human readable force curve files takes approximately one hour per image of direct interaction with the software, in addition to computational time. This is contrasted with a processing time of about 60 seconds for the methods described earlier.

4.4.2 Modulus Distributions from Force Curves

The method of determining the transverse modulus via direct calculation for the complete fiber measurement set was deemed infeasible due to the large amount of time required for the analysis. Additionally, as will be shown below, the modulus calculated via this method deviates from the expected value and from the instrument-reported value, indicating that the method applied to determine the modulus may be flawed.

For the AS4-GP-12K fiber, the modulus distribution calculated from the force curves was determined and the mean value was found. Figure 42 compares the modulus distribution determined directly from the force curves, with a mean modulus of 1.014 GPa and a relative error of 84%, to the distribution determined from the

instrument reported measurement, with a mean modulus of 13.64 GPa and a relative error of 5.87%. In addition to the clear difference in the modulus and precision, the method determined via the force curve fitting script also produces a bimodal distribution, with a very large number of measurements with a near zero modulus value. Similar findings, in which the modulus found via directly fitting the force curves was much less than that of the instrument reported values, with a much larger error and dissimilar modulus distribution, were also found for other fiber measurements.

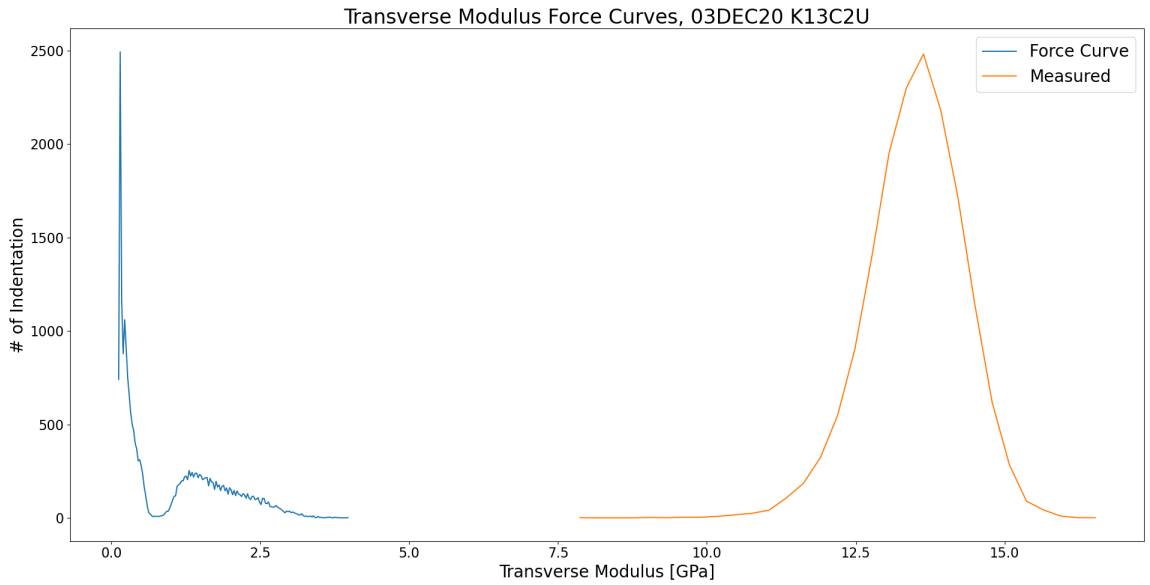

Figure 42. Comparison of Modulus Distributions Calculated from Force Curves and Reported by AFM.

4.5 Correlations Between Transverse Modulus Error and Other Measurement Errors

As described earlier by Equation 7, a theoretical minimum expected error in the modulus can be calculated from the error of the associated PeakForce, adhesion force, indentation depth, and tip radius. For any measured set of these values, the minimum

expected error sets a bound which cannot be improved upon. This minimum error bound, however, is not necessarily achievable since the DMT model used to calculate the modulus does not account for any multivariate relationships between these different measured values. For example, the error in the indentation depth and tip radius should be correlated since the tip shape is fixed, and thus the tip radius is determined by the indentation depth and the fixed tip shape. The tip also experiences minute wear while imaging, which would be expected to lead to a change in the indentation depth and the adhesion force, since an increased interaction surface between the tip and the sample is expected to lead to increased adhesive interaction.

4.5.1 Multivariate Tip-Sample Relationships

In order to study whether any of the measured tip-sample interactions behave in a multivariate manner, the distributions for the PeakForce, adhesion force, and indentation depth data, as well as the measured DMT modulus, were plotted as 2-D histograms, shown in Figures 43-52. If the data are univariate they would be expected to be centered on the heat map, with Gaussian behavior for each data distribution. If the data are multivariate, however, the 2-D histogram would show some other behavior. The 2-D histograms shown below exclude values outside one standard deviation from the mean in order to improve the contrast of the plot, however they are indicative of the relationships between the measurements when all data points are included. The striations shown on the histograms of Peak Force are due to the feedback loop iteration performed by the instrument, as seen in Figure 26.

The 2D histograms of indentation depth in Figures 43-44 show both expected behavior and unexpected behavior. The relationship between indentation depth and PeakForce shows an even distribution over both ranges, indicating that these measurements are univariate. The relationship between indentation depth and tip radius

is, as expected, strongly bivariate, since the tip radius is itself a function of the indentation depth. Because of this, any variation in the tip radius is expected to be strongly correlated with the variation in indentation depth. The relationship between indentation depth and adhesion force, however, appears only weakly bivariate, with decreasing adhesion force correlated with increasing indentation depth. This result is not expected, since a larger indentation depth should lead to increased interaction area between the tip and the sample, and so a larger area over which attractive forces can act.

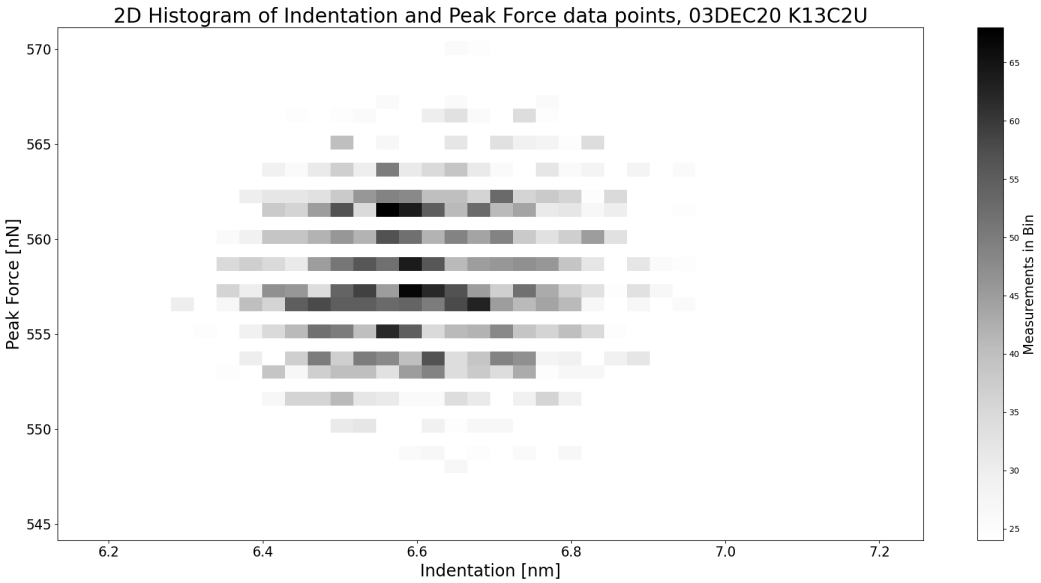

Figure 43. 2D Histogram of Indentation Depth and PeakForce.

The 2D histograms of PeakForce in Figures 46-47 show results that are expected. The relationship between PeakForce and Adhesion force appears univariate, which is expected since the adhesive forces and the applied tip force are caused by completely different physical phenomena, that of short-range attractive forces and mechanical indentation, respectively. The relationship between PeakForce and Tip Radius is similar to that between Indentation Depth and PeakForce, which is expected since

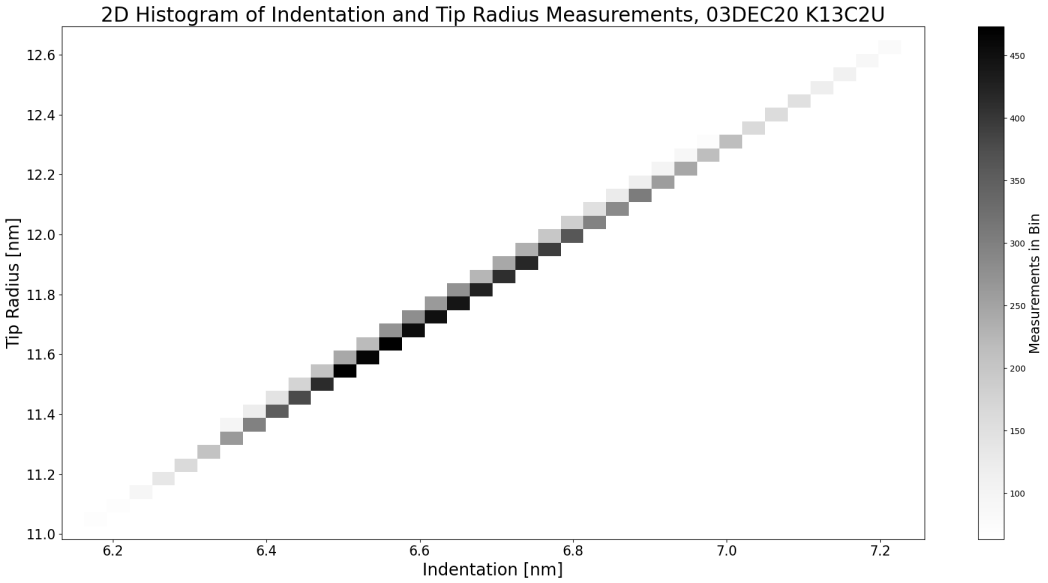

Figure 44. 2D Histograms of Indentation Depth and Tip Radius.

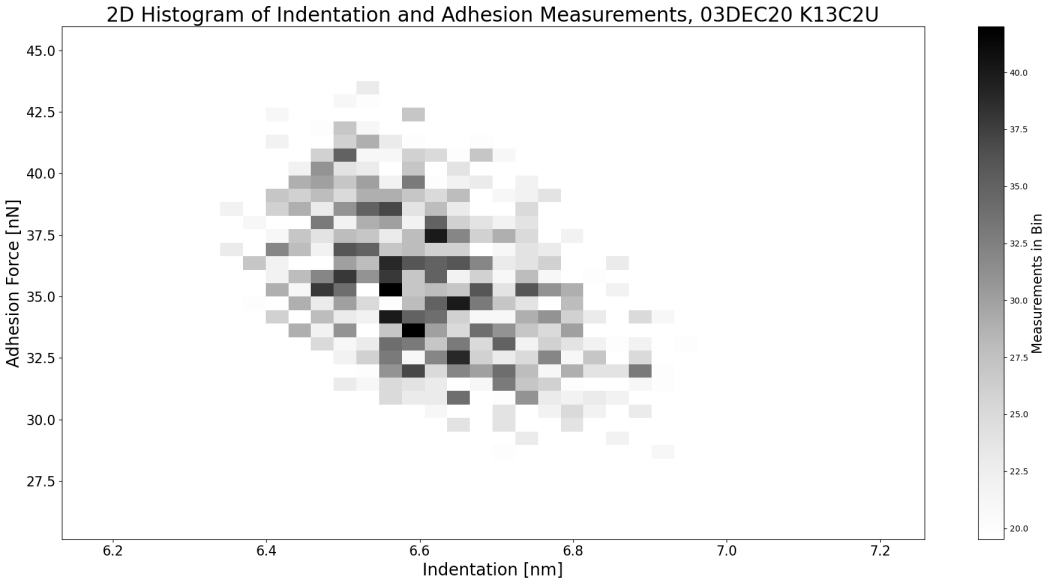

Figure 45. 2D Histogram of Indentation Depth and Adhesion.

indentation depth and tip radius are strongly correlated. Finally, the 2D histogram of adhesion force and tip radius in Figure 48 is again expected from the indentation and adhesion force results.

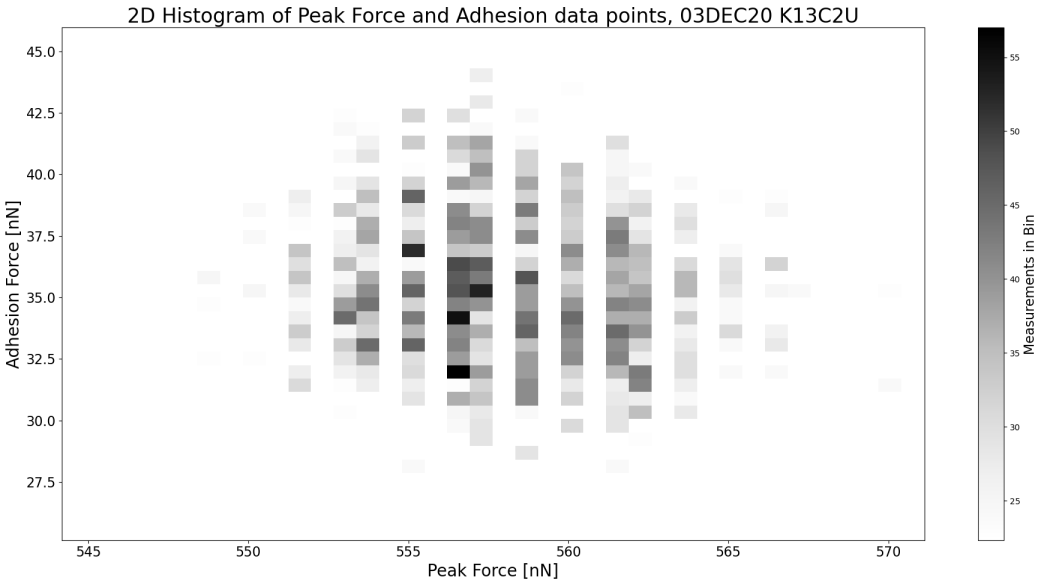

Figure 46. 2D Histogram of Peak Force and Adhesion.

The 2D histograms of the DMT modulus and PeakForce, adhesion, indentation, and tip radius shown in Figures 49-52 clearly show relationships between the measured modulus and variation in the other measured tip-sample interactions. PeakForce and DMT modulus appear univariate, which is expected since the use of the PeakForce as the feedback loop parameter implies that its variation should not influence the measured value. On the other hand, adhesion force demonstrates a weak bivariate relationship with DMT modulus, with lower modulus values correlated with lower adhesion force values. This is expected since the adhesion region, as defined in the measurement force curve, is highly non-linear. This non-linear behavior leads to variation in the lower bound for DMT modulus fitting, while a smaller adhesion force leads to a shallower slope in the force curve and thus a lower DMT modulus fit value.

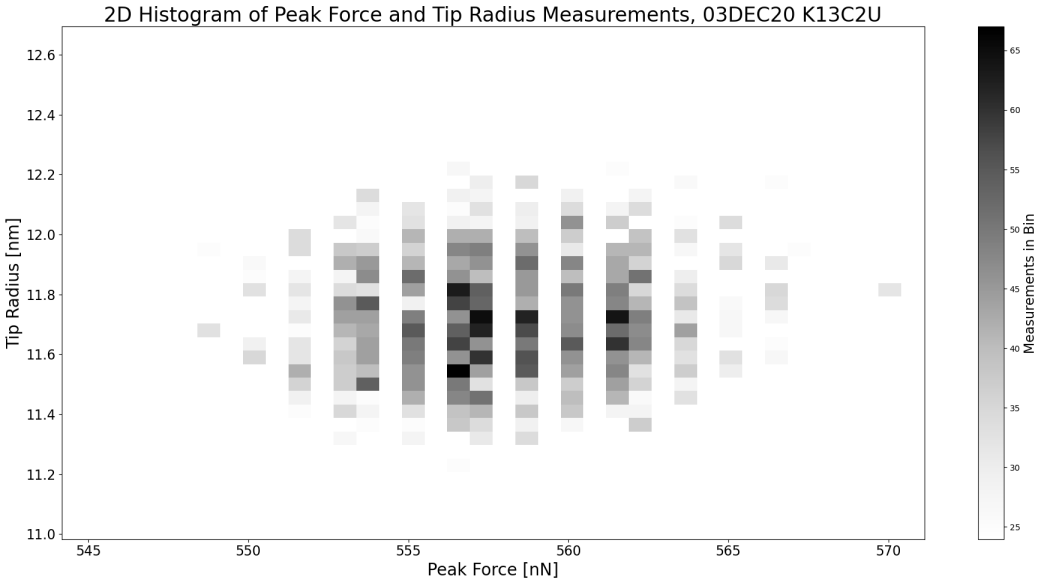

Figure 47. 2D Histogram of Peak Force and Tip Radius.

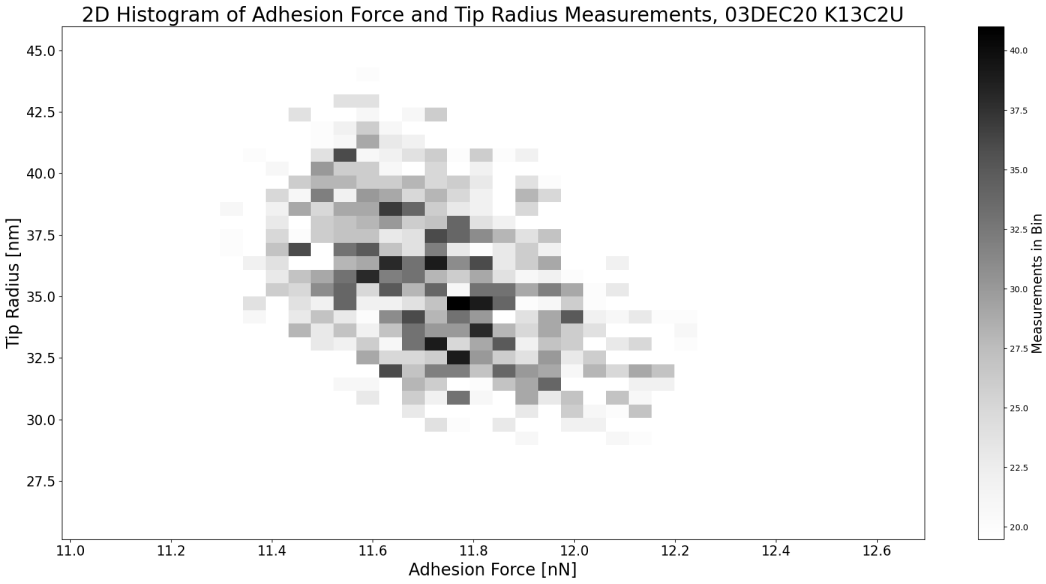

Figure 48. 2D Histogram of Adhesion Force and Tip Radius.

Finally, indentation depth is very strongly correlated to the DMT modulus, with larger modulus values associated with shallower indentation. This is expected based on the mechanical model of indentation, with a harder material being less susceptible to deformation by the indentor tip. The correlation with tip radius is similar, as expected from the relationship between indentation and tip radius described earlier.

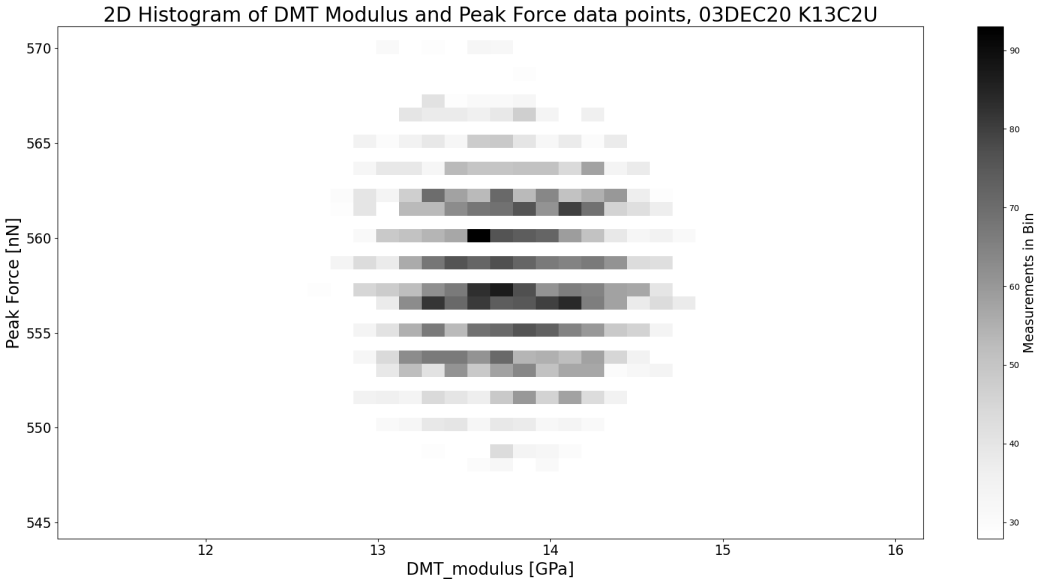

Figure 49. 2D Histogram of Transverse Modulus and Peak Force.

The bivariate relationships between the modulus and the adhesion force, indentation depth, and tip radius indicate that these channels are likely good candidates for error reduction that would produce a correlated reduction in the modulus error.

4.5.2 Correlations Between Transverse Modulus Error and Other Errors

Based on the results above, correlation testing was performed to determine if the measurement error of the modulus is in fact correlated with other measurement errors, and if so how strongly. Two correlation tests were performed, the Pearson

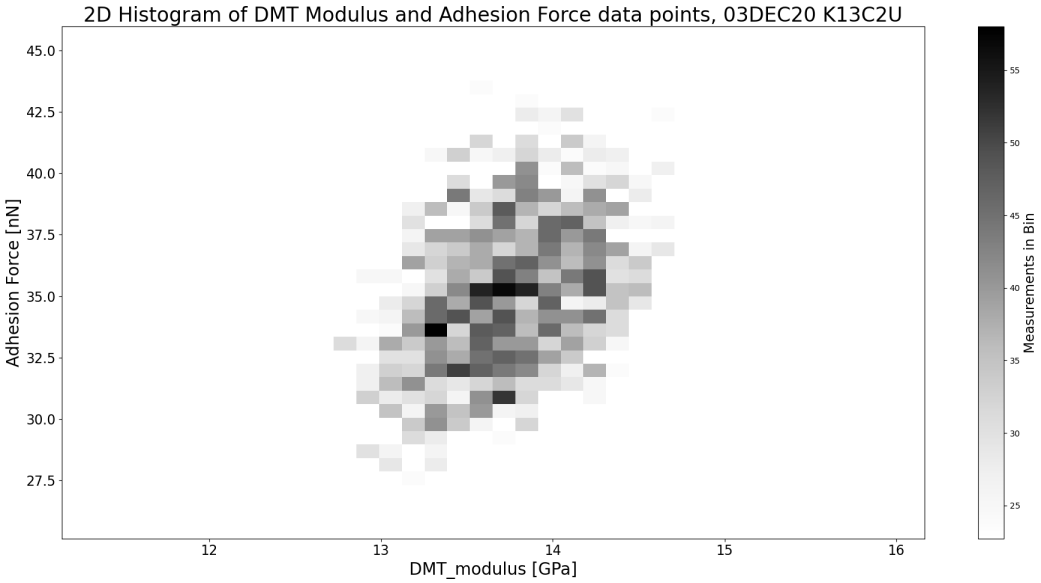

Figure 50. 2D Histogram of Transverse Modulus and Adhesion Force.

