Spring 2020

What Can Scattered Light Tell You About Your Favorite Magnetic Material?: A Magneto Optical Investigation of the Magnetic Properties of Aligned Janus Fiber Agglomerates.Influence of Dynamic Multiaxial Transverse Loading on Ultrahigh Molecular Weight Polyethylene Single Fiber Failure

Cory John Dolbashian

What Can Scattered Light Tell You About Your Favorite Magnetic Material?: A Magneto Optical Investigation of the Magnetic Properties of Aligned Janus Fiber Agglomerates.

by

Cory John Dolbashian

Bachelor of Science
Slippery Rock University, 2013

Submitted in Partial Fulfillment of the Requirements

for the Degree of Doctor of Philosophy in

Physics

College of Arts and Sciences

University of South Carolina

2020

Accepted by:

Thomas M. Crawford, Major Professor

Yanwen Wu, Committee Member

Richard Creswick, Committee Member

Andrew Greytak, Committee Member

Cheryl L. Addy, Vice Provost and Dean of the Graduate School

ACKNOWLEDGMENTS

A simple "Acknowledgements" section like this can hardly express the gratitude I have towards so many people who have been in my life throughout my graduate career or otherwise.

To begin, I would first like to thank my advisor, Dr. Thomas "Mas" Crawford, for allowing me into the research group and for guidance and motivation for completion of my research. Without his support and facilitation, none of this could have happened.

I would like to thank my advisement committee and other faculty members, Dr. Richard Creswick, Dr. Andrew Greytak, and Dr. Yanwen Wu, and Dr. Matthias Schindler.

Thank you to the staff of the USC Department of Physics and Astronomy for always being kind and helpful regarding scheduling, funding, advisement, and pretty much anything else which needed to be dealt with.

A huge thanks goes to the faculty in the Slippery Rock University Physics Department for cultivating such a close knit department, which was invaluable in motivating me to succeed and even consider graduate school.

Thank you to my Mom Dad and Brother, and my second Mom, Dad, and three sisters by marriage.

In no particular order I would like to thank all the friends I have made along the way: Adam Fisher, Clint Wiseman, Brice Janvrin, Mark Barnes II, Poonam Dhull, Matt Franklin, James Spires, Eddie Gamble III, Nathaniel Cox, Leo Dertah, Jesse Jennings, Kirk Oswell, Pookie, and Gigi.

Finally, thank you to my loving wife Rose, who is simply the best.

ABSTRACT

Here we seek to take a traditional Magneto Optic Kerr Effect (MOKE) experimental design, useful for local magnetization measurements, and apply it to measuring aligned multiferroic Janus nano fiber agglomerates. In order to achieve this we modify the traditional MOKE geometry by measuring our Kerr rotation from collimated scattered light, rather than the conventional specular reflection. Using various techniques to improve signal to noise ratio (SNR), we extend the application of this scattered MOKE geometry to build families of First Order Reversal Curves (FORC). Using an alternative analysis technique, FORC curves are processed and become a FORC diagram, which is shown to look very similar to FORC diagrams created with literature suggested methods. From the FORC diagram we gain insights into how the coercivities are distributed within the aligned agglomerates and how their magnetization evolves as a function of applied field.

TABLE OF CONTENTS

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

As a community of researchers, we have come so far from first principles when it comes to scientific exploration. The sheer vastness of the universe means that there is a neigh inexhaustible well of phenomena to discover, measure, parametrize, characterize, modify, and maximize. There is a certain joy that comes with the discovery that your research is unique and has value. Although topics for research are seemingly splintering and branching, requiring more specific measurement techniques and creative thinking, the truth remains that these phenomena exist to be discovered, and we are the ones who have been given the opportunity to do so. We all stand on the shoulders of giants, and to think otherwise is a doing a disservice to everyone that has come before us. This research is no exception. We combine various broad fields of science: nanoscience, experimental optics, nanomagnetism, data science, and automation to achieve our ultimate goal. Each of these topics in turn can be traced back farther and farther to first principles, but we need not retrace our steps with every advancement. Instead, we acknowledge the tremendous work that has been done before and graciously apply that knowledge to discover the next new physical phenomenon.