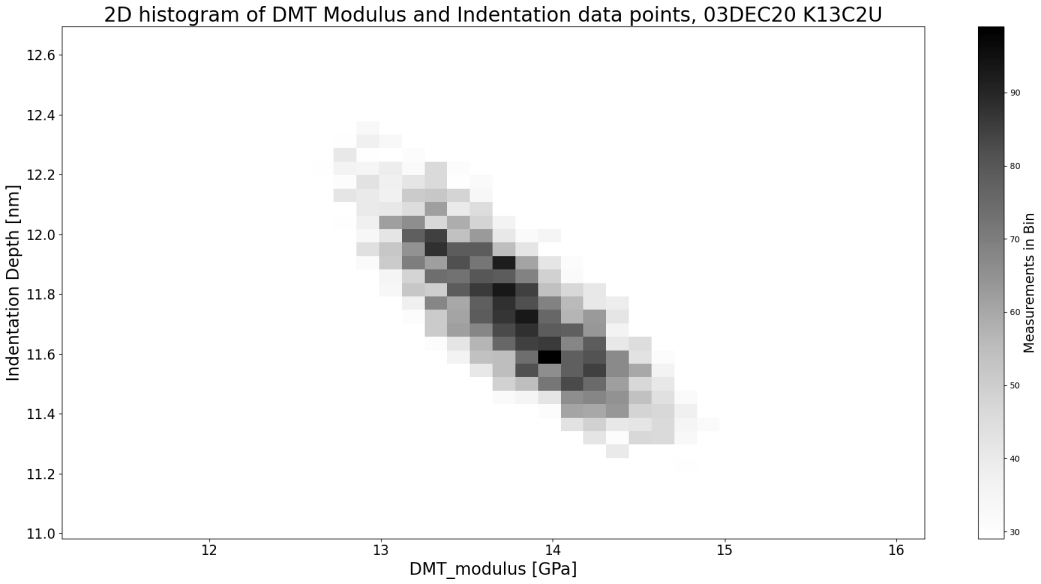

Figure 51. 2D Histogram of Transverse Modulus and Indentation Depth.

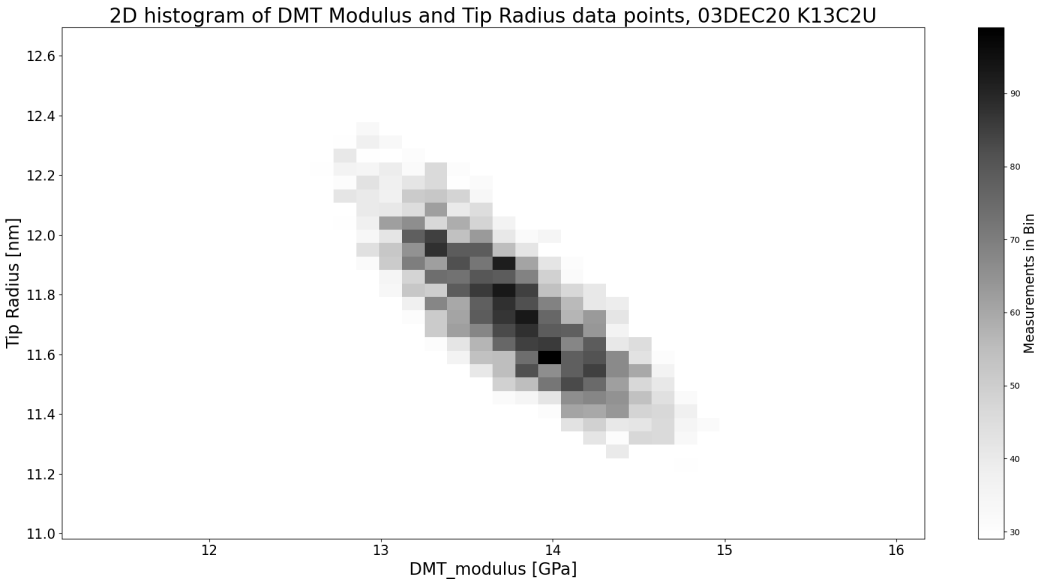

Figure 52. 2D Histogram of Transverse Modulus and Tip Radius.

R and the Spearman R. The Pearson R is a simple test for linear correlation, with the assumption that the samples are independent, normally distributed, and have the same variance. The Spearman R is a test for rank correlation, testing whether the samples are correlated by a monotonic function, not necessarily linearly. The Spearman R test requires only that the samples can be rank ordered and that they are independent and identically distributed.

The results of each correlation test are shown below in Table 7. From these tests, it was found that the proposed theoretical modulus error described earlier in Equation 7 is strongly correlated with the measured modulus. Surprisingly, accounting for a nominal variation ($\nu_{mean} = 0.3$, $\sigma_{nu} = 0.06$) in Poisson's ratio weakens the correlation between the theoretical error and the measured error in the modulus. As expected from analysis of the multivariate histogram between the modulus and the other data channels, the correlation between the modulus and the peak force is poor, while for the Spearman R test the correlation between the modulus and both the indentation

depth and adhesion force is strong.

Table 7. Correlation Testing Between Modulus Error and Other Errors.

Error Type	Pearson R	Spearman R
Theoretical	0.69	0.70
Theoretical w/ ν	0.50	0.62
Indentation	0.44	0.76
Adhesion	0.55	0.68
Peak Force	0.26	0.36

4.6 Summary

This chapter described the results of measuring the transverse modulus for single strand carbon fibers, finding a linear correlation between the longitudinal and transverse modulus with an R^2 of 0.7577 for the entire sample set. The correlation is weaker for the Pitch-based fibers, with an R^2 of 0.6334, and non-existent for PAN fibers. The influence of outlier values in indentation depth, peak force, and adhesion force on the measured modulus distribution is shown, demonstrating a small influence on the mean modulus value and a decrease in the standard deviation. Results for the three alternative means of directly calculating the modulus using the other AFM measurement data are also described, finding significant deviation from the instrument values and no correlation with the longitudinal modulus. Additionally, the technique using the measurement force curves is found to be exceptionally tedious and impossible to automate completely, preventing its application to the complete data set. Results are also described for the intrinsic modulus variation due to the tip hardness approximation and due to uncertainty in the Poisson ratio for specific carbon fiber types. The variation in the modulus resulting from changing the Poisson ratio may be an unaccounted source of the measurement error in the modulus due to difference in the fiber microstructure from point to point. Finally, correlations

between the measured modulus error and the measured errors in indentation depth, peak force, and adhesion force, finding that the theoretical modulus error derived earlier is correlated with the true measured modulus error, and that the correlations between the modulus error and both indentation depth and adhesion force are much stronger than the correlation with the peak force. This correlation indicates that optimizing the indentation depth and adhesion force measurement to minimize their variation in an image should lead to the strongest reduction in the measurement error for the fiber modulus.

V. Analysis

5.1 Overview

This section provides analysis and describes the significance of the results to supporting the research objectives. Additionally, it explains the assumptions that gave rise to deviation from the expected values in the calculated modulus methods.

5.2 Measurement of Fiber Transverse Modulus

Prior study by Veigas demonstrated a linear correlation between increased longitudinal fiber modulus and decreased transverse modulus.[3] The current research strengthened the asserted correlation and extended it to Pitch-based carbon fibers. Calculated modulus error was reduced via a combination of statistical techniques and an increased number of fiber measurements.

The fiber modulus correlation is strengthened by including a larger number of fibers across a broad longitudinal modulus range, even if the correlation appears weak or non-existent within one portion of the sample set. Specifically, the correlation in the PAN-based fibers was very poor, however their introduction into the sample set with the Pitch-based fibers improved the correlation of both subsets. It is unclear from this study why the PAN fibers showed such a poor correlation, contradicting the results of Veigas for the same fibers.

Two fibers previously studied by Veigas, 34-700WD, and AS4-GP-12K, showed large deviations below the expected values, while a new fiber, HM63 show a deviation above the expected value. A possible explanation for the deviation of the 34-700WD and AS4-GP-12K fiber is a difference in the Peak Force used. These fibers were measured early in the study using the largest Peak Force values in this study, with an average Peak Force value of 1799 nN, about twice the average of all other fibers

of 664 nN. The transverse moduli measured for the 34-700WD and AS4-GP-12K fibers were closer to the values predicted from the correlation determined by Veigas, indicating that the modulus measurement is dependent on the Peak Force Setpoint used to produce the image. When these fibers are excluded, the strength of the PAN fiber correlation improves to an R^2 of 0.5961, and the correlation for the complete sample set improve to an R^2 of 0.8884. This is shown in Figure 53.

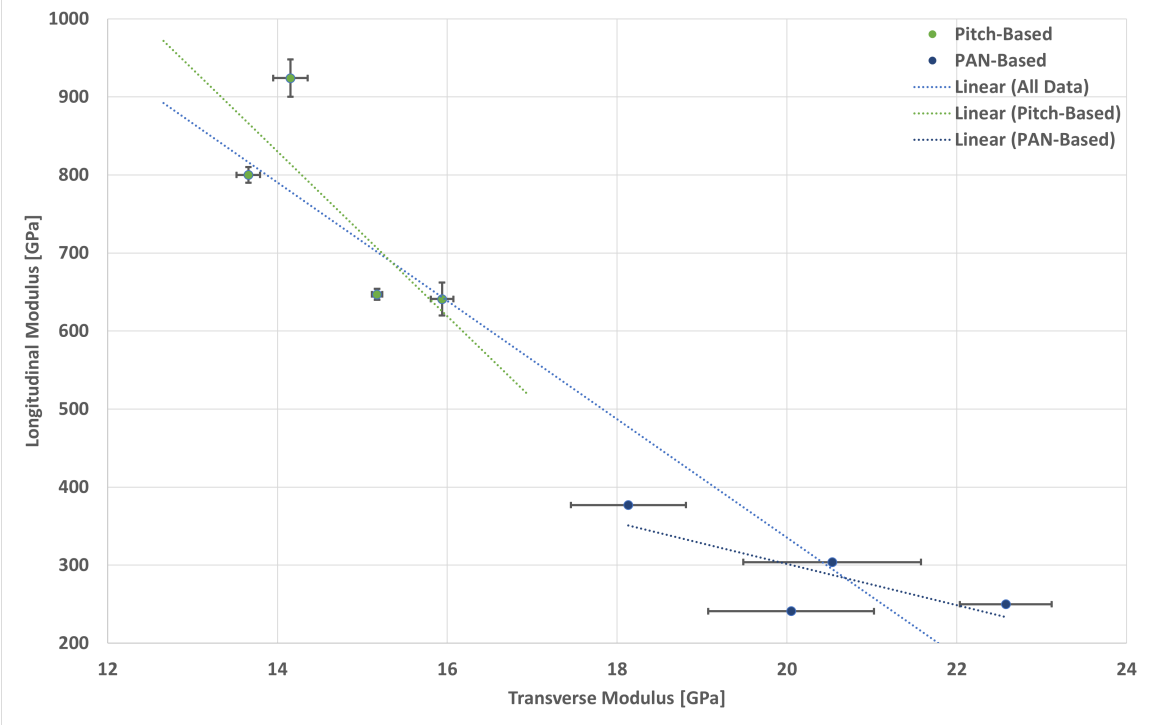

Figure 53. Measured Carbon Fiber Transverse Modulus.

Additionally, the PAN-based fiber measurements all possessed larger measurement error than measurement of Pitch-based fibers, despite using similar PeakForce Setpoint values, identical other imaging parameters, and as many or more total images. This larger measurement error may be due to the differences in fiber microstructure that would give rise to slightly different mechanical properties at different points on a fiber. Pitch-based fibers possess a microstructure characterized by both high lo-

cal order and long range order, while PAN-based fibers have a microstructure with relatively lower long range order.

The observed variation likely results from how these different microstructures result in a variation in the Poisson ratio, since changes in the orientation of the crystallites would lead to a change in how the fiber deforms under applied stress. As described in Equation 7, and its constituent terms, the Poisson ratio is squared in most of the error terms, and so any relative difference from one point to the next will be amplified. This could result in slightly different measured moduli at different points on a fiber fragment, thus decreasing the precision of the measurement.

Another material property that differentiates the Pitch and PAN-based fibers is their densities. The pitch-based fibers have densities between 2.1-2.21 $\frac{g}{cm^3}$, the PAN based fibers have densities between 1.77-1.81 $\frac{g}{cm^3}$. This difference of between 16-24% from the pitch to PAN fibers may contribute to the difference in measurement error, since the physical composition of the Pitch-based fibers will be predominantly more carbon and than the PAN-based fiber, which may posses larger void spaces or defects allowing environmental intrusion, which could result in variations from the true fiber modulus. For an indentation into one of these void spaces or defect regions, the tip will interact with a different adhesion force due to the difference in the composition of the void or defect, in addition to a change in the indentation depth due to the different modulus of the void or defect region.

Another possible reason for the larger measurement error found in the PAN fiber sample is environmental aging. While the Pitch-based fibers and the PAN-based HEXCEL HM63 and AS4D fibers were all newly procured from the manufacturers, the other fibers (M40jb, IM9, TRH50, 34-700WD, and AS4-GP-12K) had all been procured a year prior through third-party sources. These fibers were stored in an environmentally controlled lab out of direct sunlight during the period of this research,

however prior storage conditions are unknown. Additionally, during the period of this research a broken window seal was discovered resulting in water intrusion during the summer, likely increasing the relative humidity in the lab.

No studies were found specifically addressing the effect of environmental aging on the transverse modulus of single strand carbon fibers. Research into the degradation of carbon fibers due to age and environmental conditions predominantly focuses on the degradation of carbon fiber reinforced composites, not of the degradation of the fibers in isolation. Additionally, these studies focus on extreme conditions not present in the lab, such as very high or low temperature or submersion in fluid. However, these studies are valuable in assessing the likely mechanisms of fiber aging due to their characterization of the failure modes after composite aging, and for the change in mechanical properties they do characterize for the composites. The change in the composite behavior should indicate a correlated change in the properties of the carbon fibers.

A number of studies have found that aging decreases the tensile strength and tensile modulus of composites and leads to failure at the fiber-matrix interface. Baurova found that a carbon fiber composite consisting of UKN-5-500 carbon fiber and epoxide, used in conductive sensors, exposed to natural climatic conditions for one year would experience a 7.7% reduction in tensile strength and 28.1% reduction in electrical resistance, in addition to observing a thin film of contamination on the carbon fiber surface within the composite.[29] Kollia et. al found a similar result for carbon fiber-reinforced cyanate ester composites, comparing specimens aged 30 days in either an inert nitrogen or normal air environment at 230°C, resulting in a 7.5% reduction in tensile strength for the specimen exposed to air. Additionally, the failure mechanism of the composite was examined, finding both interfacial debonding and matrix cracking.[30] Shaoquan et. al. found that for a carbon fiber/bismaleimide composite

used in aircraft structural applications, a 47.4% decrease in the tensile strength and a 0.782-1.027% loss of mass results for 1000 hours of aging in 200°C air. Finally, Cheng et. al. tested carbon/flax composite fibers submerged in water at 60°C for 289 hours, finding that the tensile strength and tensile modulus of the carbon/flax fiber decreased by 20.5% and 3.26%, respectively.[31]

These aging effects are indicative of degradation of the carbon fiber within the matrix. Failure at the fiber-matrix interface indicates a change in carbon fiber surface, most likely due to oxidation with air that diffuses into the composite via the defect channels created between the matrix and fiber. Additionally, the loss of mass further indicates chemical reaction with the air leading to low levels of off-gassing from the fiber. While two of the studies described here are for extreme conditions, it is notable that the same decrease in tensile strength resulted from a year of exposure to normal climate conditions. Based on these studies, it is reasonable to conclude that aging in air for long periods of time leads to a change in the carbon fiber microstucture and surface chemistry, and this change due to aging likely would have some influence on the transverse modulus of the fiber.

5.2.1 Exclusion of Extreme Values

Exclusion of indentations characterized by outliers in indentation, Peak Force, and adhesion was found to be a relatively simple method to both improve measurement precision and remove values resulting in non-physical modulus measurement. The finding that these outliers are few in number and do not strongly influence the mean modulus is also important, since it demonstrates that the "law of large numbers" is powerfully applied by the PF-QNM method, in which a very large number of measurements are used to arrive at an accurate average value.

5.3 Error Contributions from Material Assumptions

While this study did not seek to accurately determine the true transverse Young modulus, the method could certainly be used to do so. Of more significance to this study is the degree to which any measurement by this method is limited by an inherent error bound due to the assumption underlying the instrumental technique. Specific to this method, the assumptions that were expected to contribute significantly to this intrinsic error are the assumption that the Poisson ratio used is accurate, and that the tip hardness assumption is valid, that is the tip is sufficiently hard that it does not deform when indenting the fiber surface.

It was found that not applying the tip hardness assumption results in a slightly larger measured modulus, which is expected since it is known that the AFM tip is degraded over time, and so mechanical deformation is almost certainly occurring. This deformation, however, appears to be consistent over the entire sample, since the modulus found without the tip hardness assumption is within one standard deviation of the value found using the assumption.

With regard to the assumption that the Poisson ratio is accurate, this is a question of confidence in literature or handbook values. This method requires this ratio to be known, and so internal consistency is more important than true accuracy. In order to compare to other research using a different Poisson ratio, or to provide a bound of uncertainty in reporting a "true" modulus, understanding the influence of the ratio is important. Unlike the tip hardness assumption, which generates variation that changes between images, varying the Poisson ratio results in the same relative variation in the modulus mean across all images. This is due to the Poisson ratio being part of the calculation to determine the modulus after the force curve is fit, and thus any variation is not influenced by the actual measurement. The overall variation in modulus is a function of the bounding values of the ratio, but was found to be

slightly larger than the maximum variation due to the tip hardness assumption.

5.4 Variation between Measured Modulus and Calculated Modulus

Variation in the modulus calculated using a post-processing script and that reported by the instrument was initially expected. Due to the size of the adhesion region of the force curve, the use of this absolute minimum value was expected to produce lower modulus values. Additionally, using the analogy to mechanical stress-strain curves, it is obvious that the adhesion region is non-linear and cannot be treated with simple linear relationships, just as plastic deformation of a material is inherently non-linear. For the method fitting between the Peak Force and where the forces are balanced, this was expected to produce a larger modulus because it neglects the adhesion behavior. It is unclear from this study, and from literature, why the DMT modulus fitting is appropriately applied over a portion of both the adhesion and repulsive regions of the force curve. The force fit region, however, is a user-controlled setting for the Bruker AFM used, and so it merits further study whether varying this fit region results in reduced measurement error.

5.5 Modulus Calculation Directly from Force Curves

The method of calculating the modulus directly from the raw AFM force curves was initially thought to be the most suitable for study to reduce the measurement error, however it proved to be more challenging than expected due both the obtuseness of the proprietary software and the difficulty of reproducing the results of the instrument. It was also found that this method, as implemented, did not result in improved measurement precision. This is likely due to an incorrect implementation and replication of the Bruker software routine, and so it may be possible in the future to improve upon the force curve analysis method as well as implement the other

statistical exclusion techniques described in this study.

5.6 Correlations Between Transverse Modulus Error and Other Measurement Errors

Given that the DMT model is a relatively simple analytical model, and the method of error propagation used to develop the theoretical errors are also simple, the strength of the correlation between theoretical and PeakForce modulus errors is very good and should serve as a guide for determining modifications to the current methodology to decrease the measurement error. It is also reasonable that the Spearman R correlations, which do not assume linear correlation prior to testing, are stronger than the Pearson R correlations, since the measured values of the various interaction data between the AFM tip and sample result from the single indentation force curve at the measurement point, and so the relationship between the indentation depth, peak force, and adhesion force must be more complex than a simple linear relationship. The strength of correlation between the modulus error and indentation depth error was expected based on the bivariate relationship found by analysis of the 2D histogram, while the correlation between the modulus and adhesion force was stronger than expected. For future refinement to this methodology, optimization to minimize error in these two components of the tip-sample interaction would be the most promising. Other studies have demonstrated methods of ablating the carbon fiber to expose a flat cross-section of the fiber core, which is likely the ideal method of eliminating the surface roughness of the fiber, leading to more consistent indentation and adhesion.[32]

5.7 Summary

This chapter analyzes the results of this study, with specific interest in the unexpected lack of correlation between the longitudinal and transverse modulus for the PAN fibers, and the increased measurement error for the same fibers. This deviation from the correlation found by Veigas for the same fibers is most likely attributable to the difference in fiber microstructure, which may contribute to the larger measurement error, due to the lack of long-range order in PAN fibers. Additionally, fiber aging and oxidation by the atmosphere may also play a role, due to the increased age of some of the PAN fibers and their uncertain storage conditions prior to this study.

This chapter also assesses the result of outlier exclusion, finding that the very large number of measurements possible using the PF-QNM method provides a statistically very large number of measurements that preclude the influence of individual outliers strongly effecting the overall distribution. The variation due to the tip hardness assumption and uncertainty in Poisson's ratio are also as expected, with variation similar in magnitude to the PeakForce calculated modulus error. While this variation is important to precise characterizations of the fibers, it is less significant for the purpose of demonstrating the modulus correlation when using the same instrument.

The variation of the calculated modulus results from the PeakForce calculated modulus results, while expected, merit further study to determine if the method utilizing the raw force curves can be improved and optimized, while the results of the other simpler method indicate some of the underlying assumptions used in the Bruker technique. Finally, the correlation of errors is found to be as expected, since the PF-QNM technique is capable of measuring many different interactions arising from an indentation measurement, it is reasonable to expect that all of these interactions will be correlated to some degree. The result provides direction towards further technique improvement to decrease the measurement error by reduction of the indentation depth

and adhesion force variation.

VI. Conclusion

The precision measurement and characterization of carbon fiber material properties is an important area of research that both supports their utilization in novel composite materials for new and innovative applications as well as the field of treaty monitoring and verification. While the longitudinal mechanical behavior of carbon fibers is well characterized, both within a variety of composite and alone, the transverse mechanical properties are poorly characterized, with studies producing values with relatively large errors and with difficult sample preparation procedures. [32] This study sought to refine the demonstrated method developed by Veigas, which showed that a simple sample preparation procedure and measurement method using AFM PF-QNM could be applied to measure the fiber transverse modulus. In order to improve Veigas' method, it was necessary to decrease the measurement error so that precision characterization was possible.

This study demonstrated that, for single strand carbon fiber samples for which traditional tensile testing is impossible, the transverse fiber modulus can be used as an analog for differentiating between modulus regimes and fiber precursor types. Additionally, this study demonstrated the feasibility of high precision measurement of the transverse modulus, achieving between 0.5-5% weighted measurement error for a variety of carbon fibers spanning a broad range of longitudinal modulus and both Pitch and PAN precursor fibers. This decrease in measurement error was achieved via statistical methods of excluding individual indentations within a single image measurement based on the indentation depth, peak force, and adhesion force, as well as exclusion criteria for complete images.

Alternative methods were explored for calculating the transverse modulus from other measurement data, such as the indentation depth, peak force, and adhesion force. The two simplest methods attempted to approximate the measured force curve,

but were not successful in replicating either the instrument-reported values or the modulus correlation behavior. This is due to the method by which the instrument fits the force curve to calculate the modulus, which includes a portion of both the linear and non-linear areas of the curve. A more complicated method of calculating the modulus using the raw force curve data was also attempted, but was also unsuccessful in replicating the instrument-reported modulus data. This method was also found to be infeasible for analysis of many images, with the time required to manually manipulate the data files within the instrument software increasing the single image analysis time by an order of magnitude.

The modulus correlation between the transverse and longitudinal modulus shown by this study requires further study, specifically the unexpected behavior of the PAN-based fibers. These fibers demonstrated higher measurement error than the Pitch-based and no apparent correlation between the moduli. This is suspected to be due to the storage conditions of the PAN fibers and their unknown age, which may have resulted in environmental oxidation degrading the fiber, leading to both lower than expected transverse moduli for some fibers and larger error due to differing rates of oxidation within the fiber tow. Additionally, it is not known if this difference in measurement errors between precursors is due to the microstructure of the fibers themselves, for which the long-range order within the fiber is known to vary between the Pitch and PAN fibers.

The precision measurement of the transverse modulus, however, is only a single step in developing alternative methods for determining if an unknown fiber sample is derived from an export controlled material. Since the export control criteria are defined in terms of both the specific tensile modulus and specific tensile strength, two additional methods must be developed. First, since the criteria are specific in terms of specific strength and modulus, the density of the fiber must be determined. This

is problematic since fiber oxidation is known to result in mass loss and thus density change over time. Additionally, the tensile strength must be determined via a means other than indentation since compressive strength and tensile strength can vary for a single material. Luckily, methods do exist for determining nanoscale tensile strength using micro-electromechanical systems (MEMS), though these techniques are beyond the scope of this study. [33] Efforts to apply these methods to the same samples used for this study would be a fruitful avenue for future study.

Appendix A. Data Analysis Script

1.1 Script for Analysis of Data Exported from .spm Images

This script was utilized for analysis of the data exported from the .spm image file into a .txt file, consisting of tab-separated columnar data for each of the eight data channels the Bruker AFM is capable of capturing for a single image scan.

```
1
2  import pandas as pd
3  from scipy.optimize import curve_fit
4  from scipy.stats import skewnorm
5  from scipy.stats import norm
6  from scipy.special import erf
7  from scipy.special import gamma
8  from scipy.stats import pearsonr
9  from scipy.stats import spearmanr
10 import numpy as np
11 import matplotlib.pyplot as plt
12 import math
13
14 #%% Functions
15
16 # PDF Fitting Functions
17
18 def gaussian(x, a, b, c):
19     return a*np.exp(-np.power(x - b, 2)/(2*np.power(c, 2)))
20
21 def skew(x, a, b, c, alpha):
22     return (2/c)*(a*np.exp(-np.power(x - b, 2)/(2*np.power(c,
          2)))) * (1/2) * (1 + erf((alpha*((x-b)/c))/np.sqrt(2)))
23
24 def skew_mode(b,c,alpha):
25     delta = alpha/np.sqrt(1+alpha**2)
26     mu = np.sqrt(2/np.pi)*delta
27     sigma = np.sqrt(1-mu**2)
28     gamma = ((4-np.pi)/2) * ((delta*np.sqrt(2/np.pi))
          **3/(1-(2*delta**2)/np.pi)**(3/2))
29     m_0 = mu - (gamma*sigma)/2 - (np.sign(alpha)/2)*(np.exp
          ((2*np.pi)/np.abs(alpha)))
30     mode = b + c*m_0
31     return mode
32
33 def skew_mean(b,c,alpha):
34     delta_skew = alpha/np.sqrt(1+alpha**2)
35     mean = b + c*delta_skew*np.sqrt(2/np.pi)
36     return mean
37
38 def skew_std(c,alpha):
39     delta_skew = alpha/np.sqrt(1+alpha**2)
40     std = np.sqrt(c**2 * (1-(2*delta_skew**2)/np.pi))
```

```
41      return std
42
43  def gennorm(x, a, b, c, alpha,):
44      return a*np.exp(-np.power(np.abs(x-b)/alpha,c))
45
46  def laplace(x,a,b,c):
47      return a*np.exp(-np.abs(x-b)/c)
48
49  def multi_gaussian(x, *pars):
50      offset = pars[-1]
51      g1 = gaussian(x, pars[0], pars[1], pars[2])
52      g2 = gaussian(x, pars[3], pars[4], pars[5])
53      return g1 + g2 + offset
54
55  def multi_skew(x, *pars):
56      offset = pars[-1]
57      g1 = skew(x, pars[0], pars[1], pars[2],pars[3])
58      g2 = skew(x, pars[4], pars[5], pars[6],pars[7])
59      return g1 + g2 + offset
60
61  def peak_force(PF_setpoint,PF_error):
62      PF = PF_setpoint + PF_error
63      return PF
64
65  def DMT_mod_adhes(F_tip,F_adh,radius_tip,depth,poisson_samp):
66      E_star = (3/4) * ((F_tip + F_adh)/(np.sqrt(radius_tip*
           depth**3)))
67      E_samp = (1-poisson_samp**2)*E_star
68      return E_samp
69
70  def sample_modulus(E_star,poisson_samp):
71      E_samp = (1-poisson_samp**2)*E_star
72      return E_samp
73
74  def DMT_mod_matldep(F_tip,F_adh,radius_tip,depth,poisson_samp)
        :
75      poisson_tip = 0.2
76      E_tip = 200.0
77      E_star = (3/4) * ((F_tip + F_adh)/(np.sqrt(radius_tip*
           depth**3)))
78      E_samp = (1-poisson_samp**2)/(1/E_star - (1-poisson_tip
           **2)/E_tip)
79      return E_samp
80
81  def DMT_mod_noadh(F_tip,poisson_samp,radius_tip,depth):
82      E = (3/4)*F_tip*(1-poisson_samp**2) / (np.sqrt(radius_tip*
           depth**3))
83      return E
84
85  # Tip Geometry Functions
86
87  def tip_fit(x, a, b, c, d ):
88      return a*(b*x+c)**(1/3) + d
89
90  def power_law(x, a, b):
91      return a*np.power(x, b)
92
93  def sph_indent_geo(height,tip_rad):
```

```
94      vol = (1/3)*np.pi*height**2*(3*tip_rad-height)
95      surf = 2*np.pi*(tip_rad*height-np.sqrt(height*(2*tip_rad-
            height)))
96      return vol, surf
97
98  def tip_geo(height,SA,FA,BA):
99      s = []
100     w = []
101     r = []
102     l = []
103     f = []
104     L = []
105     a = []
106     b = []
107     p1 = []
108     p2 = []
109     vol = []
110     surf = []
111     for i in range(len(SA)):
112         s.append(height/(np.sin(np.radians(90-SA[i]))))
113         w.append((height*np.sin(np.radians(SA[i])))/(np.sin(np
                .radians(90-SA[i]))))
114         r.append(height/(np.sin(np.radians(90-BA[i]))))
115         l.append((height*np.sin(np.radians(BA[i])))/(np.sin(np
                .radians(90-BA[i]))))
116         f.append(height/(np.sin(np.radians(90-FA[i]))))
117         L.append((height*np.sin(np.radians(FA[i])))/(np.sin(np
                .radians(90-FA[i]))))
118         a.append(np.sqrt(L[i]**2 + w[i]**2))
119         b.append(np.sqrt(l[i]**2 + w[i]**2))
120         p1.append((f[i]+s[i]+a[i])/2)
121         p2.append((r[i]+s[i]+b[i])/2)
122         vol.append((2*w[i]*(L[i]+l[i])*height)/3)
123         surf.append(2*np.sqrt(p1[i]*(p1[i]-f[i])*(p1[i]-s[i])
                *(p1[i]-a[i])) + 2*np.sqrt(p2[i]*(p2[i]-r[i])*(p2[i
                ]-s[i])*(p2[i]-b[i])))
124     return vol, surf
125
126 # Other Functions
127
128 def truncate(number, digits) -> float:
129     stepper = 10.0 ** digits
130     return math.trunc(stepper * number) / stepper
131
132 def zipper(y_list,x_list):
133     zipped_lists = zip(x_list, y_list)
134     sorted_zipped_lists = sorted(zipped_lists)
135     sorted_ylist = [element for _, element in
            sorted_zipped_lists]
136     return sorted_ylist
137
138 def std_bars(data,hist_y):
139     low_val = np.mean(data)-np.std(data)
140     high_val = np.mean(data)+np.std(data)
141     low_bar = (np.full((50,1),low_val),np.linspace(0,np.max(
            hist_y)))
142     high_bar = (np.full((50,1),high_val),np.linspace(0,np.max(
            hist_y)))
```

```
143     return low_bar,high_bar
144
145 def DMT_err(F_tip,F_adh,tiprad,indent):
146     a = np.mean(F_tip)
147     std_a = np.std(F_tip)
148     b = np.mean(F_adh)
149     std_b = np.std(F_adh)
150     c = np.mean(tiprad)
151     std_c = np.std(tiprad)
152     d = np.mean(indent)
153     std_d = np.std(indent)
154     da = 3/(4*np.sqrt(c*d**3))
155     db = -3/(4*np.sqrt(c*d**3))
156     dc = (3*(b-a)*d**3)/(8*(c*d**3)**(3/2))
157     dd = (9*(b-a)*c*d**2)/(8*(c*d**3)**(3/2))
158     std_dev = np.sqrt(da**2*std_a**2 + db**2*std_b**2 + dc**2*
            std_c**2 + dd**2*std_d**2)
159     return std_dev
160
161 def point_slope(x1,x2,y1,y2):
162     m = (y1-y2)/(x2-x1)
163     return m
164
165 def mod_from_pars(par,tip_rad):
166     E_star = par/((4/3)*np.sqrt(tip_rad))
167     return E_star
168
169 #%% The What Do Box
170
171 tip_name = r'03DEC20#1'
172
173 path = r'.\December\03DEC20 - K13C2U'
174 file = r'03DEC20_K13C2U.0_00098.spm.txt'
175 channel = r'{}\{}'.format(path,file)
176 fiber_name = r'03DEC20 K13C2U'
177 fig_temp = r'{}\{}'.format(path,fiber_name)
178 # mult_fit = 'Measured'
179 # mult_fit = 'Tip Fit'
180 mult_fit = 'Tip & Matl Props'
181
182 PF_setpoint = 569.9
183 PF_str = 'PF Setpoint = {} nN'.format(truncate(PF_setpoint,5))
184
185 # Material Properties
186 nu_samp = 0.3
187 nu_tip = 0.2
188 E_tip = 200.0
189 tip_str = 'nu_tip = {}, E_tip = {}'.format(nu_tip,E_tip)
190
191 glue_mean = 8.2738
192 glue_std = 2.126524
193
194 # Image Values
195 scans = 256
196 lines = 64
197
198 # Indentation Limits for Tip Fitting
199 indent_min = 0.5
```

```
200 indent_max = 20.0
201
202 # Outlier limit - multiple of measurement standard deviation
203 out_lim = 2
204
205 # Verticle Bars on plots to show number of standard deviations
      on plots
206 vert_bars = 2
207
208 # Assign data channel headings to variables
209 height_sens = 'Height_Sensor(nm)'
210 pf_error = 'Peak_Force_Error(nN)'
211 DMT_mod = 'DMTModulus(MPa)'
212 indent = 'Indentation(nm)'
213 adhes = 'Adhesion(nN)'
214 dissip = 'Dissipation(eV)'
215 height = 'Height(nm)'
216 peakforce = 'Peak_Force(nN)'
217
218 # Data Channel of interest for histogram PDF fitting
219 histfit = DMT_mod
220
221 # Print all results
222 verbose_calculations = True          # Print all numerical
    results
223 verbose_plots = True                 # Print all visual results
224
225 # Tip Analysis
226 power = False                        # Fit the tip to a power
    law function
227 poly = True                          # Fit the tip to a
    polynomial
228 tip_vol_compare = False              # Plot the nominal
    geometrical tip shape and the functional tip shape
229
230 # AFM Image Import & Display
231 import_data = True                   # Import data from the AFM
      .spm.txt file
232 no_outliers = False                   # Remove outliers from
    data set based on multiple of standard deviations
233 only_outliers = False                # Use only outliers from
    data set based on multiple of standard deviations
234 make_arrays = True                   # Assembles array of
    calculated modulus
235 force_curve = False                   # Build approximate force
      curve assuming indentation depth is zero force
236 plot_mods = True                     # Plot the modulus
    histograms
237 plot_tipfitmod = False               # Determine the effect of
    using the tip modulus and Poisson Ratio to deconvolve the
238 poisson_study = False                # Determine the effect of
    Poisson's Ratio on the calculated modulus
239 channel_hists = False                # Plot histograms of all
    data channels in the image
240
241 # Data Processing
242 outlier_test = False                 # Count the number of
    outliers in the data set based on a multiple of standard
```

```
    deviations, determine coincidence of the outliers, and
    compare values to complete data set
adhesion_func = False                # Determine relationship
    between measurement data and adhesion force
histogram_fitter = False             # Fit data histograms to
    various Probability Distribution Functions
multi_fit = False                   # Fit data to multiple
    Gaussian function

#%% Tip Profile Function Fitting Script

tip_file = r'tip_profiles.csv'
tip_data = pd.read_csv(tip_file,sep=',')

x = tip_data['Depth (nm)']
# yfit = tip_data[r'{}'.format(tip_name)]

tip_name = r'04DEC20#1'
tip_name1 = r'08DEC20#1'
yfit = tip_data[r'{}'.format(tip_name)]
yfit1 = tip_data[r'{}'.format(tip_name1)]


# Power Law Fit
if power == True:
    pars = np.array([])
    pars, cov = curve_fit(f=power_law, xdata=x, ydata=yfit, p0
        =[1,1], bounds=(-np.inf, np.inf))                # Get
        the standard deviations of the parameters (square roots
         of the # diagonal of the covariance)
    residuals = yfit - power_law(x, *pars)
    ss_res = np.sum(residuals**2)
    ss_tot = np.sum((yfit-np.mean(yfit))**2)
    r_squared = 1 - (ss_res / ss_tot)
    r2_str = 'R^2 = {}'.format(truncate(r_squared,5))
    fit_str = 'The tip fit power law is r(z) = {}*z^{}'.format
        (truncate(pars[0],3),truncate(pars[1],3))
    print(r2_str)
    print(fit_str)

# Polynomial Fit
if poly == True:

    # # Polynomial Order comparison
    # order = 4
    # poly_pars,poly_res = np.polyfit(x,yfit,deg=order,full=
        True)[0:2]
    # ss_tot = np.sum((yfit-np.mean(yfit))**2)
    # r_squared_poly = 1 - (poly_res / ss_tot)
    # r2_str_poly = 'R^2 = {}'.format(truncate(float(
        r_squared_poly),5))
    # # fit_str_poly = 'The tip fit polynomial is r(z) = {}*z
        ^3 + {}*z^2 + {}*z + {}'.format(truncate(poly_pars
        [0],5),truncate(poly_pars[1],3),truncate(poly_pars
        [2],3),truncate(poly_pars[3],3))
    # fit_str_poly = 'The tip fit polynomial is r(z) = {}*z^4
        + {}*z^3 + {}*z^2 + {}*z + {}'.format(truncate(
        poly_pars[0],3),truncate(poly_pars[1],3),truncate(
```

```
            poly_pars[2],3),truncate(poly_pars[3],3),truncate(
            poly_pars[4],3))
285         # # fit_str_poly = 'The tip fit polynomial is r(z) = {}*z
            ^5 + {}*z^4 + {}*z^3 + {}*z^2 + {}*z + {}'.format(
            truncate(poly_pars[0],3),truncate(poly_pars[1],3),
            truncate(poly_pars[2],3),truncate(poly_pars[3],3),
            truncate(poly_pars[4],3),truncate(poly_pars[5],3))
286         # print(fit_str_poly)
287         # print(r2_str_poly)
288
289         # order = 3
290         # poly_pars1,poly_res1 = np.polyfit(x,yfit,deg=order,full=
            True)[0:2]
291         # ss_tot = np.sum((yfit-np.mean(yfit))**2)
292         # r_squared_poly1 = 1 - (poly_res1 / ss_tot)
293         # r2_str_poly1 = 'R^2 = {}'.format(truncate(float(
            r_squared_poly1),5))
294         # fit_str_poly1 = 'The tip fit polynomial is r(z) = {}*z^3
             + {}*z^2 + {}*z + {}'.format(truncate(poly_pars[0],5),
            truncate(poly_pars[1],3),truncate(poly_pars[2],3),
            truncate(poly_pars[3],3))
295         # # fit_str_poly = 'The tip fit polynomial is r(z) = {}*z
            ^4 + {}*z^3 + {}*z^2 + {}*z + {}'.format(truncate(
            poly_pars1[0],3),truncate(poly_pars1[1],3),truncate(
            poly_pars1[2],3),truncate(poly_pars1[3],3),truncate(
            poly_pars1[4],3))
296         # # fit_str_poly = 'The tip fit polynomial is r(z) = {}*z
            ^5 + {}*z^4 + {}*z^3 + {}*z^2 + {}*z + {}'.format(
            truncate(poly_pars[0],3),truncate(poly_pars[1],3),
            truncate(poly_pars[2],3),truncate(poly_pars[3],3),
            truncate(poly_pars[4],3),truncate(poly_pars[5],3))
297         # print(r2_str_poly1)
298         # print(fit_str_poly1)
299
300         # order = 5
301         # poly_pars2,poly_res2 = np.polyfit(x,yfit,deg=order,full=
            True)[0:2]
302         # ss_tot = np.sum((yfit-np.mean(yfit))**2)
303         # r_squared_poly2 = 1 - (poly_res2 / ss_tot)
304         # r2_str_poly2 = 'R^2 = {}'.format(truncate(float(
            r_squared_poly2),5))
305         # # fit_str_poly = 'The tip fit polynomial is r(z) = {}*z
            ^3 + {}*z^2 + {}*z + {}'.format(truncate(poly_pars
            [0],5),truncate(poly_pars[1],3),truncate(poly_pars
            [2],3),truncate(poly_pars[3],3))
306         # fit_str_poly2 = 'The tip fit polynomial is r(z) = {}*z^4
             + {}*z^3 + {}*z^2 + {}*z + {}'.format(truncate(
            poly_pars2[0],3),truncate(poly_pars2[1],3),truncate(
            poly_pars2[2],3),truncate(poly_pars2[3],3),truncate(
            poly_pars2[4],3))
307         # # fit_str_poly = 'The tip fit polynomial is r(z) = {}*z
            ^5 + {}*z^4 + {}*z^3 + {}*z^2 + {}*z + {}'.format(
            truncate(poly_pars[0],3),truncate(poly_pars[1],3),
            truncate(poly_pars[2],3),truncate(poly_pars[3],3),
            truncate(poly_pars[4],3),truncate(poly_pars[5],3))
308         # print(r2_str_poly2)
309         # print(fit_str_poly2)
310
```

```
311     # New and Worn Tip comparison
312     order = 4
313     poly_pars,poly_res = np.polyfit(x,yfit,deg=order,full=True
            )[0:2]
314     ss_tot = np.sum((yfit-np.mean(yfit))**2)
315     r_squared_poly = 1 - (poly_res / ss_tot)
316     r2_str_poly = 'R^2 = {}'.format(truncate(float(
            r_squared_poly),5))
317     # fit_str_poly = 'The tip fit polynomial is r(z) = {}*z^3
            + {}*z^2 + {}*z + {}'.format(truncate(poly_pars[0],5),
            truncate(poly_pars[1],3),truncate(poly_pars[2],3),
            truncate(poly_pars[3],3))
318     fit_str_poly = 'The tip fit polynomial is r(z) = {}*z^4 +
            {}*z^3 + {}*z^2 + {}*z + {}'.format(truncate(poly_pars
            [0],3),truncate(poly_pars[1],3),truncate(poly_pars
            [2],3),truncate(poly_pars[3],3),truncate(poly_pars
            [4],3))
319     # fit_str_poly = 'The tip fit polynomial is r(z) = {}*z^5
            + {}*z^4 + {}*z^3 + {}*z^2 + {}*z + {}'.format(truncate
            (poly_pars[0],3),truncate(poly_pars[1],3),truncate(
            poly_pars[2],3),truncate(poly_pars[3],3),truncate(
            poly_pars[4],3),truncate(poly_pars[5],3))
320     print(fit_str_poly)
321     print(r2_str_poly)
322
323     order = 4
324     poly_pars1,poly_res1 = np.polyfit(x,yfit1,deg=order,full=
            True)[0:2]
325     ss_tot1 = np.sum((yfit1-np.mean(yfit1))**2)
326     r_squared_poly1 = 1 - (poly_res1 / ss_tot1)
327     r2_str_poly1 = 'R^2 = {}'.format(truncate(float(
            r_squared_poly1),5))
328     # fit_str_poly = 'The tip fit polynomial is r(z) = {}*z^3
            + {}*z^2 + {}*z + {}'.format(truncate(poly_pars[0],5),
            truncate(poly_pars[1],3),truncate(poly_pars[2],3),
            truncate(poly_pars[3],3))
329     fit_str_poly1 = 'The tip fit polynomial is r(z) = {}*z^4 +
             {}*z^3 + {}*z^2 + {}*z + {}'.format(truncate(poly_pars
            [0],3),truncate(poly_pars[1],3),truncate(poly_pars
            [2],3),truncate(poly_pars[3],3),truncate(poly_pars
            [4],3))
330     # fit_str_poly = 'The tip fit polynomial is r(z) = {}*z^5
            + {}*z^4 + {}*z^3 + {}*z^2 + {}*z + {}'.format(truncate
            (poly_pars[0],3),truncate(poly_pars[1],3),truncate(
            poly_pars[2],3),truncate(poly_pars[3],3),truncate(
            poly_pars[4],3),truncate(poly_pars[5],3))
331     print(fit_str_poly1)
332     print(r2_str_poly1)
333
334 if verbose_plots == True:
335
336     x_fit = np.linspace(0.5,20,40)
337     # tip_rad_poly = np.polyval(poly_pars,x_fit)
338     # tip_rad_poly1 = np.polyval(poly_pars1,x_fit)
339     # tip_rad_poly2 = np.polyval(poly_pars2,x_fit)
340
341     # plt.figure()
342     # plt.plot(x,yfit/yfit*100,'--k',label='Actual Tip')
```

```
343     # # plt.plot(x_fit,tip_rad_poly/yfit*100,label='4th Order
            Poly Fit')
344     # plt.plot(x_fit,tip_rad_poly1/yfit*100,label='3rd Order
            Poly Fit')
345     # # plt.plot(x_fit,tip_rad_poly2/yfit*100,label='5th Order
            Poly Fit')
346     # plt.plot(x_fit,power_law(x_fit,*pars)/yfit*100,'r',label
            ='Power Law Fit')
347     # # plt.plot(x,yfit1,'--g')
348     # # plt.plot(x_fit,tip_rad_poly1,'r')
349     # plt.xlabel('Indentation Depth [nm]',fontsize=20)
350     # plt.xticks(fontsize=16)
351     # plt.ylabel('Tip Radius [% Diff from Measured]',fontsize
            =20)
352     # plt.yticks(fontsize=16)
353     # plt.title('Comparison of Tip Radius Fit Methods,
            Normalized to Measured Tip Radius',fontsize=24)
354     # # plt.title('Comparison of Polynomial Fit Methods,
            Normalized to Measured Tip Radius',fontsize=24)
355     # plt.legend(fontsize=20)
356     # fig_name = r'{} tip_rad.png'.format(tip_name)
357     # # plt.savefig(r'{}\{}'.format(path,fig_name),bbox_inches
            ='tight')
358     # plt.show()
359
360     # New/Worn Tip Comparison Plot
361
362     plt.figure()
363     plt.plot(x,yfit,'--b',label='New Tip')
364     plt.plot(x,yfit1,'--r',label='Worn Tip')
365     plt.xlabel('Indentation Depth [nm]',fontsize=20)
366     plt.xticks(fontsize=16)
367     plt.ylabel('Tip Radius [nm]',fontsize=20)
368     plt.yticks(fontsize=16)
369     plt.title('Absolute Change in Tip Radius after ~200,000
            Indentations',fontsize=24)
370     # plt.title('Comparison of Polynomial Fit Methods,
            Normalized to Measured Tip Radius',fontsize=24)
371     plt.legend(fontsize=20)
372     fig_name = r'{} tip_rad.png'.format(tip_name)
373     # plt.savefig(r'{}\{}'.format(path,fig_name),bbox_inches='
            tight')
374     plt.show()
375
376     plt.figure()
377     # plt.plot(x,100-yfit/yfit*100,'--b',label='New Tip')
378     plt.plot(x,((yfit1-yfit)/yfit)-1,'--r',label='Worn Tip')
379     plt.xlabel('Indentation Depth [nm]',fontsize=20)
380     plt.xticks(fontsize=16)
381     plt.ylabel('Tip Radius [% Increase from New Tip]',fontsize
            =20)
382     plt.yticks(fontsize=16)
383     plt.title('Relative Change in Tip Radius after ~200,000
            Indentations',fontsize=24)
384     # plt.title('Comparison of Polynomial Fit Methods,
            Normalized to Measured Tip Radius',fontsize=24)
385     plt.legend(fontsize=20)
386     fig_name = r'{} tip_rad.png'.format(tip_name)
```

```
387     # plt.savefig(r'{}\{}'.format(path,fig_name),bbox_inches='
            tight')
388     plt.show()
389
390 #%% Raw Data Channel Import Script
391
392 if import_data == True:
393
394     # channel = input('What is the data channel filename?')
395     # spm = r'.\Lorin Good Data\Nov 08 2019\34-700 WD Meas 2b
            .0_00000.spm'
396     # PF_setpoint = float(input('What is the PF Setpoint in nN
            ?'))
397     # tip_rad = float(input('What is the tip radius in nm?'))
398
399     data_image = pd.read_csv(channel,sep="\s+")
400
401     # PF_setpoint = linecache.getline(spm,301)
402     # PF_setpoint = float(PF_setpoint[-4:-1])
403
404     # import linecache
405     # tip_rad = linecache.getline(spm,315)
406     # tip_rad = float(tip_rad[-5:-1])