This research was born out of a collaboration between my advisor, Dr. Thomas "Mas" Crawford, and Dr. Jennifer Andrew, from the university of Florida. By using a novel nanofabrication technique, the Andrew group was able to create a new composite meta material: biphasic multiferroic Janus nanofibers. While numerous composite multiferroic materials have been created in the material science field, novel materials with cylindrical geometries are avoided due to the difficulty in characterization and application.

Typical routes to characterizing polydisperse nanomaterials involve bulk magneto-electric measurements, although this type of measurement will likely mask any individual directionally dependent effects. An attractive method of measuring these materials without using a bulk measurement seemed to be the optical measurement known as the Magneto Optic Kerr Effect (MOKE), which uses polarized light to magnetically characterize materials on the same order of the optical spot size. Although we were unable to fully isolate and mechanically separate these fibers, we used directed magnetic self-assembly to encourage these fibers to align in parallel agglomerates of few micron diameter. We expect that individual fiber agglomerates would show very different magnetization properties when compared to the bulk due to the distribution of fiber coercivities and strong shape anisotropy. Although MOKE offers the route to microscopic magnetic surface measurements, topographically diverse surfaces serve to drastically reduce the effectiveness of this measurement technique. Rather than abandon the technique, the problem was identified that we were simply searching for a hypothetical "needle" of a signal in a "haystack" of background noise. By cleverly removing the so called "haystack", we are able to more easily find the "needle". From a successful and robust MOKE measurement technique, we expand the scope of our measurement to take First Order Reversal Curves (FORC) measurements. This technique offers information about the specific distribution of a magnetic system made of discrete magnetic components. We show that with the help of various data analysis procedures and careful data acquisition, FORC measurements on aligned nanofiber agglomerates are indeed possible and can offer insight to micromagnetic processes within a system of interacting magnetic entities.

In this document we will begin with relevant background information regarding experimental and theoretical optics, magnetism, multiferroic materials, MOKE, and FORC measurements. Once an adequate amount of background information is covered, we will show how a basic MOKE experiment is carefully and iteratively changed into a robust experiment capable of measuring FORC curves on topologically diverse nanofiber agglom-

erates. Following that, we will reveal the necessary steps required to create a so-called FORC diagram, which is the final step before micromagnetic analysis can begin. We will then attempt to characterize our materials with comparative FORC analysis.

CHAPTER 2

MUTIFERROIC MATERIALS

2.1 A BRIEF INTRODUCTION OF MULTIFERROIC MATERIALS

Many measurements of new functional materials consider the conductivity or resistivity, or, in general, the potential for storing and carrying an electrical charge in a material. When considering magnetic materials, we are often concerned with how well we can store a magnetic field in a material, how much field does it take to reverse this stored field, and how the magnetization of a material evolves with time or under the influence of various applied field gradients. Interpreting the various results of these measurements can help lead to a better understanding of the physical properties of the magnetic material and its dependence on external parameters. Multiferroic materials combine magnetic and dielectric properties by creating a coupling between electric and magnetic properties. A multiferroic material can be electrically polarized with an external magnetic field, and, likewise, magnetized with an external electric field. These materials can be intrinsic multiferroic materials, possessing this coupling between magnetization and polarization innately, or they can be composite multiferroics [1]. In the case of a composite multiferroic material, each contributing material will also have a ferroelastic property in addition to their magnetic or electric ferroicity, and thus the coupling of magnetism to dielectricity is most often mediated by a strain interaction between both materials. If a magnetostrictive material is connected to a piezoelectric material, an external field which is either magnetic, or electric, will induce a strain in the connected material, which will result in a strain-generated field in the dielectric or magnetic material, respectively. Equation 2.1 shows how a magnetoelectric effect (ME) can be

thought of as a product of strain (S) dependent polarization (P) coupled to magnetization (M) dependent strain,

$$ME \propto \frac{dP}{dS} \times \frac{dS}{dM}. \tag{2.1}$$

Composite multiferroic materials can be connected in many ways and are often referred to by the dimensionality of each contributing material. For example, if magnetic particles were suspended in a block of a dielectric material, we would refer to it as a 0-3 geometry, with the '0' referring to the dimensionality of the nanoparticle, and the '3' referring to the bulk material in which the particles are suspended. A few examples can be seen in Figure 2.1, and a full list can be found in reference [2].