407
408     # nu_samp = linecache.getline(spm,317)
409     # nu_samp = float(nu_samp[-4:-1])
410
411     if no_outliers == False:
412         DMT_mod_array = np.array(data_image[DMT_mod])*10**-3
413         DMT_mod_array = DMT_mod_array[np.where(DMT_mod_array >
                0)]
414         mod_meas_pure = 'The mean measured DMT modulus is {}
              GPa w/ std dev {} GPa.'.format(truncate(np.mean(
              DMT_mod_array),3),truncate(np.std(DMT_mod_array),3)
              )
415         indentation = np.array(data_image[indent][np.
              logical_and(data_image[indent] < indent_max,
              data_image[indent] > indent_min)])
416         PF_total = np.array(peak_force(PF_setpoint,data_image[
              pf_error][np.logical_and(data_image[indent] <
              indent_max, data_image[indent] > indent_min)]))
417         F_adhes = np.array(data_image[adhes][np.logical_and(
              data_image[indent] < indent_max, data_image[indent]
               > indent_min)])
418
419         if verbose_calculations == True:
420             print('All Data Points')
421             print(mod_meas_pure)
422             print('The modulus precision is',np.std(
                  DMT_mod_array)/np.mean(DMT_mod_array)*100,'%.')
423
424     if no_outliers == True:
425         DMT_mod_array = np.array(data_image[DMT_mod])*10**-3
426         mod_meas_pure = 'The mean measured DMT modulus before
              outlier removal is {} GPa w/ std dev {} GPa.'.
              format(truncate(np.mean(DMT_mod_array),3),truncate(
              np.std(DMT_mod_array),3))
```

```
427         # Remove indentations that cannot be approximated by
               tip fit function
428         indentation = np.array(data_image[indent][np.
               logical_and(data_image[indent] < indent_max,
               data_image[indent] > indent_min)])
429         PF_total = np.array(peak_force(PF_setpoint,data_image[
               pf_error][np.logical_and(data_image[indent] <
               indent_max, data_image[indent] > indent_min)]))
430         F_adhes = np.array(data_image[adhes][np.logical_and(
               data_image[indent] < indent_max, data_image[indent]
                > indent_min)])
431         DMT_mod_array = np.array(DMT_mod_array[np.logical_and(
               data_image[indent] < indent_max, data_image[indent]
                > indent_min)])
432         mod_meas_sml_indent = 'The mean measured DMT modulus
               after exclusion of >20nm indentation is {} GPa w/
               std dev {} GPa.'.format(truncate(np.mean(
               DMT_mod_array),3),truncate(np.std(DMT_mod_array),3)
               )
433         # Remove indentation outliers based on defined limit
434         PF_total = np.array(PF_total[np.logical_and(
               indentation < np.mean(indentation) + out_lim*np.std
               (indentation), indentation > np.mean(indentation) -
                out_lim*np.std(indentation))])
435         F_adhes = np.array(F_adhes[np.logical_and(indentation
               < np.mean(indentation) + out_lim*np.std(indentation
               ), indentation > np.mean(indentation) - out_lim*np.
               std(indentation))])
436         DMT_mod_array = np.array(DMT_mod_array[np.logical_and(
               indentation < np.mean(indentation) + out_lim*np.std
               (indentation), indentation > np.mean(indentation) -
                out_lim*np.std(indentation))])
437         indentation = np.array(indentation[np.logical_and(
               indentation < np.mean(indentation) + out_lim*np.std
               (indentation), indentation > np.mean(indentation) -
                out_lim*np.std(indentation))])
438         mod_meas_indent_outs = 'The mean measured DMT modulus
               after indentation outlier removal is {} GPa w/ std
               dev {} GPa.'.format(truncate(np.mean(DMT_mod_array)
               ,3),truncate(np.std(DMT_mod_array),3))
439         # Remove PF outliers
440         indentation = np.array(indentation[np.logical_and(
               PF_total < np.mean(PF_total) + out_lim*np.std(
               PF_total), PF_total > np.mean(PF_total) - out_lim*
               np.std(PF_total))])
441         F_adhes = np.array(F_adhes[np.logical_and(PF_total <
               np.mean(PF_total) + out_lim*np.std(PF_total),
               PF_total > np.mean(PF_total) - out_lim*np.std(
               PF_total))])
442         DMT_mod_array = np.array(DMT_mod_array[np.logical_and(
               PF_total < np.mean(PF_total) + out_lim*np.std(
               PF_total), PF_total > np.mean(PF_total) - out_lim*
               np.std(PF_total))])
443         PF_total = np.array(PF_total[np.logical_and(PF_total <
                np.mean(PF_total) + out_lim*np.std(PF_total),
               PF_total > np.mean(PF_total) - out_lim*np.std(
               PF_total))])
```

```
444             mod_meas_pf_outs = 'The mean measured DMT modulus
                    after peak force outlier removal is {} GPa w/ std
                    dev {} GPa.'.format(truncate(np.mean(DMT_mod_array)
                    ,3),truncate(np.std(DMT_mod_array),3))
445             # Remove Adhesion outliers
446             indentation = np.array(indentation[np.logical_and(
                    F_adhes < np.mean(F_adhes) + out_lim*np.std(F_adhes
                    ), F_adhes > np.mean(F_adhes) - out_lim*np.std(
                    F_adhes))])
447             PF_total = np.array(PF_total[np.logical_and(F_adhes <
                    np.mean(F_adhes) + out_lim*np.std(F_adhes), F_adhes
                     > np.mean(F_adhes) - out_lim*np.std(F_adhes))])
448             DMT_mod_array = np.array(DMT_mod_array[np.logical_and(
                    F_adhes < np.mean(F_adhes) + out_lim*np.std(F_adhes
                    ), F_adhes > np.mean(F_adhes) - out_lim*np.std(
                    F_adhes))])
449             F_adhes = np.array(F_adhes[np.logical_and(F_adhes < np
                    .mean(F_adhes) + out_lim*np.std(F_adhes), F_adhes >
                     np.mean(F_adhes) - out_lim*np.std(F_adhes))])
450             mod_meas_adhes_outs = 'The mean measured DMT modulus
                    after adhesion force outlier removal is {} GPa w/
                    std dev {} GPa.'.format(truncate(np.mean(
                    DMT_mod_array),3),truncate(np.std(DMT_mod_array),3)
                    )
451
452             if verbose_calculations == True:
453                 print('Outliers Excluded')
454                 print(mod_meas_pure)
455                 print(mod_meas_sml_indent)
456                 print(mod_meas_indent_outs)
457                 print(mod_meas_pf_outs)
458                 print(mod_meas_adhes_outs)
459
460         if only_outliers == True:
461             DMT_mod_array = np.array(data_image[DMT_mod])*10**-3
462             mod_meas_pure = 'The mean measured DMT modulus before
                    outlier removal is {} GPa w/ std dev {} GPa.'.
                    format(truncate(np.mean(DMT_mod_array),3),truncate(
                    np.std(DMT_mod_array),3))
463             # Remove indentations that cannot be approximated by
                    tip fit function
464             indentation = np.array(data_image[indent][np.
                    logical_and(data_image[indent] < indent_max,
                    data_image[indent] > indent_min)])
465             PF_total = np.array(peak_force(PF_setpoint,data_image[
                    pf_error][np.logical_and(data_image[indent] <
                    indent_max, data_image[indent] > indent_min)]))
466             F_adhes = np.array(data_image[adhes][np.logical_and(
                    data_image[indent] < indent_max, data_image[indent]
                     > indent_min)])
467             DMT_mod_array = np.array(DMT_mod_array[np.logical_and(
                    data_image[indent] < indent_max, data_image[indent]
                     > indent_min)])
468             mod_meas_sml_indent = 'The mean measured DMT modulus
                    after exclusion of >20nm indentation is {} GPa w/
                    std dev {} GPa.'.format(truncate(np.mean(
                    DMT_mod_array),3),truncate(np.std(DMT_mod_array),3)
                    )
```

```
469         # Remove indentation outliers based on defined limit
470         PF_total = np.array(PF_total[np.logical_or(indentation
                > np.mean(indentation) + out_lim*np.std(
            indentation), indentation < np.mean(indentation) -
            out_lim*np.std(indentation))])
471         F_adhes = np.array(F_adhes[np.logical_or(indentation >
                np.mean(indentation) + out_lim*np.std(indentation)
            , indentation < np.mean(indentation) - out_lim*np.
            std(indentation))])
472         DMT_mod_array = np.array(DMT_mod_array[np.logical_or(
            indentation > np.mean(indentation) + out_lim*np.std
            (indentation), indentation < np.mean(indentation) -
                out_lim*np.std(indentation))])
473         indentation = np.array(indentation[np.logical_or(
            indentation > np.mean(indentation) + out_lim*np.std
            (indentation), indentation < np.mean(indentation) -
                out_lim*np.std(indentation))])
474         mod_meas_indent_outs = 'The mean measured DMT modulus
            after indentation outlier removal is {} GPa w/ std
            dev {} GPa.'.format(truncate(np.mean(DMT_mod_array)
            ,3),truncate(np.std(DMT_mod_array),3))
475         # Remove PF outliers
476         indentation = np.array(indentation[np.logical_or(
            PF_total > np.mean(PF_total) + out_lim*np.std(
            PF_total), PF_total < np.mean(PF_total) - out_lim*
            np.std(PF_total))])
477         F_adhes = np.array(F_adhes[np.logical_or(PF_total > np
            .mean(PF_total) + out_lim*np.std(PF_total),
            PF_total < np.mean(PF_total) - out_lim*np.std(
            PF_total))])
478         DMT_mod_array = np.array(DMT_mod_array[np.logical_or(
            PF_total > np.mean(PF_total) + out_lim*np.std(
            PF_total), PF_total < np.mean(PF_total) - out_lim*
            np.std(PF_total))])
479         PF_total = np.array(PF_total[np.logical_or(PF_total >
            np.mean(PF_total) + out_lim*np.std(PF_total),
            PF_total < np.mean(PF_total) - out_lim*np.std(
            PF_total))])
480         mod_meas_pf_outs = 'The mean measured DMT modulus
            after peak force outlier removal is {} GPa w/ std
            dev {} GPa.'.format(truncate(np.mean(DMT_mod_array)
            ,3),truncate(np.std(DMT_mod_array),3))
481         # Remove Adhesion outliers
482         indentation = np.array(indentation[np.logical_or(
            F_adhes > np.mean(F_adhes) + out_lim*np.std(F_adhes
            ), F_adhes < np.mean(F_adhes) - out_lim*np.std(
            F_adhes))])
483         PF_total = np.array(PF_total[np.logical_or(F_adhes >
            np.mean(F_adhes) + out_lim*np.std(F_adhes), F_adhes
                < np.mean(F_adhes) - out_lim*np.std(F_adhes))])
484         DMT_mod_array = np.array(DMT_mod_array[np.logical_or(
            F_adhes > np.mean(F_adhes) + out_lim*np.std(F_adhes
            ), F_adhes < np.mean(F_adhes) - out_lim*np.std(
            F_adhes))])
485         F_adhes = np.array(F_adhes[np.logical_or(F_adhes > np.
            mean(F_adhes) + out_lim*np.std(F_adhes), F_adhes <
            np.mean(F_adhes) - out_lim*np.std(F_adhes))])
```

```
486          mod_meas_adhes_outs = 'The mean measured DMT modulus
                after adhesion force outlier removal is {} GPa w/
                std dev {} GPa.'.format(truncate(np.mean(
                DMT_mod_array),3),truncate(np.std(DMT_mod_array),3)
                )
487
488          if verbose_calculations == True:
489              print('Outliers Only')
490              print(mod_meas_pure,mod_meas_sml_indent,
                    mod_meas_indent_outs,mod_meas_pf_outs,
                    mod_meas_adhes_outs)
491
492
493  if make_arrays == True:
494
495      numbins = int(2*(len(DMT_mod_array))**(1/3))
                                  # Number of Bins determine by "
                Rice's Rule"
496      DMT_mod_hist = np.histogram(DMT_mod_array,bins=numbins)
497
498      if power == True:
499          E_adh_pwr = DMT_mod_adhes(PF_total,F_adhes,power_law(
                indentation,*pars),indentation,nu_samp)
500          E_adh_pwr_array = np.array(E_adh_pwr)
501          E_adh_pwr_str = 'The mean calculated modulus w/ power
                law tip is {} GPa w/ std dev {} GPa.'.format(
                truncate(np.mean(E_adh_pwr_array),3),truncate(np.
                std(E_adh_pwr_array),3))
502          E_matl_pwr = DMT_mod_matldep(PF_total,F_adhes,
                power_law(indentation,*pars),indentation,nu_samp)
503          E_matl_pwr_array = np.array(E_matl_pwr)
504          E_matl_pwr_str = 'The mean calculated modulus w/ power
                 law tip & tip prop assn is {} GPa w/ std dev {}
                GPa.'.format(truncate(np.mean(E_matl_pwr_array),3),
                truncate(np.std(E_matl_pwr_array),3))
505          E_adh_pwr_hist = np.histogram(E_adh_pwr_array,bins=
                numbins)
506          E_matl_pwr_hist = np.histogram(E_matl_pwr_array,bins=
                numbins)
507
508          E_adh_array = E_adh_pwr_array
509          E_matl_array = E_matl_pwr_array
510          E_adh_hist = E_adh_pwr_hist
511          E_matl_hist = E_matl_pwr_hist
512
513
514          if verbose_calculations == True:
515              print('Power Law Tip')
516              print(E_adh_pwr_str)
517              print(E_matl_pwr_str)
518
519      if poly == True:
520          E_adh_poly = DMT_mod_adhes(PF_total,F_adhes,np.polyval
                (poly_pars,indentation),indentation,nu_samp)
521          E_adh_poly_array = np.array(E_adh_poly)
522          E_adh_poly_str = 'The mean calculated modulus w/ poly
                tip is {} GPa w/ std dev {} GPa.'.format(truncate(
                np.mean(E_adh_poly_array),3),truncate(np.std(
```

```
            E_adh_poly_array),3))
523         E_matl_poly = DMT_mod_matldep(PF_total,F_adhes,np.
               polyval(poly_pars,indentation),indentation,nu_samp)
524         E_matl_poly_array = np.array(E_matl_poly)
525         E_matl_poly_str = 'The mean calculated modulus w/ poly
                tip & tip prop assn is {} GPa w/ std dev {} GPa.'.
               format(truncate(np.mean(E_matl_poly_array),3),
               truncate(np.std(E_matl_poly_array),3))
526         E_adh_poly_hist = np.histogram(E_adh_poly_array,bins=
               numbins)
527         E_matl_poly_hist = np.histogram(E_matl_poly_array,bins
               =numbins)
528
529         E_adh_array = E_adh_poly_array
530         E_matl_array = E_matl_poly_array
531         E_adh_hist = E_adh_poly_hist
532         E_matl_hist = E_matl_poly_hist
533
534         if verbose_calculations == True:
535             print('Polynomial Tip')
536             print(E_adh_poly_str)
537             print(E_matl_poly_str)
538
539 #%% Plot Modulus Distributions
540
541 if plot_mods == True:
542
543     diff = np.mean(E_adh_array)-np.mean(E_matl_array)
544     rel_diff = diff/np.mean(E_adh_array)*100
545     num_devs = np.abs(diff/np.std(E_adh_array))
546
547     diff = np.std(E_adh_array)-np.std(E_matl_array)
548     rel_diff = diff/np.std(E_adh_array)*100
549
550     xbox = 1.5  #np.max(DMT_mod_array)
551     ybox1 = 0.8*np.max(E_adh_hist[0])
552     ybox2 = 0.9*np.max(E_adh_hist[0])
553     ybox3 = 0.85*np.max(E_adh_hist[0])
554     ybox4 = 1.0*np.max(E_adh_hist[0])
555     ybox5 = 0.95*np.max(E_adh_hist[0])
556
557     DMT_mean = np.mean(DMT_mod_array)
558     DMT_std = np.std(DMT_mod_array)
559     DMT_mod = np.array(data_image[DMT_mod]*10**-3)
560
561     # DMT_mod.append(DMT_mod_array[np.logical_or((
            DMT_mod_array < (DMT_mean + 3*DMT_std)),(DMT_mod_array
            > (DMT_mean - 3*DMT_std)))])
562     # DMT_index.append(np.where((DMT_mod_array > (DMT_mean +
            2*DMT_std))))
563     # DMT_index.append(np.where((DMT_mod_array < (DMT_mean -
            2*DMT_std))))
564     # DMT_mod = DMT_mod[0]
565     # DMT_mod = DMT_mod[0]
566
567     numbins_DMT = int((2*len(DMT_mod))**(1/3))
568
569     DMT_hist = np.histogram(DMT_mod,bins=numbins_DMT)
```

```
570      pd.DataFrame(DMT_hist).to_csv(r'{}\{}'.format(path,'
             DMT_hist'),index=None)
571
572  if force_curve == True:
573
574      point1 = []
575      point2 = []
576      m = []
577      E = []
578
579      for i in range(len(PF_total)):
580          point1.append((0,PF_total[i]))
581          point2.append((indentation[i],0))
582          m.append(point_slope(point1[i][0],point2[i][0],point1[
                 i][1],point2[i][1]))
583
584      E = mod_from_pars(m,np.polyval(poly_pars,indentation))
585
586      print('The mean via approximate force curve is',np.mean(E)
             ,'GPa.')
587      print('The std devn via approximate force curve is',np.std
             (E),'GPa.')
588
589      numbins = int((2*len(E))**(1/3))
590
591      curve_hist = np.histogram(E,bins=numbins)
592
593      if verbose_plots == True:
594          plt.figure()
595          plt.plot(DMT_mod_hist[1][0:-1],DMT_mod_hist[0],label='
                 Measured')
596          # plt.plot(E_adh_hist[1][0:-1],E_adh_hist[0],label='
                 Calc')
597          # plt.plot(E_matl_hist[1][0:-1],E_matl_hist[0],label='
                 Calc w/ Tip Props')
598          # plt.plot(curve_hist[1][0:-1],curve_hist[0],label='
                 Calc w/o Adhesion')
599          # plt.plot(DMT_hist[1][0:-1],DMT_hist[0])
600          # plt.legend(fontsize=20)
601          # plt.title('Tranverse Modulus Distribution Comparison
                 , Measured and Calculated, {}'.format(fiber_name),
                 fontsize=24)
602          plt.title('Transverse Modulus Distribution, {}'.format
                 (fiber_name),fontsize=24)
603          plt.xlabel('Transverse Modulus [GPa]',fontsize=20)
604          plt.xticks(fontsize=16)
605          plt.yticks(fontsize=16)
606          fig_name = r'{} mod_dist.png'.format(fiber_name)
607          # plt.savefig(r'{}\{}'.format(path,fig_name),
                 bbox_inches='tight')
608          plt.show
609
610          plt.figure()
611          plt.plot(E_adh_hist[1][0:-1],E_adh_hist[0],label='Calc
                 ')
612          plt.plot(E_matl_hist[1][0:-1],E_matl_hist[0],label='
                 Calc w/ Tip Props')
613          plt.legend(fontsize=20)
```

```
614         plt.title('Calculated Tranverse Modulus Comparison, {}
               '.format(fiber_name),fontsize=24)
615         plt.xlabel('Transverse Modulus [GPa]',fontsize=20)
616         plt.xticks(fontsize=16)
617         plt.yticks(fontsize=16)
618         fig_name = r'{} mod_dist.png'.format(fiber_name)
619         # plt.savefig(r'{}\{}'.format(path,fig_name),
               bbox_inches='tight')
620         plt.show
621
622         plt.figure()
623         plt.plot(curve_hist[1][0:-1],curve_hist[0],label='No
               Adhesion')
624         plt.plot(DMT_mod_hist[1][0:-1],DMT_mod_hist[0],label='
               Raw Data')
625         # plt.plot(DMT_hist[1][0:-1],DMT_hist[0],label='Raw
               Data')
626         plt.title('Tranverse Modulus Distribution, No Adhesion
               , {}'.format(fiber_name),fontsize=24)
627         plt.xlabel('Transverse Modulus [GPa]',fontsize=20)
628         plt.xticks(fontsize=16)
629         plt.yticks(fontsize=16)
630         plt.legend(fontsize=20)
631         fig_name = r'{} force_curve.png'.format(fiber_name)
632         # plt.savefig(r'{}\{}'.format(path,fig_name),
               bbox_inches='tight')
633         plt.show()
634
635 #%% Distribution Function Fitter
636
637 if histogram_fitter == True:
638
639     data_histfit = DMT_mod_array
640
641     mean_data = np.mean(data_histfit)
642     std_data = np.std(data_histfit)
643     rel_error_data = std_data/mean_data
644
645     fit_bins = int(2*(len(data_histfit))**(1/3))
646     hist_fit = np.histogram(data_histfit,bins=fit_bins)
647
648     x_hist = hist_fit[1][0:-1]
649     y_hist = hist_fit[0]
650
651     # Fit to Gaussian distribution
652     try:
653         pars_gauss, cov_gauss = curve_fit(f=gaussian, xdata=
               x_hist, ydata=y_hist, p0=[np.max(y_hist), np.mean(
               data_histfit), np.std(data_histfit)], bounds=(0, np
               .inf),maxfev=10000)
654         mean_gauss = pars_gauss[1]
655         std_gauss = pars_gauss[2]
656         rel_error_gauss = std_gauss/mean_gauss
657
658         res_gauss = y_hist - gaussian(x_hist,pars_gauss[0],
               pars_gauss[1],pars_gauss[2])
659         ss_res_gauss = np.sum(res_gauss**2)
660         ss_tot = np.sum((y_hist-np.mean(y_hist))**2)
```

```
661             r2_gauss = 1 - (ss_res_gauss / ss_tot)
662         except RuntimeError:
663             print('The Guassian fit failed to converge.')
664             pars_gauss = np.zeros(3)
665             res_gauss = y_hist
666             r2_gauss = 0
667 
668         # Fit to skew normal distribution
669         try:
670             pars_skew, cov_skew = curve_fit(f=skew, xdata=x_hist,
                    ydata=y_hist, p0=[np.max(y_hist), np.mean(
                    data_histfit), np.std(data_histfit), -2], bounds=(-
                    np.inf, np.inf),maxfev=10000)
671             mean_skew = skew_mean(pars_skew[1],pars_skew[2],