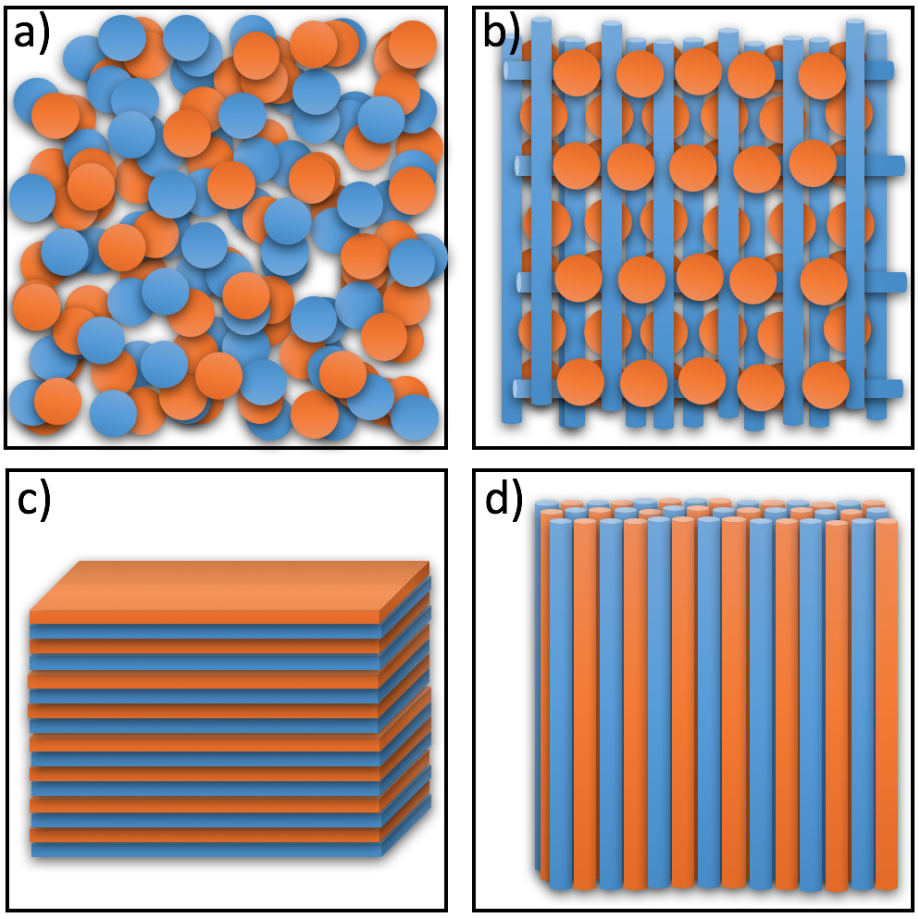

Figure 2.1 A representation of different multiferroic connectivities. a) 0-0 b) 0-1 c) 2-2 and d) 1-1.

While the coupling between magnetization and polarization is a critical parameter for assessing a multiferroic material, the research that follows seeks only to characterize the specific magnetic properties of the multiferroic material described in the next section.

2.2 Sample Fabrication

The ultimate goal of this project is to determine a relatively easy way to magnetically characterize multiferroic Janus nanofibers, eventually leading to the characterization of the ME coupling between composite elements. Before describing the extensive steps to the aforementioned characterization, we begin by describing the multiferroic material and the fabrication technique used. This project is done in collaboration with the Andrew group at the University of Florida. The Andrew group focuses on fabrication of novel materials, and we, in turn, seek to functionalize and characterize novel materials. The material in this case is a collection of randomly oriented multiferroic nanofibers in the 1-1 geometry. The nanofibers used have been created with the nanofabrication method of electrospinning. Electrospinning has been extensively used to create single nanofibers [3] and have been used as a means to facilitate both regular nanoparticle growth [4], as well as Janus nanoparticle growth [5]. The term 'Janus' refers to the distinct phase separation at a specific interface of a composite material. Several examples can be seen in Figure 2.2.

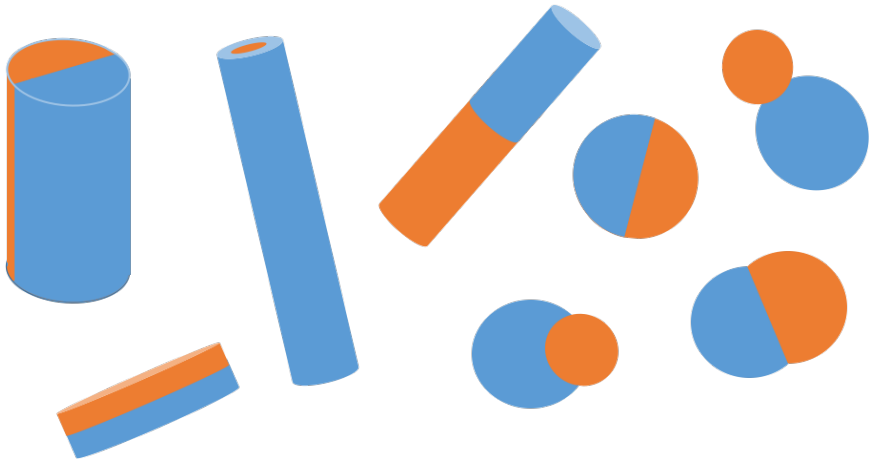

Figure 2.2 Janus nanomaterials come in a variety of geometries. The most common, and which has the most synthesis techniques, is the Janus particle. Janus rods, or fibers, can be attached at their long axis, end-to-end, or as a core-shell material.

The Andrew group has extended this fabrication technique to create Janus-style composite multiferroic nanofibers. Previous attempts at biphasic nanofiber electrospinning has resulted in core-shell [6] and randomly biphasic [7] nanofibers. At the time of publication, little to no work had been done to create a hemicylindrically biphasic Janus-style nanofiber of which both materials are ceramics with different crystal structures. The desire to create this type of material was motivated by a publication which suggested a situationally stronger ME coupling for a 1-1 connected multiferroic material [8] . Although the BTO-CFO Janus was revisited in 2015 by creating a core-shell fiber, the main disadvantage of the core-shell style was the restriction of motion of the core with respect to the radial direction, as it is fully enclosed by the shell [9]. This restriction would often lead to a cracking of the outer shell [10]. The basics of the electrospinning process is shown in Figure 2.3 but is fully described in [9].

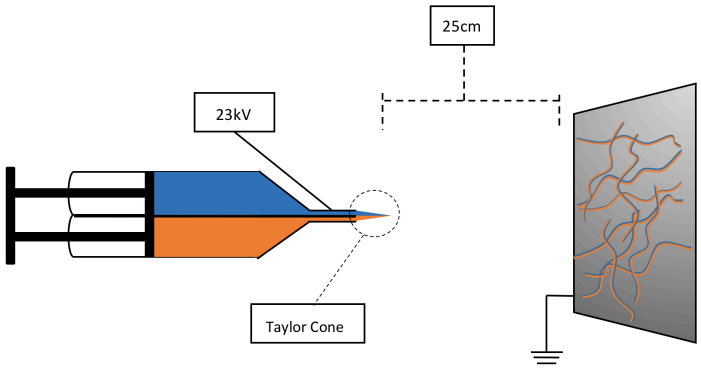

Figure 2.3 The Janus nanofiber electrospinning technique as described in ref. [62]. Precursor gel solutions of cobalt ferrite and barium titanate are loaded into a dual channel syringe. Mechanical depression of the syringe with the addition of a high voltage at the tip forms a biphasic Taylor cone. A grounding plate located 25 cm away from the syringe assists in the extrusion of the nanofiber geometry due to the large voltage difference between syringe tip and grounding plate. The result of the electrospinning process is a mat of randomly aligned and hemicylindrically biphasic Janus nanofibers.