                    pars_skew[3])
672             std_skew = skew_std(pars_skew[2],pars_skew[3])
673             rel_error_skew = std_skew/mean_skew
674 
675             res_skew = y_hist - skew(x_hist,pars_skew[0],pars_skew
                    [1],pars_skew[2],pars_skew[3])
676             ss_res_skew = np.sum(res_skew**2)
677             ss_tot = np.sum((y_hist-np.mean(y_hist))**2)
678             r2_skew = 1 - (ss_res_skew / ss_tot)
679         except RuntimeError:
680             print('The Skew Guassian fit failed to converge.')
681             pars_skew = np.zeros(4)
682             res_skew = y_hist
683             r2_skew = 0
684 
685         # Fit to Generalied Normal Distribution
686         try:
687             pars_gennorm, cov_gennorm = curve_fit(f=gennorm, xdata
                    =x_hist, ydata=y_hist, p0=[np.max(y_hist), np.mean(
                    data_histfit), np.std(data_histfit), 1], bounds=(0,
                     np.inf),maxfev=10000)
688             mean_gennorm = pars_gennorm[1]
689             std_gennorm = np.sqrt((pars_gennorm[3]**2*gamma(3/
                    pars_gennorm[2]))/gamma(1/pars_gennorm[2]))
690             rel_error_gennorm = std_gennorm/mean_gennorm
691 
692             res_gennorm = y_hist - gennorm(x_hist,pars_gennorm[0],
                    pars_gennorm[1],pars_gennorm[2],pars_gennorm[3])
693             ss_res_gennorm = np.sum(res_gennorm**2)
694             ss_tot = np.sum((y_hist-np.mean(y_hist))**2)
695             r2_gennorm = 1 - (ss_res_gennorm / ss_tot)
696         except RuntimeError:
697             print('The Generalized Normal fit failed to converge.'
                    )
698             pars_gennorm = np.zeros(3)
699             res_gennorm = y_hist
700             r2_gennorm = 0
701 
702         # Fit to Laplace Distribution
703         try:
704             pars_laplace, cov_laplace = curve_fit(f=laplace, xdata
                    =x_hist, ydata=y_hist, p0=[np.max(y_hist), np.mean(
                    data_histfit), np.std(data_histfit)], bounds=(0, np
                    .inf),maxfev=10000)
```

```
705             mean_laplace = pars_laplace[1]
706             std_laplace = np.sqrt(2*pars_laplace[2]**2)
707             rel_error_laplace = std_laplace/mean_laplace
708
709             res_laplace = y_hist - laplace(x_hist,pars_laplace[0],
                    pars_laplace[1],pars_laplace[2])
710             ss_res_laplace = np.sum(res_laplace**2)
711             ss_tot = np.sum((y_hist-np.mean(y_hist))**2)
712             r2_laplace = 1 - (ss_res_laplace / ss_tot)
713         except RuntimeError:
714             print('The Laplace fit failed to converge.')
715             pars_laplace = np.zeros(3)
716             res_laplace = y_hist
717             r2_laplace = 0
718
719         if verbose_calculations == True:
720             if np.logical_and(np.logical_and(r2_gauss > r2_skew,
                    r2_gauss > r2_gennorm), r2_gauss > r2_laplace):
721                 print('The distribution for this measurement is
                        Guassian.')
722                 print('The arithmetic mean for the {} channel is
                        {} +/- {} %.'.format(histfit,truncate(mean_data
                        ,3),truncate(rel_error_data*100,3)))
723                 print('The Gaussian fit mean is {} +/- {} %.'.
                        format(truncate(mean_gauss,3),truncate(
                        rel_error_gauss*100,3)))
724                 print('The R^2 value for the Gaussian fit is {}.'.
                        format(truncate(r2_gauss,3)))
725                 is_gauss = True
726
727             elif np.logical_and(np.logical_and(r2_skew > r2_gauss,
                    r2_skew > r2_gennorm), r2_skew > r2_laplace):
728                 print('The distribution for this measurement is
                        Skew Guassian.')
729                 print('The arithmetic mean for the {} channel is
                        {} +/- {} %.'.format(histfit,truncate(mean_data
                        ,3),truncate(rel_error_data*100,3)))
730                 print('The skew fit mean is {} +/- {} %.'.format(
                        truncate(mean_skew,3),truncate(rel_error_skew
                        *100,3)))
731                 print('The R^2 value for the Gaussian fit is {}.'.
                        format(truncate(r2_skew,3)))
732                 is_skew = True
733
734             elif np.logical_and(np.logical_and(r2_gennorm >
                    r2_gauss, r2_gennorm > r2_skew), r2_gennorm >
                    r2_laplace):
735                 print('The distribution for this measurement is
                        Generalized Normal.')
736                 print('The arithmetic mean for the {} channel is
                        {} +/- {} %.'.format(histfit,truncate(mean_data
                        ,3),truncate(rel_error_data*100,3)))
737                 print('The Generalized Normal fit mean is {} +/-
                        {} %.'.format(truncate(mean_gennorm,3),truncate
                        (rel_error_gennorm*100,3)))
738                 print('The R^2 value for the Generalised Normal
                        fit is {}.'.format(truncate(r2_gennorm,3)))
739                 is_gennorm = True
```

```
740
741         elif np.logical_and(np.logical_and(r2_laplace >
               r2_gauss, r2_laplace > r2_skew), r2_laplace >
               r2_gennorm):
742             print('The distribution for this measurement is
                   Laplace.')
743             print('The arithmetic mean for the {} channel is
                   {} +/- {} %.'.format(histfit,truncate(mean_data
                   ,3),truncate(rel_error_data*100,3)))
744             print('The Laplace fit mean is {} +/- {} %.'.
                   format(truncate(mean_laplace,3),truncate(
                   rel_error_laplace*100,3)))
745             print('The R^2 value for the Laplace fit is {}.'.
                   format(truncate(r2_laplace,3)))
746             is_laplace = True
747
748     if verbose_plots == True:
749
750         xlims = np.linspace(mean_data - vert_bars*std_data,
               mean_data + vert_bars*std_data,100)
751
752         plt.figure()
753         plt.plot(x_hist,y_hist,'k--')
754         plt.plot(x_hist,gaussian(x_hist,pars_gauss[0],
               pars_gauss[1],pars_gauss[2]))
755         plt.plot(x_hist,skew(x_hist,pars_skew[0],pars_skew[1],
               pars_skew[2],pars_skew[3]))
756         plt.plot(x_hist,gennorm(x_hist,pars_gennorm[0],
               pars_gennorm[1],pars_gennorm[2],pars_gennorm[3]))
757         plt.plot(x_hist,laplace(x_hist,pars_laplace[0],
               pars_laplace[1],pars_laplace[2]))
758         # plt.plot(xlims,np.zeros(len(xlims)),'k')
759         plt.plot(std_bars(data_histfit,y_hist)[0][0],std_bars(
               data_histfit,y_hist)[0][1],'k--')
760         plt.plot(std_bars(data_histfit,y_hist)[1][0],std_bars(
               data_histfit,y_hist)[1][1],'k--')
761         plt.title('Comparison of Measurement & Various PDFs
               for DMT Modulus',fontsize=16)
762         plt.xlabel('Transverse Modulus [GPa]',fontsize=14)
763         # plt.title('Comparison of Measurement & Various PDFs
               for Peak Force',fontsize=16)
764         # plt.xlabel('Peak Force [nN]',fontsize=14)
765         # plt.title('Comparison of Measurement & Various PDFs
               for Indentation',fontsize=16)
766         # plt.xlabel('Indentation Depth [nm]',fontsize=14)
767         # plt.title('Comparison of Measurement & Various PDFs
               for Adhesion',fontsize=16)
768         # plt.xlabel('Adhesion Force [nN]',fontsize=14)
769         plt.legend(['Measured','Gaussian, R^2 = {}'.format(
               truncate(float(r2_gauss),3)),'Skew Gaussian, R^2 =
               {}'.format(truncate(float(r2_skew),3)),'Generalized
                Normal, R^2 = {}'.format(truncate(float(r2_gennorm
               ),3)),'Laplace, R^2 = {}'.format(truncate(float(
               r2_laplace),3)),'Measurement Standard Deviation'],
               fontsize=14)
770         plt.xlim((xlims[0],xlims[-1]))
771         fig_name = r'{} hist_fits.png'.format(fiber_name)
```

```
772         # plt.savefig(r'{}\{}'.format(path,fig_name),
                bbox_inches='tight')
773         plt.show()
774
775         plt.figure()
776         plt.plot(x_hist,res_gauss)
777         plt.plot(x_hist,res_skew)
778         plt.plot(x_hist,res_gennorm)
779         plt.plot(x_hist,res_laplace)
780         # plt.plot(xlims,np.zeros(len(xlims)),'k')
781         plt.plot(std_bars(data_histfit,y_hist)[0][0],std_bars(
                data_histfit,y_hist)[0][1],'k--')
782         plt.plot(std_bars(data_histfit,y_hist)[1][0],std_bars(
                data_histfit,y_hist)[1][1],'k--')
783         plt.title('Comparison of Fit Residuals for Various
                PDFs for DMT Modulus',fontsize=16)
784         plt.xlabel('Transverse Modulus [GPa]',fontsize=14)
785         # plt.title('Comparison of Fit Residuals for Various
                PDFs of Peak Force',fontsize=16)
786         # plt.xlabel('Peak Force [nN]',fontsize=14)
787         # plt.title('Comparison of Fit Residuals for Various
                PDFs for Indentation',fontsize=16)
788         # plt.xlabel('Indentation Depth [nm]',fontsize=14)
789         # plt.title('Comparison of Fit Residuals for Various
                PDFs for Adhesion',fontsize=16)
790         # plt.xlabel('Adhesion Force [nN]',fontsize=14)
791         plt.legend(['Gaussian','Skew Gaussian','Generalized
                Normal','Laplace'],fontsize=14)
792         plt.xlim((xlims[0],xlims[-1]))
793         fig_name = r'{} hist_fit_res.png'.format(fiber_name)
794         plt.savefig(r'{}\{}'.format(path,fig_name),bbox_inches
                ='tight')
795         plt.show()
796
797 #%% Multiple Gaussian Fitting
798
799 if multi_fit == True:
800
801     data_histfit = DMT_mod_array
802     # data_histfit = E_adh_array
803     # data_histfit = E_matl_array
804
805     fit_bins = int(2*(len(data_histfit))**(1/3))
806     hist_fit = np.histogram(data_histfit,bins=fit_bins)
807
808     x_hist = hist_fit[1][0:-1]
809     y_hist = hist_fit[0]
810
811     # Initial guesses for the parameters to fit:
812     # 3 amplitudes, means and standard deviations plus a
            continuum offset.
813     # guess_glue = [np.max(y_hist)/2,glue_mean,glue_std,np.max
            (y_hist)/2,np.mean(x_hist),np.std(x_hist),0]
814     # guess_gauss = [950,3.5,1.5,1300,9.5,1.5,0]
815     # guess_gauss = [3000,18,2,0,0,0,0]
816     guess_gauss = [0,0,0,np.max(y_hist)/2,np.mean(x_hist),np.
            std(x_hist),0]
```

```
817     guess_skew = [0,0,0,0,np.max(y_hist)/2,np.mean(x_hist),np.
            std(x_hist),0,0]
818
819     # Glue peak check
820     # try:
821     #     popt_glue, pcov = curve_fit(multi_gaussian, x_hist,
            y_hist, guess_glue, bounds=(0, np.inf), maxfev=10000)
822     #     glue1 = gaussian(x_hist, popt_glue[0],popt_glue[1],
            popt_glue[2])
823     #     glue2 = gaussian(x_hist, popt_glue[3],popt_glue[4],
            popt_glue[5])
824
825     #     if popt_glue[0] > popt_glue[3]:
826     #         peak_big = popt_glue[0]
827     #         mean_big = popt_glue[1]
828     #         std_big = popt_glue[2]
829     #         peak_sml = popt_glue[3]
830     #         mean_sml = popt_glue[4]
831     #         std_sml = popt_glue[5]
832     #     else:
833     #         peak_big = popt_glue[3]
834     #         mean_big = popt_glue[4]
835     #         std_big = popt_glue[5]
836     #         peak_sml = popt_glue[0]
837     #         mean_sml = popt_glue[1]
838     #         std_sml = popt_glue[2]
839     #     print('Statistics for the glue check fit')
840     #     print('For Gauss 1, A = ',popt_glue[0],'mean =',
            popt_glue[1],'and std devn =',popt_glue[2])
841     #     print('For Gauss 2, A = ',popt_glue[3],'mean =',
            popt_glue[4],'and std devn =',popt_glue[5])
842     #     res_glue = y_hist - multi_gaussian(x_hist,*popt_glue
            )
843     #     ss_res_glue = np.sum(res_glue**2)
844     #     ss_tot = np.sum((y_hist-np.mean(y_hist))**2)
845     #     r2_glue = 1 - (ss_res_glue / ss_tot)
846     #     print('R^2 for the glue check fit is',r2_glue)
847     #     if np.abs(mean_big-mean_sml)/(2*np.sqrt(std_big*
            std_sml)) <= 1:
848     #         print('This peak is unimodal, by the liklihood
            test for bimodality.')
849     #     else:
850     #         print('This peak is bimodal, by the liklihood
            test for bimodality.')
851     #         D = np.sqrt(2)*(np.abs(mean_big-mean_sml)/np.
            sqrt(std_big**2*std_sml**2))
852     #         print("Ashman's D for this distribution is",D)
853     #         if D > 2:
854     #             print('The peaks can be cleanly seperated.')
855     #         else:
856     #             print('The peaks cannot be cleanly seperated
            .')
857     #         print('The bimodal seperation is',(mean_big-
            mean_sml)/(2*(std_big*std_sml)))
858     #         print('The bimodality amplitude is',(peak_big-
            peak_sml)/peak_big)
859     #         print('The bimodality ratio is',peak_sml/
            peak_big)
```

```
860     # except RuntimeError:
861     #     print('The multi-peak gaussian fit failed to
            converge.')
862
863     # Multi-Gaussian fitter
864     try:
865         popt_gauss, pcov = curve_fit(multi_gaussian, x_hist,
              y_hist, guess_gauss, bounds=(0, np.inf), maxfev
              =10000)
866         gauss1 = gaussian(x_hist, popt_gauss[0],popt_gauss[1],
              popt_gauss[2])
867         gauss2 = gaussian(x_hist, popt_gauss[3],popt_gauss[4],
              popt_gauss[5])
868
869         if popt_gauss[0] > popt_gauss[3]:
870             peak_big = popt_gauss[0]
871             mean_big = popt_gauss[1]
872             std_big = popt_gauss[2]
873             peak_sml = popt_gauss[3]
874             mean_sml = popt_gauss[4]
875             std_sml = popt_gauss[5]
876         else:
877             peak_big = popt_gauss[3]
878             mean_big = popt_gauss[4]
879             std_big = popt_gauss[5]
880             peak_sml = popt_gauss[0]
881             mean_sml = popt_gauss[1]
882             std_sml = popt_gauss[2]
883         print('Statistics for the multi-gaussian fit')
884         print('The deviation from the arithmetic mean for the
              large peak is',np.abs(mean_big-np.mean(data_histfit
              ))/np.mean(data_histfit)*100,'%')
885         if np.abs(mean_big-np.mean(data_histfit)) > np.std(
              data_histfit):
886             print('The fit peak mean deviates from the data
                  mean by more than 1 std devn')
887         print('The change in precision is',(np.std(
              data_histfit)/np.mean(data_histfit) - std_big/
              mean_big)/(np.std(data_histfit)/np.mean(
              data_histfit))*100,'%.')
888         print('For Gauss 1, A = ',popt_gauss[0],'mean =',
              popt_gauss[1],'and std devn =',popt_gauss[2])
889         print('For Gauss 2, A = ',popt_gauss[3],'mean =',
              popt_gauss[4],'and std devn =',popt_gauss[5])
890         print('The fit parameters are',popt_gauss)
891         res_multgauss = y_hist - multi_gaussian(x_hist,*
              popt_gauss)
892         ss_res_multgauss = np.sum(res_multgauss**2)
893         ss_tot = np.sum((y_hist-np.mean(y_hist))**2)
894         r2_multgauss = 1 - (ss_res_multgauss / ss_tot)
895         print('R^2 for the multi-Gaussian fit is',r2_multgauss
              )
896         if np.abs(mean_big-mean_sml)/(2*np.sqrt(std_big*
              std_sml)) <= 1:
897             print('This peak is unimodal, by the liklihood
                  test for bimodality.')
898         else:
```

```
899             print('This peak is bimodal, by the liklihood test
                    for bimodality.')
900             D = np.sqrt(2)*(np.abs(mean_big-mean_sml)/np.sqrt(
                   std_big**2*std_sml**2))
901             print("Ashman's D for this distribution is",D)
902             if D > 2:
903                 print('The peaks can be cleanly seperated.')
904             else:
905                 print('The peaks cannot be cleanly seperated.'
                        )
906             print('The bimodal seperation is',(mean_big-
                   mean_sml)/(2*(std_big*std_sml)))
907             print('The bimodality amplitude is',(peak_big-
                   peak_sml)/peak_big)
908             print('The bimodality ratio is',peak_sml/peak_big)
909     except RuntimeError:
910         print('The multi-peak gaussian fit failed to converge.
                ')
911
912     # Multi-Skew Gaussian Fitter
913     # try:
914     #     popt_skew, pcov = curve_fit(multi_skew, x_hist,
           y_hist, guess_skew, bounds=(0, np.inf), maxfev=10000)
915     #     skew1 = skew(x_hist, popt_skew[0],popt_skew[1],
           popt_skew[2],popt_skew[3])
916     #     skew2 = skew(x_hist, popt_skew[4],popt_skew[5],
           popt_skew[6],popt_skew[7])
917
918     #     if popt_skew[0] > popt_skew[4]:
919     #         peak_big = skew_mode(popt_skew[1],popt_skew[2],
           popt_skew[3])
920     #         mean_big = skew_mean(popt_skew[1],popt_skew[2],
           popt_skew[3])
921     #         std_big = skew_std(popt_skew[2],popt_skew[3])
922     #         peak_sml = skew_mode(popt_skew[5],popt_skew[6],
           popt_skew[7])
923     #         mean_sml = skew_mean(popt_skew[5],popt_skew[6],
           popt_skew[7])
924     #         std_sml = skew_std(popt_skew[6],popt_skew[7])
925     #     else:
926     #         peak_big = skew_mode(popt_skew[5],popt_skew[6],
           popt_skew[7])
927     #         mean_big = skew_mean(popt_skew[5],popt_skew[6],
           popt_skew[7])
928     #         std_big = skew_std(popt_skew[6],popt_skew[7])
929     #         peak_sml = skew_mode(popt_skew[1],popt_skew[2],
           popt_skew[3])
930     #         mean_sml = skew_mean(popt_skew[1],popt_skew[2],
           popt_skew[3])
931     #         std_sml = skew_std(popt_skew[2],popt_skew[3])
932     #     print('Statistics for the multi-skew fit')
933     #     print('The deviation from the arithmetic mean for
           the large peak is',np.abs(mean_big-np.mean(data_histfit
           ))/np.mean(data_histfit)*100,'%')
934     #     if np.abs(mean_big-np.mean(data_histfit)) > np.std(
           data_histfit):
935     #         print('The fit peak mean deviates from the data
           mean by more than 1 std devn')
```

```
936     #     print('The change in precision is',(np.std(
            data_histfit)/np.mean(data_histfit) - std_big/mean_big)
            /(np.std(data_histfit)/np.mean(data_histfit))*100,'%.')
937     #     print('For Skew 1, A = ',skew_mode(popt_skew[1],
            popt_skew[2],popt_skew[3]),'mean =',skew_mean(popt_skew
            [1],popt_skew[2],popt_skew[3]),'and std devn =',
            skew_std(popt_skew[2],popt_skew[3]))
938     #     print('For Skew 2, A = ',skew_mode(popt_skew[5],
            popt_skew[6],popt_skew[7]),'mean =',skew_mean(popt_skew
            [5],popt_skew[6],popt_skew[7]),'and std devn =',
            skew_std(popt_skew[6],popt_skew[7]))
939     #     if np.abs(mean_big-mean_sml)/(2*np.sqrt(std_big*
            std_sml)) <= 1:
940     #         print('This peak is unimodal, by the liklihood
            test for bimodality.')
941     #     else:
942     #         print('This peak is bimodal, by the liklihood
            test for bimodality.')
943     #         D = np.sqrt(2)*(np.abs(mean_big-mean_sml)/np.
            sqrt(std_big**2*std_sml**2))
944     #         print("Ashman's D for this distribution is",D)
945     #         if D > 2:
946     #             print('The peaks can be cleanly seperated.')
947     #         else:
948     #             print('The peaks cannot be cleanly seperated
            .')
949     #         print('The bimodal seperation is',(mean_big-
            mean_sml)/(2*(std_big*std_sml)))
950     #         print('The bimodality amplitude is',(peak_big-
            peak_sml)/peak_big)
951     #         print('The bimodality ratio is',peak_sml/
            peak_big)
952     # except RuntimeError:
953     #     print('The multi-peak skew fit failed to converge.')
954
955     # if verbose_plots == True:
956     if verbose_plots == False:
957
958         # plt.figure()
959         # plt.plot(x_hist, y_hist, '-', linewidth=4, label='
              Data')
960         # plt.plot(x_hist, multi_gaussian(x_hist, *popt_glue),
              'k--', linewidth=2, label='Fit')
961         # plt.plot(x_hist, glue1, 'r--', linewidth=2, label='
              Fit1')
962         # plt.plot(x_hist, glue2, 'g--', linewidth=2, label='
              Fit2')
963         # plt.title('Glue Check Fit, {} {}'.format(fiber_name,
              mult_fit),fontsize=16)
964         # plt.xlabel('Transverse Modulus[GPa]',fontsize=14)
965         # plt.ylabel('# of Measurements',fontsize=14)
966         # plt.xlim([0,30])
967         # plt.legend(fontsize=14)
968         # fig_name = r'{} glue_fit.png'.format(fiber_name)
969         # # plt.savefig(r'{}\{}'.format(path,fig_name),
              bbox_inches='tight')
970         # plt.show()
971
```

```
972             plt.figure()
973             plt.plot(x_hist, y_hist, '-', linewidth=4, label='Data
                    ')
974             plt.plot(x_hist, multi_gaussian(x_hist, *popt_gauss),