Electrospun Janus fibers are created as a mat of non-preferentially aligned fibers. We seek to functionalize these fibers, and measure their magnetic properties. In order to proceed, the fibers from the spun-mat are crushed and delivered to us in powder form. As mentioned earlier, we are not interested in the bulk properties of these materials, as these types of measurements are indifferent to the unique geometry of the Janus material. Instead, we align the fibers into chains within an external magnetic field. There have been numerous published experiments regarding the chaining of magnetic nanoparticles within an external field. We extend these experiments to our rod-shaped sub-micron diameter nanofibers and have found chaining parameters that differ from the expected nanoparticle chaining parameters [11]. Initially fibers were simply suspended in DI water or Methanol, in order to provide a medium in which the fibers could move freely within during the chaining process. It was later discovered that the alignment of these fibers was partially dependent on the viscosity of the medium in which chaining was occurring. This is likely due to the rod-like geometry of the fibers, and their subsequent rotation to align within the external

field. One of the biggest issues encountered early in alignment experimentation was the aggregation of many fibers into very large agglomerates with diameters between 20-100 μm in diameter (Figure 2.4), compared to the $\sim 1\mu$m diameter of a single nanofiber. In this case, the result was more of a large thin film with a very topographically uneven surface, rather than an agglomerate of a few aligned fibers.