                    'k--', linewidth=2, label='Fit')
975             plt.plot(x_hist, gauss1, 'r--', linewidth=2, label='
                    Fit1')
976             plt.plot(x_hist, gauss2, 'g--', linewidth=2, label='
                    Fit2')
977             plt.title('Multiple-Gaussian Fit, {} {}'.format(
                    fiber_name,mult_fit),fontsize=16)
978             plt.xlabel('Transverse Modulus[GPa]',fontsize=14)
979             plt.ylabel('# of Measurements',fontsize=14)
980             plt.xlim([0,30])
981             plt.legend(fontsize=14)
982             fig_name = r'{} mult_gauss_fit.png'.format(fiber_name)
983             # plt.savefig(r'{}\{}'.format(path,fig_name),
                    bbox_inches='tight')
984             plt.show()
985
986             # plt.figure()
987             # plt.plot(x_hist, y_hist, '-', linewidth=4, label='
                    Data')
988             # plt.plot(x_hist, multi_skew(x_hist, *popt_skew), 'k
                    --', linewidth=2, label='Fit')
989             # plt.plot(x_hist, skew1, 'r--', linewidth=2, label='
                    Fit1')
990             # plt.plot(x_hist, skew2, 'g--', linewidth=2, label='
                    Fit2')
991             # plt.title('Multiple-Skew Fit, {} {}'.format(
                    fiber_name,mult_fit),fontsize=16)
992             # plt.xlabel('Transverse Modulus[GPa]',fontsize=14)
993             # plt.ylabel('# of Measurements',fontsize=14)
994             # plt.legend(fontsize=14)
995             # fig_name = r'{} mult_skew_fit.png'.format(fiber_name
                    )
996             # # plt.savefig(r'{}\{}'.format(path,fig_name),
                    bbox_inches='tight')
997             # plt.show()
998
999             # plt.figure()
1000            # plt.plot(x_hist, y_hist-gauss1, 'r', linewidth=4,
                    label='Data-Fit1')
1001            # plt.plot(x_hist, y_hist-gauss2, 'g', linewidth=4,
                    label='Data-Fit2')
1002            # plt.plot(x_hist, y_hist-gauss3, 'b', linewidth=4,
                    label='Data-Fit3')
1003            # plt.plot(x_hist, multi_gaussian(x_hist, *popt), 'k
                    --', linewidth=2, label='Fit')
1004            # plt.legend()
1005            # fig_name = r'{} mult_gauss_fit.png'.format(
                    fiber_name)
1006            # # plt.savefig(r'{}\{}'.format(path,fig_name),
                    bbox_inches='tight')
1007            # plt.show()
1008
1009    #%% Outlier Testing
1010
```

```
1011 if outlier_test == True:
1012
1013     DMT_mean = np.mean(DMT_mod_array)
1014     DMT_std = np.std(DMT_mod_array)
1015     indent_mean = np.mean(indentation)
1016     indent_std = np.std(indentation)
1017     PF_total_mean = np.mean(PF_total)
1018     PF_total_std = np.std(PF_total)
1019     adhes_mean = np.mean(F_adhes)
1020     adhes_std = np.std(F_adhes)
1021
1022     DMT_index = []
1023     indent_index = []
1024     PF_index = []
1025     adhes_index = []
1026
1027     DMT_index.append(np.where(np.logical_or((DMT_mod_array > (
            DMT_mean + out_lim*DMT_std)),(DMT_mod_array < (DMT_mean
             - out_lim*DMT_std)))))
1028     DMT_index = DMT_index[0]
1029     DMT_index = DMT_index[0]
1030
1031     indent_index.append(np.where(np.logical_or((indentation >
            (indent_mean + out_lim*indent_std)),(indentation < (
            indent_mean - out_lim*indent_std)))))
1032     indent_index = indent_index[0]
1033     indent_index = indent_index[0]
1034
1035     PF_index.append(np.where(np.logical_or((PF_total > (
            PF_total_mean + out_lim*PF_total_std)),(PF_total < (
            PF_total_mean - out_lim*PF_total_std)))))
1036     PF_index = PF_index[0]
1037     PF_index = PF_index[0]
1038
1039     adhes_index.append(np.where(np.logical_or((F_adhes > (
            adhes_mean + out_lim*adhes_std)),(F_adhes < (adhes_mean
             - out_lim*adhes_std)))))
1040     adhes_index = adhes_index[0]
1041     adhes_index = adhes_index[0]
1042
1043     indent_PF_counter = 0
1044     indent_adhes_counter = 0
1045     PF_adhes_counter = 0
1046     treble_counter = 0
1047     indent_PF_tuples = []
1048     indent_adhes_tuples = []
1049     PF_adhes_tuples = []
1050     trebles = []
1051
1052     for i in range(len(indent_index)):
1053         for j in range(len(PF_index)):
1054             if indent_index[i] == PF_index[j]:
1055                 indent_PF_tuples.append((i,j,i))
1056                 indent_PF_counter = indent_PF_counter + 1
1057         for k in range(len(adhes_index)):
1058             if indent_index[i] == adhes_index[k]:
1059                 indent_adhes_tuples.append((i,k,i))
```

```
                indent_adhes_counter = indent_adhes_counter +
                    1
    for l in range(len(PF_index)):
        for m in range(len(adhes_index)):
            if PF_index[l] == adhes_index[m]:
                PF_adhes_tuples.append((l,m))
                PF_adhes_counter = PF_adhes_counter + 1
                for n in range(len(indent_index)):
                    if indent_index[n] == PF_index[l]:
                        trebles.append((i,j,m))
                        treble_counter = treble_counter + 1

    PF_outlier_mods = []
    indent_outlier_mods = []
    adhes_outlier_mods = []

    PF_outlier_mods.append(DMT_mod_array[PF_index])
    E_PF_out_hist = np.histogram(PF_outlier_mods,bins = int
       ((2*len(PF_outlier_mods[0])**(1/3))))
    indent_outlier_mods.append(DMT_mod_array[indent_index])
    E_indent_out_hist = np.histogram(indent_outlier_mods,bins
       = int((2*len(indent_outlier_mods[0])**(1/3))))
    adhes_outlier_mods.append(DMT_mod_array[adhes_index])
    E_adhes_out_hist = np.histogram(adhes_outlier_mods,bins =
       int((2*len(adhes_outlier_mods[0])**(1/3))))

    if verbose_calculations == True:
        print('Outliers Counting and Coincidence')
        print('Total Outliers in Data Channel, {}'.format(
           fiber_name))
        print('The number of indentations >{} std devn from
           the mean are'.format(out_lim),len(indent_index),'or
           ',len(indent_index)/len(indentation)*100,'% of all
           measurements.')
        print('The number of PeakForce values >{} std devn
           from the mean are'.format(out_lim),len(PF_index),'
           or',len(PF_index)/len(indentation)*100,'% of all
           measurements.')
        print('The number of adhesion values >{} std devn from
            the mean are'.format(out_lim),len(adhes_index),'or
           ',len(adhes_index)/len(indentation)*100,'% of all
           measurements.')

        print('The number of coincident indent-PF >{} std devn
            outliers is'.format(out_lim),indent_PF_counter,'or
           ',indent_PF_counter/len(indentation)*100,'% of all
           measurements.')
        print('The number of coincident indent-adhes >{} std
           devn outliers is'.format(out_lim),
           indent_adhes_counter,'or',indent_adhes_counter/len(
           indentation)*100,'% of all measurements.')
        print('The number of coincident PF-adhes >{} std devn
           outliers is'.format(out_lim),PF_adhes_counter,'or',
           PF_adhes_counter/len(indentation)*100,'% of all
           measurements.')
        print('The number of coincident indent-PF-adhes >{}
           std devn outliers is'.format(out_lim),
           treble_counter,'or',treble_counter/len(indentation)
```

```
            *100,'% of all measurements.')
1093
1094        if verbose_plots == True:
1095
1096            plt.figure()
1097            plt.plot(indent_index,range(len(indent_index)))
1098            plt.plot(PF_index,range(len(PF_index)))
1099            plt.plot(adhes_index,range(len(adhes_index)))
1100            plt.xlabel('Index Location in Original Data',fontsize
                =20)
1101            plt.xticks(fontsize=16)
1102            plt.yticks(fontsize=16)
1103            plt.ylabel('Outlier #',fontsize=20)
1104            plt.legend(['Indentation','PeakForce','Adhesion'],
                fontsize=20)
1105            plt.title('Comparison of Locations of >{} Std Devn
                Values, {}'.format(out_lim,fiber_name),fontsize=24)
1106            # fig_name = r'{} outlier_index.png'.format(fiber_name
                )
1107            # plt.savefig(r'{}\{}'.format(path,fig_name),
                bbox_inches='tight')
1108            plt.show()
1109
1110            plt.figure()
1111            plt.plot(E_indent_out_hist[1][0:-1],E_indent_out_hist
                [0])
1112            plt.plot(E_adhes_out_hist[1][0:-1],E_adhes_out_hist
                [0])
1113            plt.plot(E_PF_out_hist[1][0:-1],E_PF_out_hist[0])
1114            plt.plot(DMT_mod_hist[1][0:-1],DMT_mod_hist[0])
1115            plt.legend(['Indent','Adhesion','PeakForce','All
                Measurements'],fontsize=20)
1116            plt.title('Tranverse Modulus Distribution with {} Std
                Devn Outliers, {}'.format(out_lim,fiber_name),
                fontsize=24)
1117            plt.xlabel('Transverse Modulus [GPa]',fontsize=20)
1118            plt.xticks(fontsize=16)
1119            plt.yticks(fontsize=16)
1120            # fig_name = r'{} coin_out_dists.png'.format(
                fiber_name)
1121            # plt.savefig(r'{}\{}'.format(path,fig_name),
                bbox_inches='tight')
1122            plt.show
1123
1124        ident_PF_outlier_mods = []
1125        indent_adhes_outlier_mods = []
1126        PF_adhes_outlier_mods = []
1127        treble_outlier_mods = []
1128
1129        for i in range(indent_PF_counter):
1130            ident_PF_outlier_mods.append(DMT_mod_array[i])
1131            # ident_PF_outlier_mods.append(DMT_mod_matldep(
                PF_total[indent_PF_tuples[i][1]],F_adhes[
                indent_PF_tuples[i][0]],np.polyval(poly_pars,
                indentation[indent_PF_tuples[i][0]]),indentation[
                indent_PF_tuples[i][0]],nu_samp))
1132            # ident_PF_outlier_mods.append(DMT_mod_matldep(
                PF_total[indent_PF_tuples[i][1]],F_adhes[
```

```
                indent_PF_tuples[i][0]],power_law(indentation[
                indent_PF_tuples[i][0]],*pars),indentation[
                indent_PF_tuples[i][0]],nu_samp))
1133
1134        for i in range(indent_adhes_counter):
1135            indent_adhes_outlier_mods.append(DMT_mod_array[i])
1136            # indent_adhes_outlier_mods.append(DMT_mod_matldep(
                PF_total[indent_adhes_tuples[i][1]],F_adhes[
                indent_adhes_tuples[i][0]],np.polyval(poly_pars,
                indentation[indent_adhes_tuples[i][0]]),indentation
                [indent_adhes_tuples[i][0]],nu_samp))
1137            # indent_adhes_outlier_mods.append(DMT_mod_matldep(
                PF_total[indent_adhes_tuples[i][1]],F_adhes[
                indent_adhes_tuples[i][0]],power_law(indentation[
                indent_adhes_tuples[i][0]],*pars),indentation[
                indent_adhes_tuples[i][0]],nu_samp))
1138
1139        for i in range(PF_adhes_counter):
1140            PF_adhes_outlier_mods.append(DMT_mod_array[i])
1141            # PF_adhes_outlier_mods.append(DMT_mod_matldep(
                PF_total[PF_adhes_tuples[i][1]],F_adhes[
                PF_adhes_tuples[i][0]],np.polyval(poly_pars,
                indentation[PF_adhes_tuples[i][0]]),indentation[
                PF_adhes_tuples[i][0]],nu_samp))
1142            # PF_adhes_outlier_mods.append(DMT_mod_matldep(
                PF_total[PF_adhes_tuples[i][1]],F_adhes[
                PF_adhes_tuples[i][0]],power_law(indentation[
                PF_adhes_tuples[i][0]],*pars),indentation[
                PF_adhes_tuples[i][0]],nu_samp))
1143
1144        for i in range(treble_counter):
1145            treble_outlier_mods.append(DMT_mod_array[i])
1146            # treble_outlier_mods.append(DMT_mod_matldep(PF_total[
                trebles[i][1]],F_adhes[trebles[i][2]],np.polyval(
                poly_pars,indentation[trebles[i][0]]),indentation[
                trebles[i][0]],nu_samp))
1147            # treble_outlier_mods.append(DMT_mod_matldep(PF_total[
                trebles[i][1]],F_adhes[trebles[i][2]],power_law(
                indentation[trebles[i][0]],*pars),indentation[
                trebles[i][0]],nu_samp))
1148
1149        if verbose_plots == True:
1150
1151            numbins_indent_PF_out = int((2*indent_PF_counter)
                **(1/3))
1152            numbins_indent_adhes = int((2*indent_adhes_counter)
                **(1/3))
1153            numbins_PF_adhes = int((2*PF_adhes_counter)**(1/3))
1154            numbins_treble = int((2*treble_counter)**(1/3))
1155
1156            E_indent_PF_out_hist = np.histogram(
                ident_PF_outlier_mods,bins=numbins_indent_PF_out)
1157            E_indent_adhes_out_hist = np.histogram(
                indent_adhes_outlier_mods,bins=numbins_indent_adhes
                )
1158            E_PF_adhes_out_hist = np.histogram(
                PF_adhes_outlier_mods,bins=numbins_PF_adhes)
```

```
1159          E_treble_out_hist = np.histogram(treble_outlier_mods,
                 bins=numbins_treble)
1160
1161          plt.figure()
1162          plt.plot(E_indent_PF_out_hist[1][0:-1],
                 E_indent_PF_out_hist[0])
1163          plt.plot(E_indent_adhes_out_hist[1][0:-1],
                 E_indent_adhes_out_hist[0])
1164          plt.plot(E_PF_adhes_out_hist[1][0:-1],
                 E_PF_adhes_out_hist[0])
1165          plt.plot(E_treble_out_hist[1][0:-1],E_treble_out_hist
                 [0])
1166          plt.plot(DMT_mod_hist[1][0:-1],DMT_mod_hist[0])
1167          plt.legend(['Indent-PF','Indent-Adhes','PF-Adhes','
                 Triple Outliers','All Measurements'],fontsize=20)
1168          plt.title('Tranverse Modulus Distribution with {} Std
                 Devn Outliers, {}'.format(out_lim,fiber_name),
                 fontsize=24)
1169          plt.xlabel('Transverse Modulus [GPa]',fontsize=20)
1170          plt.xticks(fontsize=16)
1171          # fig_name = r'{} coin_out_dists.png'.format(
                 fiber_name)
1172          # plt.savefig(r'{}\{}'.format(path,fig_name),
                 bbox_inches='tight')
1173          plt.show
1174
1175  #%% Adhesion Force-Indentation-Tip Shape Functional
        Interaction
1176
1177  if adhesion_func == True:
1178      PF = PF_total
1179      adh = F_adhes
1180      depth = indentation
1181      tip_rad = np.polyval(poly_pars,indentation)
1182      tip_surf = 4*np.pi*tip_rad**2
1183
1184      length = len(adh)
1185
1186      dep_tip_tuples = []
1187      dep_adh_tuples = []
1188      dep_PF_tuples = []
1189      PF_adh_tuples = []
1190      PF_tip_tuples = []
1191      adh_tip_tuples = []
1192      DMT_dep_tuples = []
1193      DMT_PF_tuples = []
1194      DMT_adh_tuples = []
1195      DMT_tip_tuples = []
1196
1197      for i in range(length):
1198          dep_tip_tuples.append((depth[i],tip_rad[i]))
1199          dep_adh_tuples.append((depth[i],adh[i]))
1200          PF_adh_tuples.append((PF[i],adh[i]))
1201          PF_tip_tuples.append((PF[i],tip_rad[i]))
1202          adh_tip_tuples.append((tip_rad[i],adh[i]))
1203          dep_PF_tuples.append((depth[i],PF[i]))
1204          DMT_dep_tuples.append((DMT_mod_array[i],depth[i]))
1205          DMT_PF_tuples.append((DMT_mod_array[i],PF[i]))
```

```
1206            DMT_adh_tuples.append((DMT_mod_array[i],adh[i]))
1207            DMT_tip_tuples.append((DMT_mod_array[i],tip_rad[i]))
1208
1209        dep_tip_tuples.sort(key=lambda tup: tup[0])
1210        dep_adh_tuples.sort(key=lambda tup: tup[0])
1211        dep_PF_tuples.sort(key=lambda tup: tup[0])
1212        PF_adh_tuples.sort(key=lambda tup: tup[0])
1213        PF_tip_tuples.sort(key=lambda tup: tup[0])
1214        adh_tip_tuples.sort(key=lambda tup: tup[0])
1215        DMT_dep_tuples.sort(key=lambda tup: tup[0])
1216        DMT_PF_tuples.sort(key=lambda tup: tup[0])
1217        DMT_adh_tuples.sort(key=lambda tup: tup[0])
1218        DMT_tip_tuples.sort(key=lambda tup: tup[0])
1219
1220        dep_tip_plot = []
1221        tip_dep_plot = []
1222        adh_dep_plot = []
1223        dep_adh_plot = []
1224        tip_adh_plot = []
1225        adh_tip_plot = []
1226        PF_adh_plot = []
1227        adh_PF_plot = []
1228        depth_PF_plot = []
1229        PF_depth_plot = []
1230        PF_tip_plot = []
1231        tip_PF_plot = []
1232        DMT_dep_plot = []
1233        dep_DMT_plot = []
1234        DMT_PF_plot = []
1235        PF_DMT_plot = []
1236        DMT_adh_plot = []
1237        adh_DMT_plot = []
1238        DMT_tip_plot = []
1239        tip_DMT_plot = []
1240
1241        for i  in range(length):
1242            adh_tip_plot.append(adh_tip_tuples[i][0])
1243            tip_adh_plot.append(adh_tip_tuples[i][1])
1244            dep_tip_plot.append(dep_tip_tuples[i][0])
1245            tip_dep_plot.append(dep_tip_tuples[i][1])
1246            dep_adh_plot.append(dep_adh_tuples[i][0])
1247            adh_dep_plot.append(dep_adh_tuples[i][1])
1248            PF_adh_plot.append(PF_adh_tuples[i][0])
1249            adh_PF_plot.append(PF_adh_tuples[i][1])
1250            depth_PF_plot.append(dep_PF_tuples[i][0])
1251            PF_depth_plot.append(dep_PF_tuples[i][1])
1252            PF_tip_plot.append(PF_tip_tuples[i][0])
1253            tip_PF_plot.append(PF_tip_tuples[i][1])
1254            DMT_dep_plot.append(DMT_dep_tuples[i][0])
1255            dep_DMT_plot.append(DMT_dep_tuples[i][1])
1256            DMT_PF_plot.append(DMT_PF_tuples[i][0])
1257            PF_DMT_plot.append(DMT_PF_tuples[i][1])
1258            DMT_adh_plot.append(DMT_adh_tuples[i][0])
1259            adh_DMT_plot.append(DMT_adh_tuples[i][1])
1260            DMT_tip_plot.append(DMT_tip_tuples[i][0])
1261            tip_DMT_plot.append(DMT_tip_tuples[i][1])
1262
1263        if verbose_plots == True:
```

```
1264
1265        # Generate 2D histogram heatmap of indentation depth
               and tip radius
1266        plt.figure()
1267        N_numbers = len(dep_tip_plot) + len(tip_dep_plot)
1268        N_bins = int((2*N_numbers)**(1/3))
1269        x, y = dep_tip_plot, tip_dep_plot
1270        plt.hist2d(x, y, bins=N_bins, density=False, cmap='
               Greys')
1271        hist_test = np.histogram2d(x, y,bins=N_bins)
1272        cb = plt.colorbar()
1273        cb.set_label('Measurements in Bin',fontsize=16)
1274        plt.clim(np.mean(hist_test[0])+np.std(hist_test[0]))
1275        plt.title('2D Histogram of Indentation and Tip Radius
               Measurements, {}'.format(fiber_name),fontsize=24)
1276        plt.xlabel('Indentation [nm]',fontsize=20)
1277        plt.xticks(fontsize=16)
1278        plt.ylabel('Tip Radius [nm]',fontsize=20)
1279        plt.yticks(fontsize=16)
1280        fig_name = r'{} indent-adhes_heatmap.png'.format(
               fiber_name)
1281        # plt.savefig(r'{}\{}'.format(path,fig_name),
               bbox_inches='tight')
1282        plt.show()
1283
1284        # Generate 2D histogram heatmap  of indentation depth
               and adhesion force
1285        plt.figure()
1286        N_numbers = len(dep_adh_plot) + len(adh_dep_plot)
1287        N_bins = int((2*N_numbers)**(1/3))
1288        x, y = dep_adh_plot, adh_dep_plot
1289        plt.hist2d(x, y, bins=N_bins, density=False, cmap='
               Greys')
1290        hist_test = np.histogram2d(x, y,bins=N_bins)
1291        cb = plt.colorbar()
1292        cb.set_label('Measurements in Bin',fontsize=16)
1293        plt.clim(np.mean(hist_test[0])+np.std(hist_test[0]))
1294        plt.title('2D Histogram of Indentation and Adhesion
               Measurements, {}'.format(fiber_name),fontsize=24)
1295        plt.xlabel('Indentation [nm]',fontsize=20)
1296        plt.xticks(fontsize=16)
1297        plt.ylabel('Adhesion Force [nN]',fontsize=20)
1298        plt.yticks(fontsize=16)
1299        fig_name = r'{} indent-adhes_heatmap.png'.format(
               fiber_name)
1300        # plt.savefig(r'{}\{}'.format(path,fig_name),
               bbox_inches='tight')
1301        plt.show()
1302
1303        # Generate 2D histogram heatmap of indentation depth
               and Peak force
1304        plt.figure()
1305        N_numbers = len(depth_PF_plot) + len(PF_depth_plot)
1306        N_bins = int((2*N_numbers)**(1/3))
1307        x, y = depth_PF_plot, PF_depth_plot
1308        plt.hist2d(x, y, bins=N_bins, density=False, cmap='
               Greys')
1309        hist_test = np.histogram2d(x, y,bins=N_bins)
```

```
1310            cb = plt.colorbar()
1311            cb.set_label('Measurements in Bin',fontsize=16)
1312            plt.clim(np.mean(hist_test[0])+np.std(hist_test[0]))
1313            plt.title('2D Histogram of Indentation and Peak Force
                   data points, {}'.format(fiber_name),fontsize=24)
1314            plt.xlabel('Indentation [nm]',fontsize=20)
1315            plt.xticks(fontsize=16)
1316            plt.ylabel('Peak Force [nN]',fontsize=20)
1317            plt.yticks(fontsize=16)
1318            fig_name = r'{} PF-adhes_heatmap.png'.format(
                   fiber_name)
1319            # plt.savefig(r'{}\{}'.format(path,fig_name),
                   bbox_inches='tight')
1320            plt.show()
1321
1322            # Generate 2D histogram heatmap  of Peak Force and
                   adhesion force
1323            plt.figure()
1324            N_numbers = len(PF_adh_plot) + len(adh_PF_plot)
1325            N_bins = int((2*N_numbers)**(1/3))
1326            x, y = PF_adh_plot, adh_PF_plot
1327            plt.hist2d(x, y, bins=N_bins, density=False, cmap='
                   Greys')
1328            hist_test = np.histogram2d(x, y,bins=N_bins)
1329            cb = plt.colorbar()
1330            cb.set_label('Measurements in Bin',fontsize=16)
1331            plt.clim(np.mean(hist_test[0])+np.std(hist_test[0]))
1332            plt.title('2D Histogram of Peak Force and Adhesion
                   data points, {}'.format(fiber_name),fontsize=24)
1333            plt.xlabel('Peak Force [nN]',fontsize=20)
1334            plt.xticks(fontsize=16)
1335            plt.ylabel('Adhesion Force [nN]',fontsize=20)
1336            plt.yticks(fontsize=16)
1337            fig_name = r'{} PF-adhes_heatmap.png'.format(
                   fiber_name)
1338            # plt.savefig(r'{}\{}'.format(path,fig_name),
                   bbox_inches='tight')
1339            plt.show()
1340
1341            # Generate 2D histogram heatmap of peak force and tip
                   radius
1342            plt.figure()
1343            N_numbers = len(PF_tip_plot) + len(tip_PF_plot)
1344            N_bins = int((2*N_numbers)**(1/3))
1345            x, y = PF_tip_plot, tip_PF_plot
1346            plt.hist2d(x, y, bins=N_bins, density=False, cmap='
                   Greys')
1347            hist_test = np.histogram2d(x, y,bins=N_bins)
1348            cb = plt.colorbar()
1349            cb.set_label('Measurements in Bin',fontsize=16)
1350            plt.clim(np.mean(hist_test[0])+np.std(hist_test[0]))
1351            plt.title('2D Histogram of Peak Force and Tip Radius
                   Measurements, {}'.format(fiber_name),fontsize=24)
1352            plt.xlabel('Peak Force [nN]',fontsize=20)
1353            plt.xticks(fontsize=16)
1354            plt.ylabel('Tip Radius [nm]',fontsize=20)
1355            plt.yticks(fontsize=16)
```

```
1356        fig_name = r'{} indent-adhes_heatmap.png'.format(
               fiber_name)
1357        # plt.savefig(r'{}\{}'.format(path,fig_name),
               bbox_inches='tight')
1358        plt.show()
1359
1360        # Generate 2D histogram heatmap of adhesion force and
               tip radius
1361        plt.figure()
1362        N_numbers = len(adh_tip_plot) + len(tip_adh_plot)
1363        N_bins = int((2*N_numbers)**(1/3))
1364        x, y = adh_tip_plot, tip_adh_plot
1365        plt.hist2d(x, y, bins=N_bins, density=False, cmap='
               Greys')
1366        hist_test = np.histogram2d(x, y,bins=N_bins)
1367        cb = plt.colorbar()
1368        cb.set_label('Measurements in Bin',fontsize=16)
1369        plt.clim(np.mean(hist_test[0])+np.std(hist_test[0]))
1370        plt.title('2D Histogram of Adhesion Force and Tip
               Radius Measurements, {}'.format(fiber_name),
               fontsize=24)
1371        plt.xlabel('Adhesion Force [nN]',fontsize=20)
1372        plt.xticks(fontsize=16)
1373        plt.ylabel('Tip Radius [nm]',fontsize=20)
1374        plt.yticks(fontsize=16)
1375        fig_name = r'{} indent-adhes_heatmap.png'.format(
               fiber_name)
1376        # plt.savefig(r'{}\{}'.format(path,fig_name),
               bbox_inches='tight')
1377        plt.show()
1378
1379        # Generate 2D histogram heatmap of DMT Modulus and
               Peak force
1380        plt.figure()
1381        N_numbers = len(DMT_PF_plot) + len(PF_DMT_plot)
1382        N_bins = int((2*N_numbers)**(1/3))
1383        x, y = DMT_PF_plot, PF_DMT_plot
1384        plt.hist2d(x, y, bins=N_bins, density=False, cmap='
               Greys')
1385        hist_test = np.histogram2d(x, y,bins=N_bins)
1386        cb = plt.colorbar()
1387        cb.set_label('Measurements in Bin',fontsize=16)
1388        plt.clim(np.mean(hist_test[0])+np.std(hist_test[0]))
1389        plt.title('2D Histogram of DMT Modulus and Peak Force
               data points, {}'.format(fiber_name),fontsize=24)
1390        plt.xlabel('DMT_modulus [GPa]',fontsize=20)
1391        plt.xticks(fontsize=16)
1392        plt.ylabel('Peak Force [nN]',fontsize=20)
1393        plt.yticks(fontsize=16)
1394        fig_name = r'{} PF-adhes_heatmap.png'.format(
               fiber_name)
1395        # plt.savefig(r'{}\{}'.format(path,fig_name),
               bbox_inches='tight')
1396        plt.show()
1397
1398        # Generate 2D histogram heatmap of DMT Modulus and
               adhesion force
1399        plt.figure()
```

```
1400        N_numbers = len(DMT_adh_plot[0:-2]) + len(adh_DMT_plot
                )
1401        N_bins = int((2*N_numbers)**(1/3))
1402        x, y = DMT_adh_plot, adh_DMT_plot
1403        plt.hist2d(x, y, bins=N_bins, density=False, cmap='
                Greys')
1404        hist_test = np.histogram2d(x, y,bins=N_bins)
1405        cb = plt.colorbar()
1406        cb.set_label('Measurements in Bin',fontsize=16)
1407        plt.clim(np.mean(hist_test[0])+np.std(hist_test[0]))
1408        plt.title('2D Histogram of DMT Modulus and Adhesion
                Force data points, {}'.format(fiber_name),fontsize
                =24)
1409        plt.xlabel('DMT_modulus [GPa]',fontsize=20)
1410        plt.xticks(fontsize=16)
1411        plt.ylabel('Adhesion Force [nN]',fontsize=20)
1412        plt.yticks(fontsize=16)
1413        fig_name = r'{} PF-adhes_heatmap.png'.format(
                fiber_name)
1414        # plt.savefig(r'{}\{}'.format(path,fig_name),
                bbox_inches='tight')
1415        plt.show()
1416
1417        # Generate 2D histogram heatmap of DMT Modulus and
                indentation
1418        plt.figure()
1419        N_numbers = len(DMT_dep_plot) + len(dep_DMT_plot)
1420        N_bins = int((2*N_numbers)**(1/3))
1421        x, y = DMT_dep_plot, dep_DMT_plot
1422        plt.hist2d(x, y, bins=N_bins, density=False, cmap='
                Greys')
1423        hist_test = np.histogram2d(x, y,bins=N_bins)
1424        cb = plt.colorbar()
1425        cb.set_label('Measurements in Bin',fontsize=16)
1426        plt.clim(np.mean(hist_test[0])+np.std(hist_test[0]))
1427        plt.title('2D histogram of DMT Modulus and Indentation
                 data points, {}'.format(fiber_name),fontsize=24)
1428        plt.xlabel('DMT_modulus [GPa]',fontsize=20)
1429        plt.xticks(fontsize=16)
1430        plt.ylabel('Indentation Depth [nm]',fontsize=20)
1431        plt.yticks(fontsize=16)
1432        fig_name = r'{} PF-adhes_heatmap.png'.format(
                fiber_name)
1433        # plt.savefig(r'{}\{}'.format(path,fig_name),
                bbox_inches='tight')
1434        plt.show()
1435
1436        # Generate 2D histogram heatmap of DMT Modulus and tip
                 radius
1437        plt.figure()
1438        N_numbers = len(DMT_tip_plot) + len(tip_DMT_plot)
1439        N_bins = int((2*N_numbers)**(1/3))
1440        x, y = DMT_tip_plot, tip_DMT_plot
1441        plt.hist2d(x, y, bins=N_bins, density=False, cmap='
                Greys')
1442        hist_test = np.histogram2d(x, y,bins=N_bins)
1443        cb = plt.colorbar()
1444        cb.set_label('Measurements in Bin',fontsize=16)
```

```
1445            plt.clim(np.mean(hist_test[0])+np.std(hist_test[0]))
1446            plt.title('2D histogram of DMT Modulus and Tip Radius
                   data points, {}'.format(fiber_name),fontsize=24)
1447            plt.xlabel('DMT_modulus [GPa]',fontsize=20)
1448            plt.xticks(fontsize=16)
1449            plt.ylabel('Tip Radius [nm]',fontsize=20)
1450            plt.yticks(fontsize=16)
1451            fig_name = r'{} PF-adhes_heatmap.png'.format(
                   fiber_name)
1452            # plt.savefig(r'{}\{}'.format(path,fig_name),
                   bbox_inches='tight')
1453            plt.show()
1454
1455            # plt.figure()
1456            # plt.scatter(tip_rad,adh)
1457            # plt.title('Tip Radius vs. Adhesion Force')
1458            # plt.xlabel('Tip Radius [nm]')
1459            # plt.ylabel('Adhesion Force [nN]')
1460            # plt.show()
1461
1462            # plt.figure()
1463            # plt.scatter(tip_surf,adh)
1464            # plt.title('Contact Surface Area vs. Adhesion Force')
1465            # plt.xlabel('Contact Area [nm^2]')
1466            # plt.ylabel('Adhesion Force [nN]')
1467            # plt.show()
1468
1469    #%% Compare Approximate Tip Volume to Sphere of same radius
1470
1471    if tip_vol_compare == True:
1472
1473        front_angle = (15.0,13.0,17.0)
1474        side_angle = (17.5,15.5,19.5)
1475        back_angle = (25.0,23.0,27.0)
1476
1477        height = np.linspace(0.5,20,40)
1478        radius = yfit
1479        tip_rad_poly = np.polyval(poly_pars,height)
1480
1481        tip_vol_nom = tip_geo(height,side_angle,front_angle,
               back_angle)[0][0]
1482        tip_vol_min = tip_geo(height,side_angle,front_angle,
               back_angle)[0][1]
1483        tip_vol_max = tip_geo(height,side_angle,front_angle,
               back_angle)[0][2]
1484        tip_SA_nom = tip_geo(height,side_angle,front_angle,
               back_angle)[1][0]
1485        tip_SA_min = tip_geo(height,side_angle,front_angle,
               back_angle)[1][1]
1486        tip_SA_max = tip_geo(height,side_angle,front_angle,
               back_angle)[1][2]
1487
1488        indent_vol = sph_indent_geo(height,radius)[0]
1489        indent_SA = sph_indent_geo(height,radius)[1]
1490
1491        if verbose_plots == True:
1492
1493            plt.figure()
```

```
1494            plt.plot(height,np.sqrt(tip_SA_nom/(4*np.pi)),label='
                   Nominal Tip')
1495            plt.plot(height,np.sqrt(tip_SA_min/(4*np.pi)),label='
                   Min Tip')
1496            plt.plot(height,np.sqrt(tip_SA_max/(4*np.pi)),label='
                   Max Tip')
1497            plt.plot(height,np.sqrt(indent_SA/(4*np.pi)),label='
                   Hemispherical Tip')
1498            plt.plot(x,yfit,'--k',label='Actual Tip')
1499            plt.title('Comparison of Measured Tip Radius to
                   Geometric Tip Radius',fontsize=24)
1500            plt.xlabel('Indentation [nm]',fontsize=20)
1501            plt.xticks(fontsize=16)
1502            plt.ylabel('Tip Radius [nm]',fontsize=20)
1503            plt.yticks(fontsize=16)
1504            plt.legend(fontsize=20)
1505            fig_name = r'{} tip_surfarea.png'.format(fiber_name)
1506            # plt.savefig(r'{}\{}'.format(path,fig_name),
                   bbox_inches='tight')
1507            plt.show()
1508
1509            plt.figure()
1510            plt.semilogy(height,tip_vol_nom,label='Nominal Tip')
1511            plt.semilogy(height,tip_vol_min,label='Min Tip')
1512            plt.semilogy(height,tip_vol_max,label='Max Tip')
1513            plt.semilogy(height,indent_vol,label='Hemispherical
                   Tip')
1514            plt.semilogy()
1515            plt.title('Comparison of Tip and Sphere Volume',
                   fontsize=24)
1516            plt.xlim((0,10))
1517            plt.ylim((0,1000))
1518            plt.xlabel('Indentation [nm]',fontsize=20)
1519            plt.xticks(fontsize=16)
1520            plt.ylabel('Volume [nm^3]',fontsize=20)
1521            plt.yticks(fontsize=16)
1522            plt.legend(fontsize=20)
1523            fig_name = r'{} tip_vol.png'.format(fiber_name)
1524            # plt.savefig(r'{}\{}'.format(path,fig_name),
                   bbox_inches='tight')
1525            plt.show()
1526
1527  #%% Poisson Ratio Study
1528
1529  if poisson_study == True:
1530
1531      nu_study = np.array([0.1,0.2,0.3,0.4,0.5])
1532
1533      numbins = int(2*(len(DMT_mod_array))**(1/3))
1534
1535      E_nu1 = DMT_mod_adhes(PF_total,F_adhes,np.polyval(
                   poly_pars,indentation),indentation,nu_study[0])
1536      E_nu1_hist = np.histogram(E_nu1,bins=numbins)
1537
1538      E_nu2 = DMT_mod_adhes(PF_total,F_adhes,np.polyval(
                   poly_pars,indentation),indentation,nu_study[1])
1539      E_nu2_hist = np.histogram(E_nu2,bins=numbins)
1540
```

```
1541        E_nu3 = DMT_mod_adhes(PF_total,F_adhes,np.polyval(
               poly_pars,indentation),indentation,nu_study[2])
1542        E_nu3_hist = np.histogram(E_nu3,bins=numbins)
1543
1544        E_nu4 = DMT_mod_adhes(PF_total,F_adhes,np.polyval(
               poly_pars,indentation),indentation,nu_study[3])
1545        E_nu4_hist = np.histogram(E_nu4,bins=numbins)
1546
1547        E_nu5 = DMT_mod_adhes(PF_total,F_adhes,np.polyval(
               poly_pars,indentation),indentation,nu_study[4])
1548        E_nu5_hist = np.histogram(E_nu5,bins=numbins)
1549
1550        diff2 = np.mean(E_nu3)-np.mean(E_nu2)
1551        rel_diff2 = diff2/np.mean(E_nu3)*100
1552        num_devs2 = np.abs(diff2/np.std(E_nu3))
1553
1554        diff_std2 = np.std(E_nu3)-np.std(E_nu2)
1555        rel_diff_std2 = diff_std2/np.std(E_nu3)*100
1556
1557        diff4 = np.mean(E_nu3)-np.mean(E_nu4)
1558        rel_diff4 = diff4/np.mean(E_nu3)*100
1559        num_devs4 = np.abs(diff4/np.std(E_nu3))
1560
1561        diff_std4 = np.std(E_nu3)-np.std(E_nu4)
1562        rel_diff_std4 = diff_std4/np.std(E_nu3)*100
1563
1564        diff_tot = np.abs(rel_diff_std2) + np.abs(rel_diff_std4)
1565
1566        if verbose_calculations == True:
1567            print("Poisson's Study")
1568            print('The mean calculated modulus w/ nu = {} is'.
                   format(nu_study[0]),np.mean(E_nu1),'GPa w/ std dev'
                   ,np.std(E_nu1),'GPa.')
1569            print('The mean calculated modulus w/ nu = {} is'.
                   format(nu_study[1]),np.mean(E_nu2),'GPa w/ std dev'
                   ,np.std(E_nu2),'GPa.')
1570            print('The mean calculated modulus w/ nu = {} is'.
                   format(nu_study[2]),np.mean(E_nu3),'GPa w/ std dev'
                   ,np.std(E_nu3),'GPa.')
1571            print('The mean calculated modulus w/ nu = {} is'.
                   format(nu_study[3]),np.mean(E_nu4),'GPa w/ std dev'
                   ,np.std(E_nu4),'GPa.')
1572            print('The mean calculated modulus w/ nu = {} is'.
                   format(nu_study[4]),np.mean(E_nu5),'GPa w/ std dev'
                   ,np.std(E_nu5),'GPa.')
1573            print('Deviations from base case')
1574            print('The relative deviation from the base case (nu
                   ={}, DMT model) for nu={} is'.format(nu_study[2],
                   nu_study[1]))
1575            print(truncate(rel_diff2,3),'%, equivalent to',
                   truncate(num_devs2,3),'base case standard
                   deviations.')
1576            print('The relative deviation from the base case (nu
                   ={}, DMT model) for nu={} is'.format(nu_study[2],
                   nu_study[3]))
1577            print(truncate(rel_diff4,3),'%, equivalent to',
                   truncate(num_devs4,3),'base case standard
                   deviations.')
```

```
1578          print('The total modulus span between the mean moduli
                 for nu={} & {} is'.format(nu_study[1],nu_study[3]))
1579          print(truncate(diff_tot*np.mean(E_nu3)/100,3),'GPa, or
                 ',truncate(diff_tot,3),'% of the base case (nu={},
                 DMT model) mean.'.format(nu_study[2]))
1580
1581      if verbose_plots == True:
1582          plt.figure()
1583          # plt.plot(E_nu1_hist[1][0:-1],E_nu1_hist[0])
1584          plt.plot(E_nu2_hist[1][0:-1],E_nu2_hist[0])
1585          plt.plot(E_nu3_hist[1][0:-1],E_nu3_hist[0],lw=5)
1586          plt.plot(E_nu4_hist[1][0:-1],E_nu4_hist[0])
1587          # plt.plot(E_nu5_hist[1][0:-1],E_nu5_hist[0])
1588          plt.legend(['nu={}'.format(nu_study[1]),'nu={}'.format
                 (nu_study[2]),'nu={}'.format(nu_study[3])],fontsize
                 =20)    #,'nu={}'.format(nu_study[3]),'nu={}'.
                 format(nu_study[4])],fontsize=14)
1589          plt.title('Comparison of Tranverse Modulus Change due
                 to Poisson Ratio=0.2-0.4, {}'.format(fiber_name),
                 fontsize=24)
1590          plt.xlabel('Transverse Modulus [GPa]',fontsize=20)
1591          plt.xticks(fontsize=16)
1592          plt.yticks(fontsize=16)
1593          fig_name = r'{} poisson.png'.format(fiber_name)
1594          # plt.savefig(r'{}\{}'.format(path,fig_name),
                 bbox_inches='tight')
1595          plt.show
1596
1597  #%% Build histograms to compare different data channel
        distributions
1598
1599  if channel_hists == True:
1600
1601      numbins = int(2*(len(DMT_mod_array))**(1/3))
1602
1603      err_DMT = DMT_err(PF_total,F_adhes,np.polyval(poly_pars,
             indentation),indentation)
1604      print('The expected DMT modulus error is',err_DMT/np.mean(
             DMT_mod_array)*100,'%')
1605
1606      indent_mean = np.mean(data_image[indent])
1607      indent_std = np.std(data_image[indent])
1608      print('The indentation error is',indent_std/indent_mean
             *100,'%.')
1609
1610      adhes_mean = np.mean(data_image[adhes])
1611      adhes_std = np.std(data_image[adhes])
1612      print('The adhesion error is',adhes_std/adhes_mean*100,'%.
             ')
1613
1614      pf_mean = np.mean(PF_total)
1615      pf_std = np.std(PF_total)
1616      print('The peak force error is',pf_std/pf_mean*100,'%.')
1617
1618      tiprad_mean = np.mean(np.polyval(poly_pars,indentation))
1619      tiprad_std = np.std(np.polyval(poly_pars,indentation))
1620      print('The tip radius error is',tiprad_std/tiprad_mean
             *100,'%.')
```

```
1621
1622        # print('The Spearman R for indentation-modulus is',
               spearmanr(data_image[indent],DMT_mod_array,nan_policy='
               omit')[0],'.')
1623        # print('The Spearman R for adhesion-modulus is',spearmanr
               (data_image[adhes],DMT_mod_array,nan_policy='omit')
               [0],'.')
1624        # print('The Spearman R for peak force-modulus is',
               spearmanr(PF_total,DMT_mod_array,nan_policy='omit')
               [0],'.')
1625        # print('The Spearman R for tip radius-modulus is',
               spearmanr(np.polyval(poly_pars,indentation),
               DMT_mod_array,nan_policy='omit')[0],'.')
1626
1627        # print('The Pearson R for indentation-modulus is',
               pearsonr(data_image[indent],DMT_mod_array)[0],'.')
1628        # print('The Pearson R for adhesion-modulus is',pearsonr(
               data_image[adhes],DMT_mod_array)[0],'.')
1629        # print('The Pearson R for peak force-modulus is',pearsonr
               (PF_total,DMT_mod_array)[0],'.')
1630        # print('The Pearson R for tip radius-modulus is',pearsonr
               (np.polyval(poly_pars,indentation),DMT_mod_array)
               [0],'.')
1631
1632        indent_hist = np.histogram(indentation,bins=numbins)
1633        # tiprad_hist = np.histogram(power_law(data_image[indent
               ],*pars),bins=numbins)
1634        # pferr_hist = np.histogram(data_image[pf_error],bins=
               numbins)
1635        adhes_hist = np.histogram(data_image[adhes],bins=numbins)
1636        tipforce_hist = np.histogram(PF_total,bins=numbins)
1637        tiprad_hist = np.histogram(np.polyval(poly_pars,
               indentation),bins=numbins)
1638
1639        if verbose_plots == True:
1640
1641            plt.figure()
1642            plt.plot(indent_hist[1][0:-1],indent_hist[0])
1643            plt.title('Indentation Histogram, {}'.format(
                   fiber_name),fontsize=24)
1644            plt.xlabel('Indentation Depth [nm]',fontsize=20)
1645            plt.ylabel('# of Indentations',fontsize=20)
1646            plt.xticks(fontsize=16)
1647            plt.yticks(fontsize=16)
1648            fig_name = r'{} indent_hist.png'.format(fiber_name)
1649            # plt.savefig(r'{}\{}'.format(path,fig_name),
                   bbox_inches='tight')
1650            plt.show
1651
1652            plt.figure()
1653            plt.plot(adhes_hist[1][0:-1],adhes_hist[0])
1654            plt.title('Adhesion Histogram, {}'.format(fiber_name),
                   fontsize=24)
1655            plt.xlabel('Adhesion Force [nN]',fontsize=20)
1656            plt.ylabel('# of Indentations',fontsize=20)
1657            plt.xticks(fontsize=16)
1658            plt.yticks(fontsize=16)
1659            fig_name = r'{} adhes_hist.png'.format(fiber_name)
```

```
1660          # plt.savefig(r'{}\{}'.format(path,fig_name),
                 bbox_inches='tight')
1661          plt.show
1662
1663          plt.figure()
1664          plt.plot(tipforce_hist[1][0:-1],tipforce_hist[0])
1665          plt.title('PeakForce Distribution, {}'.format(
                 fiber_name),fontsize=24)
1666          plt.xlabel('PeakForce [nN]',fontsize=20)
1667          plt.ylabel('# of Indentations',fontsize=20)
1668          plt.xticks(fontsize=16)
1669          plt.yticks(fontsize=16)
1670          fig_name = r'{} PF_hist.png'.format(fiber_name)
1671          # plt.savefig(r'{}\{}'.format(path,fig_name),
                 bbox_inches='tight')
1672          plt.show
1673
1674          plt.figure()
1675          plt.plot(tiprad_hist[1][0:-1],tiprad_hist[0])
1676          plt.title('Tip Radius Distribution, {}'.format(
                 fiber_name),fontsize=24)
1677          plt.xlabel('Tip Radius [nm]',fontsize=20)
1678          plt.ylabel('# of Indentations',fontsize=20)
1679          plt.xticks(fontsize=16)
1680          plt.yticks(fontsize=16)
1681          plt.show
```

1.2 Script for Analysis of Data Exported from .pfc Images or .hsdc Files

This script was utilized for analysis of the raw force curve data exported from either .pfc image files into .txt files or from .hsdc files into .txt files. Each of the exported .txt files consists of tab-separated columnar data for height and force.

```
1
2  import pandas as pd
3  from scipy.optimize import curve_fit
4  import numpy as np
5  import matplotlib.pyplot as plt
6  import math
7  import glob
8
9  def truncate(number, digits) -> float:
10     stepper = 10.0 ** digits
11     return math.trunc(stepper * number) / stepper
12
13 import_files = True
14 plot_hist = True
15 plot_curves = True
16 single_curve = True
17
```

```
18 # Unused modulus
19 # import csv
20 # import statistics as stats
21 # from pathlib import Path
22
23 #%% Force Curve Import
24
25 # if import_files == True:
26
27 #     curve_dir = r'D:\School\Thesis\Frey Data\September\25
       SEP20 - Imaging - H2-1\H2-1 HSDC\Forces Curves'
28 #     fiber_name = r'25SEP20 HM63'
29
30 #     all_files = glob.glob(curve_dir + "/*.txt")
31
32 #     defl = 'Defl_nm_Rt'
33 #     force = 'Defl_pN_Rt'
34 #     height = 'Height_Sensor_nm_Rt'
35
36 #     num_files = 3903
37
38 #     all_curves = []
39 #     defl_plot = []
40 #     force_plot = []
41 #     height_plot = []
42
43 #     for filename in all_files:
44 #         curve_data = pd.read_csv(filename, sep='\s+',
       index_col=None, header=0)
45 #         all_curves.append(curve_data)
46
47 #         defl_plot.append(curve_data[defl])
48 #         force_plot.append(curve_data[force]*10**-3)
49 #         height_plot.append(curve_data[height])
50 #         height_plot = height_plot[::-1]
51
52 #     forces = np.zeros(len(all_curves))
53 #     force_mean = np.zeros(len(curve_data))
54 #     force_std = np.zeros(len(curve_data))
55
56 #     for i in range(len(curve_data)):
57 #         for j in range(len(all_curves)):
58 #             forces[j] = force_plot[j][i]
59 #         force_mean[i] = np.mean(forces)
60 #         force_std[i] = np.std(forces)
61
62 #     plt.figure()
63 #     plt.errorbar(height_plot[0],force_mean,yerr=force_std)
64 #     plt.title('Errorbar Plot of Average Force w/ Height,
       {}'.format(fiber_name),fontsize=16)
65 #     plt.xlabel('Height Sensor [nm]',fontsize=14)
66 #     plt.ylabel('Force [nN]',fontsize=14)
67 #     plt.show()
68
69 # frame_col_cat = pd.concat(li, axis=0, ignore_index=True)
70 # frame_row_cat = pd.concat(li, axis=0, ignore_index=True)
71
72 #%% Test
```

```
73
74 if import_files == True:
75
76     curve_dir = r'D:\School\Thesis\Frey Data\December\03DEC20
          - K13C2U\PFC1\Force Curves\New folder'
77     fiber_name = r'03DEC20 K13C2U'
78     tip_name = r'03DEC20#1'
79
80     all_files = glob.glob(curve_dir + "/*.txt")
81
82     defl = 'Defl_nm_Rt'
83     force = 'nN'
84     height = 'nm'
85
86     num_files = len(all_files)
87
88     all_curves = []
89     defl_plot = []
90     force_plot = []
91     height_plot = []
92
93     for filename in all_files:
94         curve_data = pd.read_csv(filename, sep='\s+',
              index_col=None, header=0)
95         all_curves.append(curve_data)
96
97         # defl_plot.append(curve_data[defl])
98         force_plot.append(curve_data[force])
99         height_plot.append(curve_data[height])
100        height_plot = height_plot[::-1]
101
102    forces = np.zeros(len(all_curves))
103    force_mean = np.zeros(len(curve_data))
104    force_std = np.zeros(len(curve_data))
105
106    for i in range(len(curve_data)):
107        for j in range(len(all_curves)):
108            forces[j] = force_plot[j][i]
109        force_mean[i] = np.mean(forces)
110        force_std[i] = np.std(forces)
111
112    plt.figure()
113    plt.errorbar(height_plot[0],force_mean,yerr=force_std)
114    plt.title('Errorbar Plot of Average Force vs. Height, {}'.
          format(fiber_name),fontsize=24)
115    plt.xlim([160,200])
116    plt.xlabel('Height Sensor [nm]',fontsize=20)
117    plt.xticks(fontsize=16)
118    plt.ylabel('Force [nN]',fontsize=20)
119    plt.yticks(fontsize=16)
120    plt.show()
121 #%% Tip Fitter
122
123 tip_file = r'tip_profiles.csv'
124 tip_data = pd.read_csv(tip_file,sep=',')
125
126 x = tip_data['Depth (nm)']
127 yfit = tip_data[r'{}'.format(tip_name)]
```

```
128
129 # Power Law Fit
130 # pars = np.array([])
131 # for i in range(len(tip_array)):
132 #     pars[i] = curve_fit(f=power_law, xdata=x, ydata=
      tip_array[i,:], p0=[1,1], bounds=(-np.inf, np.inf))[0]
                    # Get the standard deviations of the
      parameters (square roots of the # diagonal of the
      covariance)
133 # pars, cov = curve_fit(f=power_law, xdata=x, ydata=yfit, p0
      =[1,1], bounds=(-np.inf, np.inf))                # Get the
      standard deviations of the parameters (square roots of the
      # diagonal of the covariance)
134 # residuals = yfit - power_law(x, *pars)
135 # ss_res = np.sum(residuals**2)
136 # ss_tot = np.sum((yfit-np.mean(yfit))**2)
137 # r_squared = 1 - (ss_res / ss_tot)
138 # r2_str = 'R^2 = {}'.format(truncate(r_squared,5))
139 # fit_str = 'The tip fit power law is r(z) = {}*z^{}'.format(
      truncate(pars[0],3),truncate(pars[1],3))
140 # print('The fit parameters are a =',pars[0],'and b =',pars
      [1],'.')
141
142 order = 4
143 poly_pars,poly_res = np.polyfit(x,yfit,deg=order,full=True)
      [0:2]
144 ss_tot = np.sum((yfit-np.mean(yfit))**2)
145 r_squared_poly = 1 - (poly_res / ss_tot)
146 r2_str_poly = 'R^2 = {}'.format(truncate(float(r_squared_poly)
      ,5))
147
148 #%% Linearization for Fitting
149
150 def mod_fit(x,a,b):
151     y = a*x**(3/2) + b
152     return y
153
154 def mod_from_pars(par,tip_rad):
155     E_star = par/((4/3)*np.sqrt(tip_rad))
156     return E_star
157
158 max_force = 0.8
159 min_force = 0.4
160 indent_set = 20
161
162 height_store = []
163 force_store = []
164 height_store2 = []
165 force_store2 = []
166 force_fit = []
167 height_fit = []
168 best_fit = []
169 moduli = []
170 E_array = []
171 E_poly_array0 = []
172 E_poly_array1 = []
173 E_poly_array2 = []
174 E_star = []
```

```
175
176 for i in range(num_files):
177     height_store.append(height_plot[i][height_plot[i] > (np.
        max(height_plot[i])-indent_set)])
178     force_store.append(force_plot[i][height_plot[i] > (np.max(
        height_plot[i])-indent_set)])
179
180 for i in range(len(height_store)):
181     height_store2.append(height_store[i][force_store[i] <
        max_force*np.max(force_store[i])])
182     force_store2.append(force_store[i][force_store[i] <
        max_force*np.max(force_store[i])])
183
184 # for i in range(len(height_store)):
185 #     height_fit.append(height_store2[i][force_store2[i] >
    min_force*np.min(force_store2[i])])
186 #     force_fit.append(force_store2[i][force_store2[i] >
    min_force*np.min(force_store2[i])])
187
188 for i in range(len(height_store2)):
189     # height_fit.append(height_store2[i][force_store2[i] >
        force_store2[np.where(np.isclose(force_store2[i],0,atol
        =5e-1))[1][0]]])
190     # force_fit.append(force_store2[i][force_store2[i] >
        force_store2[np.where(np.isclose(force_store2[i],0,atol
        =5e-1))[1][0]]])
191     height_fit.append(height_store2[i][force_store2[i] > 0])
192     force_fit.append(force_store2[i][force_store2[i] > 0])
193
194     pars, cov = curve_fit(f=mod_fit, xdata=height_fit[i],
        ydata=force_fit[i], p0=[1,1], bounds=(-np.inf, np.inf))
195     E_array.append(mod_from_pars(pars[0],np.polyval(poly_pars
        ,200-height_fit[i])))
196     E_star = np.concatenate(E_array)
197
198     pars_poly = np.polyfit(height_fit[i], force_fit[i], deg=2)
199     E_poly_array0.append(pars_poly[0])
200     E_poly_array1.append(pars_poly[1])
201     E_poly_array2.append(pars_poly[2])
202
203     # fit_pars = np.polyfit(height_fit[i],force_fit[i],1)
204     # moduli.append(fit_pars[0])
205     # best_fit.append(fit_pars[0]*height_fit[i] + fit_pars[1])
206
207 print('The mean DMT modulus by the force curve method is',np.
    mean(E_star),'GPa w/ std dev',np.std(E_star),'GPa.')
208 print('The mean DMT modulus by the polynomial fit force curve
    method is',truncate(np.mean(E_poly_array0),3),'GPa w/ rel
    err',truncate(np.std(E_poly_array0)/np.mean(E_poly_array0)
    *100,3),'%.')
209
210 numbins = int(2*(len(E_poly_array0))**(1/3))
211 E_poly_hist0 = np.histogram(E_poly_array0,bins=numbins)
212 E_poly_hist1 = np.histogram(E_poly_array1,bins=numbins)
213 E_poly_hist2 = np.histogram(E_poly_array2,bins=numbins)
214
215 plt.figure()
216 plt.plot(E_poly_hist0[1][0:-1],E_poly_hist0[0],label='n=2')
```

```
217 plt.plot(E_poly_hist1[1][0:-1]*1e-3,E_poly_hist1[0],label='n=1
    ')
218 plt.plot(E_poly_hist2[1][0:-1]*1e-5,E_poly_hist2[0],label='n=0
    ')
219 plt.title('Pseudo-Modulus Parameter Force Curves, {}'.format(
    fiber_name),fontsize=24)
220 plt.xlabel('Polyfit Pseudo-Modulus Paramter',fontsize=20)
221 plt.xticks(fontsize=16)
222 plt.yticks(fontsize=16)
223 plt.legend(fontsize=16)
224 plt.show
225
226 numbins = int(2*(len(E_star))**(1/3))
227 E_star_hist = np.histogram(E_star,bins=numbins)
228
229 mod_plot = pd.read_csv(r'D:\School\Thesis\Frey Data\December
    \03DEC20 - K13C2U\DMT_hist')
230
231 # blank = DMT_mod_hist.to_numpy
232
233 plt.figure()
234 plt.plot(E_star_hist[1][0:-1],E_star_hist[0],label='Force
    Curve')
235 plt.plot(mod_plot.iloc[1][0:-1],mod_plot.iloc[0][0:-1],label='
    Measured')
236 plt.title('Transverse Modulus Force Curves, {}'.format(
    fiber_name),fontsize=24)
237 plt.xlabel('Transverse Modulus [GPa]',fontsize=20)
238 plt.ylabel('# of Indentation',fontsize=20)
239 plt.legend(fontsize=20)
240 plt.xticks(fontsize=16)
241 plt.yticks(fontsize=16)
242 plt.show
243
244 forces = np.zeros(len(all_curves))
245 force_mean = np.zeros(len(curve_data))
246 force_std = np.zeros(len(curve_data))
247
248 for i in range(len(curve_data)):
249     for j in range(len(all_curves)):
250         forces[j] = force_plot[j][i]
251     force_mean[i] = np.mean(forces)
252     force_std[i] = np.std(forces)
253
254 # pars_avg, cov_avg = curve_fit(f=mod_fit, xdata=height_plot
    [0][0:50], ydata=force_mean[0:50], p0=[10], bounds=(-np.inf
    , np.inf))
255 x = height_plot[0][0:int(np.where(force_mean == np.min(
    force_mean))[0])]
256 y = force_mean[0:int(np.where((force_mean == np.min(force_mean
    )))[0])]
257 force_std = np.std(force_mean[0:int(np.where((force_mean == np
    .min(force_mean)))[0])])
258 pars_avg = np.polyfit(x, y, deg=2)
259 print(pars_avg)
260
261 plt.figure()
```

```
# plt.errorbar(height_plot[0][0:35],force_mean[0:35],yerr=
    force_std[0:35])
# plt.plot(height_plot[0][0:35],np.polyval(pars_avg,
    height_plot[0][0:35]))
plt.errorbar(x,y,yerr=force_std)
plt.plot(x,np.polyval(pars_avg,x))
plt.title('Errorbar Plot of Average Force w/ Height, {}'.
    format(fiber_name),fontsize=24)
plt.xlabel('Height Sensor [nm]',fontsize=20)
plt.ylabel('Force [nN]',fontsize=20)
plt.xticks(fontsize=16)
plt.yticks(fontsize=16)
plt.show()

if plot_hist == True:

    numbins = int(2*(len(E_star))**(1/3))
    mod_hist = np.histogram(E_star,bins=numbins)
    # height_hist = np.histogram(height_fit,bins=numbins)
    # force_hist = np.histogram(force_fit,bins=numbins)


    plt.figure()
    plt.plot(mod_hist[1][0:-1],mod_hist[0])
    # plt.plot(height_hist[1][0:-1],height_hist[0])
    # plt.plot(force_hist[1][0:-1],force_hist[0])
    plt.title('Transverse Modulus from Force Curves, {}'.
        format(fiber_name),fontsize=24)
    # plt.xlabel('Transverse Modulus [GPa]',fontsize=14)
    # plt.xlabel('Indentation [nm]',fontsize=14)
    # plt.xlabel('Force [nN]',fontsize=14)
    plt.show

# E_str1 = 'The fit line slopes range from'
# E_str2 = '{} to {}.'.format(truncate(np.max(moduli),4),
    truncate(np.min(moduli),4))
# E_str3 = 'a relative variation of {}%.'.format(truncate
    (((1-(np.min(moduli)/np.max(moduli)))*100),4))
# mods = moduli


if plot_curves == True:

    plt.figure()
    for i in range(len(force_fit)):
        plt.scatter(height_plot[i],force_plot[i])
        # plt.scatter(height_fit[i],force_fit[i])
        # plt.scatter(height_fit[i],best_fit[i])
        plt.title('Force Curve Best Fit Lines, {}'.format(
            fiber_name),fontsize=24)
        # plt.legend(['Force Curve','Best Fit'],fontsize=14)
        # E_str1 = 'The fit line slopes range from'
        # E_str2 = '{} to {}.'.format(truncate(np.max(mods),4)
            ,truncate(np.min(mods),4))
        # E_str3 = 'a relative variation of {}%.'.format(
            truncate(((1-(np.min(mods)/np.max(mods)))*100),4))
        # plt.text(192,0.35,E_str1,fontsize=12)
        # plt.text(192,0.3,E_str2,fontsize=12)
```

```
311             # plt.text(192,0.25,E_str3,fontsize=12)
312             plt.ylabel('Force [nN]',fontsize=20)
313             plt.xlabel('Height Sensor [nm]',fontsize=20)
314             plt.xticks(fontsize=16)
315             plt.yticks(fontsize=16)
316         plt.show()
317
318 #%% Force Curve Reader
319
320 if single_curve == True:
321
322         curve_file = r'September\25SEP20 - Imaging - H2-1\H2-1
                HSDC\Forces Curves'
323
324         curve_data = pd.read_csv(curve_file,sep='\s+')
325
326         defl = 'Defl_nm_Rt'
327         force = 'Defl_pN_Rt'
328         height = 'Height_Sensor_nm_Rt'
329
330         defl_plot = np.array(curve_data[defl])
331         force_plot = np.array(curve_data[force])*10**-6
332         height_plot = np.array(curve_data[height])
333         height_plot = height_plot[::-1]
334
335         min_force = 0.05
336
337         # force_fit = force_plot[force_plot > 0]
338         force_fit = force_plot[force_plot > min_force*np.max(
                force_plot)]
339         # height_fit = height_plot[force_plot > 0]
340         height_fit = height_plot[force_plot > min_force*np.max(
                force_plot)]
341
342         fit_pars = np.polyfit(height_fit,force_fit,1)
343         print(fit_pars)
344
345         best_fit = fit_pars[0]*height_fit + fit_pars[1]
346
347         plt.figure()
348         plt.plot(height_plot,force_plot)
349         plt.title('Force Curve Test, {}'.format(fiber_name),
                fontsize=24)
350         plt.ylabel('Force [uN]',fontsize=20)
351         plt.xlabel('Height Sensor [nm]',fontsize=20)
352         plt.xticks(fontsize=16)
353         plt.yticks(fontsize=16)
354         plt.show
355
356         plt.figure()
357         plt.plot(height_fit,force_fit)
358         plt.plot(height_fit,best_fit)
359         plt.title('Force Curve Test, {}'.format(fiber_name),
                fontsize=24)
360         plt.ylabel('Force [uN]',fontsize=20)
361         plt.xlabel('Height Sensor [nm]',fontsize=20)
362         plt.xticks(fontsize=16)
363         plt.yticks(fontsize=16)
```

```
364         plt.show
```

Bibliography

1. P. Delhaes, "Carbon Filaments, Composites and Heterogenous Media," in *Carbon-based Solids and Materials*. Hoboken: John Wiley & Sons , Inc, 2011, ch. 14, pp. 553–590.

2. Mitsubishi Chemical Carbon Fiber and Composites, Inc, "Selector Guide PAN Fiber," Japan, 2020. [Online]. Available: http://mccfc.com/pan-fiber/ [Accessed: 5/1/20]

3. L. Veigas, "Countering Proliferation: Carbon Fiber Characterization with PeakForce QNM," Master's thesis, Air Force Institute of Technology, 2020.

4. M. A. Montes-Morán and R. J. Young, *Carbon*, vol. 40, no. 6, pp. 857–875, 2002.

5. R. Maurin, P. Davies, N. Baral, and C. Baley, "Transverse properties of carbon fibres by nano-indentation and micro-mechanics," *Applied Composite Materials*, vol. 15, no. 2, pp. 61–73, 2008.

6. International Atomic Energy Agency, "INFCIRC/540 - Model Protocol Additional to the Agreement(s) Between State(s) and the International Atomic Energy Agency for the Application of Safeguards," Austria, 1997.

7. Nanoscience Instruments, "Atomic Force Microscopy," Phoenix, AZ, 2020. [Online]. Available: https://www.nanoscience.com/techniques/atomic-force-microscopy/ [Accessed: 5/1/20]

8. Y. Leng, in *Materials Characterization*. Weinheim, Germany: John Wiley & Sons, Inc, 2013.

9. Y. Hua, "PeakForce-QNM Advanced Applications Training 2014," 2014.

10. F. Marinello, S. Carmignato, A. Voltan, E. Savio, and L. De Chiffre, "Error sources in atomic force microscopy for dimensional measurements: Taxonomy and modeling," *Journal of Manufacturing Science and Engineering, Transactions of the ASME*, vol. 132, no. 3, pp. 0 309 031–0 309 038, 2010.

11. I. N. Sneddon, "The relation between load and penetration in the axisymmetric boussinesq problem for a punch of arbitrary profile," *International Journal of Engineering Science*, vol. 3, no. 1, pp. 47–57, 1965.

12. B. V. Derjaguin, V. M. Muller, and Y. P. Toporov, "Effect of contact deformation on the adhesion of particles," vol. 53, no. 2, pp. 314–326, 1975.

13. V. L. Popov, M. Heß, and E. Willert, *Handbook of contact mechanics: Exact solutions of axisymmetric contact problems*. Berlin: Springer-Verlag, 2019.

14. Bruker Corporation, "RTESPA-525," Billerica, MA, 2020. [Online]. Available: https://www.brukerafmprobes.com/p-3915-rtespa-525.aspx [Accessed: 5/1/20]

15. J. S. Villarrubia, "Tip characterization for dimensional nanometrology," in *Applied Scanning Probe Methods*, B. Bhushan, H. Fuchs, and S. Hosaka, Eds. Berlin: Springer-Verlag, 2019, vol. 0, ch. 5, pp. 147–168.

16. Bruker Corporation, "Bruker NanoScope Software User Guide," Billerica, MA, 2011.

17. J. Slaon, "Carbon fiber suppliers gear up for next-gen growth," Cincinnati, OH, 2020. [Online]. Available: https://www.compositesworld.com/articles/carbon-fiber-suppliers-gear-up-for-next-gen-growth [Accessed: 5/1/20]

18. D. D. Edie and E. G. Stoner, "Effects of Microstructure and Shape on Carbon Fiber Properties," in *Carbon-Carbon Materials and Composites*, J. D. Buckley

and D. D. Edie, Eds. Park Ridge, NJ: Noyes Publications, 1993, ch. 3, pp. 41–69.

19. G. Savage, "Carbon Fibres," in *Carbon-Carbon Composites*. Springer, 1993, ch. 2, pp. 37–83.

20. S.-J. Park, "History and Structure of Carbon Fibers," in *Carbon Fibers*. Singapore: Springer Nature, 2018, ch. 1, pp. 553–590.

21. H. Peterlik, "Carbon Fibers," in *Ceramic Matrix Composites - Materials, Modeling and Technology*, J. L. Narottam P. Bansal, Ed. Hoboken, NJ: John Wiley & Sons, Inc, 2015, ch. 2, pp. 27–39.

22. K. Naito, Y. Tanaka, J. M. Yang, and Y. Kagawa, "Tensile properties of ultrahigh strength PAN-based, ultrahigh modulus pitch-based and high ductility pitch-based carbon fibers," *Carbon*, vol. 46, no. 2, pp. 189–195, 2008.

23. K. Fujita, Y. Sawada, and Y. Nakanishi, "Effect of cross-sectional textures on transverse compressive properties of pitch-based carbon fibers," *Materials Science Research International*, vol. 7, no. 2, pp. 116–121, 2001.

24. Mitsubishi Chemical Carbon Fiber and Composites, Inc, "Selector Guide PAN Fiber," Japan, 2020. [Online]. Available: http://mccfc.com/pan-fiber/ [Accessed: 5/1/20]

25. HEXCEL Corporation, "Selector Guide PAN Fiber," Stamford, CT, 2020. [Online]. Available: https://www.hexcel.com/user_area/content_media/raw/HexTowSelectorGuide.pdf [Accessed: 5/1/20]

26. Toray Industries, Inc, "TORAYCA yarn Product data," Japan, 2019. [Online]. Available: https://www.torayca.com/en/lineup/product/pro_001_01.html [Accessed: 5/1/20]

27. Japan Carbon Fiber Manufacturer's Association, "Carbon Fibers: Product Types by Mechanical," Japan, 2020. [Online]. Available: https://www.carbonfiber.gr.jp/english/material/images/graph01.gif [Accessed: 5/1/20]

28. Bruker Corporation, "Dimension Icon General Support Software," Billerica, MA, 2020. [Online]. Available: https://www.brukersupport.com/ProductDetail/1096?select=Software [Accessed: 5/1/20]

29. N. I. Baurova, "Surface Structure of Fractured Carbon-Fiber Composites Before and After Climatic Aging," *Fibre Chemistry*, vol. 46, no. 4, pp. 241–244, 2014.

30. E. Kollia, A. Vavouliotis, and V. Kostopoulos, "Thermal Ageing of Carbon Fiber-Reinforced Cyanate Ester Composites Under Inert and Oxidative Environment," *Polymers and Polymer Composites*, vol. 16, no. 2, pp. 101–113, 2019.

31. M. Cheng, Y. Zhong, H. Pueh, S. Li, U. Kureemun, D. Cao, and H. Hu, "Environmental durability of carbon/flax fiber hybrid composites," *Composite Structures*, vol. 234, no. September 2019, 2020.

32. S. Duan, F. Liu, T. Pettersson, C. Creighton, and L. E. Asp, "Determination of transverse and shear moduli of single carbon fibres," *Carbon*, vol. 158, pp. 772–782, 2020.

33. W. N. Sharpe, Jr., K. J. Hemker, and R. L. Edwards, "AFRL-IF-RS-TR-2004-76 Mechanical properties of MEMS materials," Air Force Research Laboratory, Rome, NY, Tech. Rep. March, 2004.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704–0188

1. REPORT DATE *(DD–MM–YYYY)*	2. REPORT TYPE	3. DATES COVERED *(From — To)*
25–03–2021	Master's Thesis	April 2020 — Mar 2021

4. TITLE AND SUBTITLE

Error Reduction for the Determination of Transverse Moduli of Single-Strand Carbon Fibers via Atomic Force Microscopy

5a. CONTRACT NUMBER

HDTRA-1033292

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER

6. AUTHOR(S)

Frey, Joshua D., Major, USA

5d. PROJECT NUMBER

5e. TASK NUMBER

5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

Air Force Institute of Technology
Graduate School of Engineering and Management (AFIT/EN)
2950 Hobson Way
WPAFB OH 45433-7765

8. PERFORMING ORGANIZATION REPORT NUMBER

AFIT-ENP-MS-21-M-115

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)

Defense Threat Reduction Agency

10. SPONSOR/MONITOR'S ACRONYM(S)

DTRA RD-ECS

11. SPONSOR/MONITOR'S REPORT NUMBER(S)

Defense Threat Reduction Agency

12. DISTRIBUTION / AVAILABILITY STATEMENT

DISTRIBUTION STATEMENT A:
APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.

13. SUPPLEMENTARY NOTES

14. ABSTRACT

The transverse modulus of single strand carbon fibers is measured using PeakForce Atomic Force Microscopy - Quantitative Nanomechanical Measurement to less than 5% error for 11 types of carbon fiber with longitudinal moduli between 924-231 GPA, including export-controlled fibers. Statistical methods are employed to improve the quality of data to exclude outliers within an measurement and within the sample set. A positive linear correlation between the longitudinal and transverse modulus with an R^2=0.76 is found. Pitch-based fibers exhibit lower measurement error than PAN-based fibers, while PAN fibers exhibited no apparent modulus correlation when the Pitch fibers are excluded. Three alternative methods for calculating the transverse modulus using the raw instrument data were studied. Two methods approximated the indentation force curve, while the third performs a linear fit to the measured force curves.

15. SUBJECT TERMS

Carbon Fiber, Atomic Force Microscopy, Transverse Modulus, Nuclear Forensics, Nanoindentation, Materials Characterization

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			Dr. Abigail Bickley, AFIT/ENP
U	U	U	UU	166	19b. TELEPHONE NUMBER *(include area code)* (937) 255-3636 x4555; abigail.bickley@afit.edu