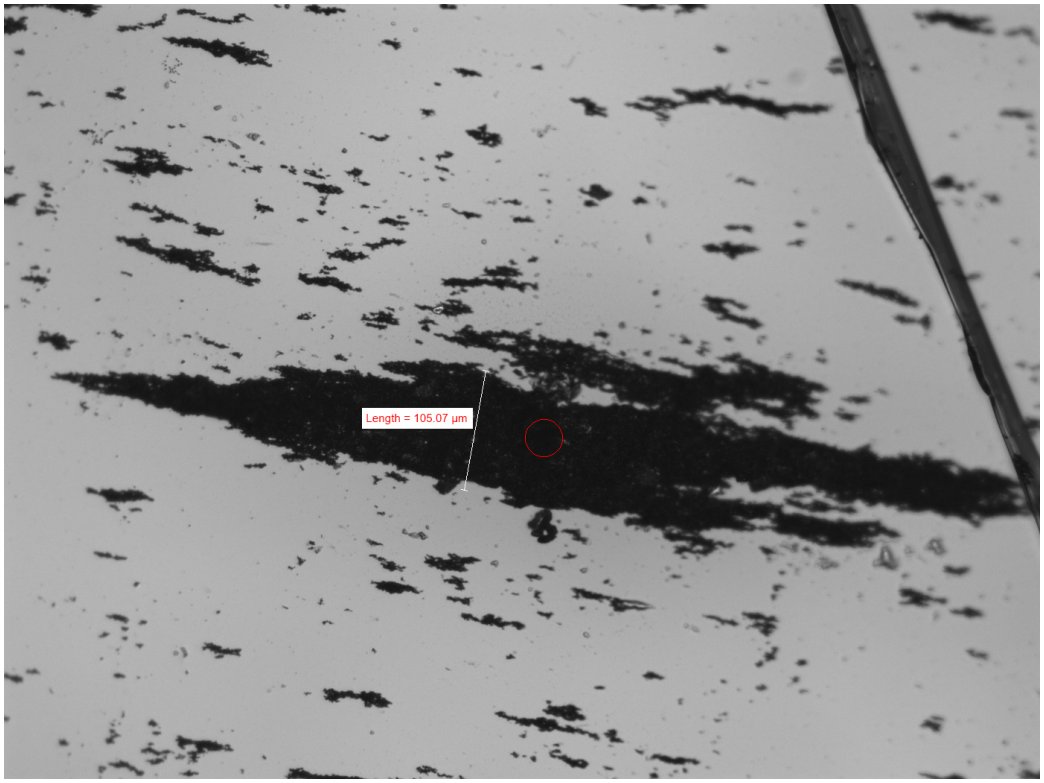

Figure 2.4 Initial alignment tests on pure CFO fibers dissolved in methanol. Fibers would be separated with tip sonication, and mixed into methanol. the methanol-fiber solution was evenly distributed on a glass substrate using a spin coater. The sample was then placed within an external magnetic field until the methanol was fully evaporated. The result is pictured above: aggregates that hardly show that they are actually composed of nanofibers. The circled red spot represents a hole burned into the fiber-island during an early MOKE measurement.

This problem was addressed by suspending the fibers within a viscous, air curable PVA solution and coating the fibers in a citric acid. The viscosity of the PVA would reduce the range in which fibers would aggregate, while the introduction of citric acid would act to

assist in stabilization of fibers within solution; preventing them from aggregating in the absence of an external alignment field. The result of these efforts were agglomerates of fibers which were few μm in diameter and tens of μm in length as shown in Figure 2.5.

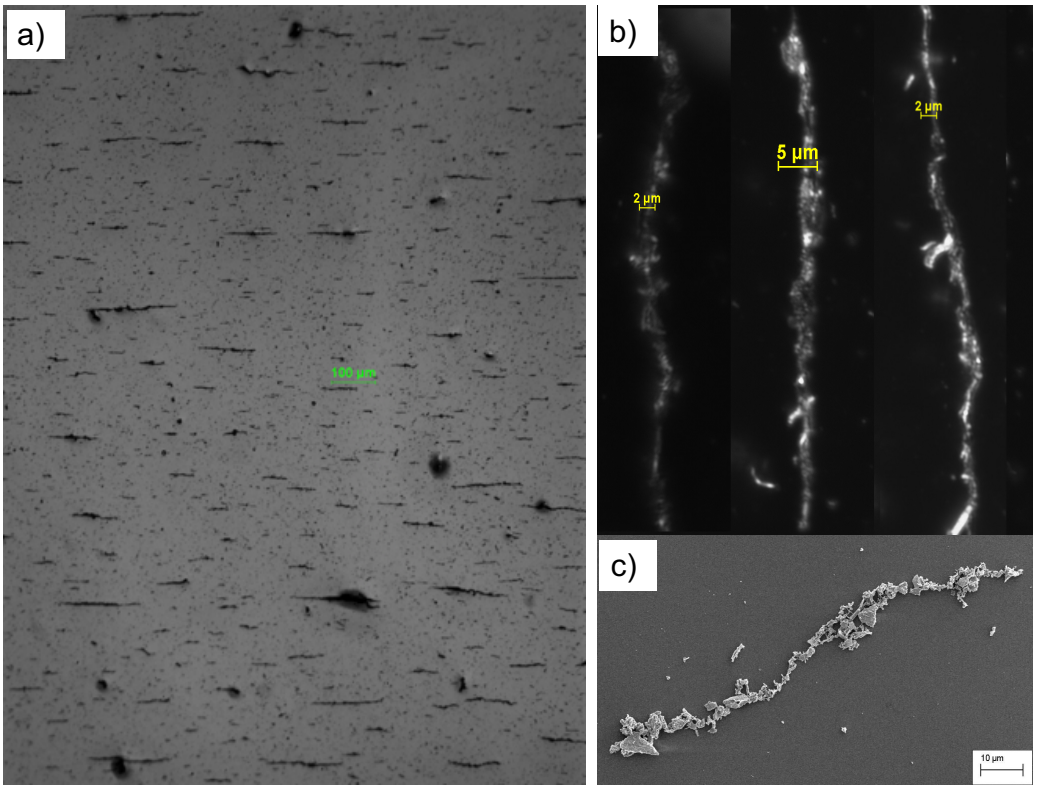

Figure 2.5 a) A portion of the full substrate surface containing aligned and chained fibers which have been suspended in PVA solution. A typical sample covers a 7x7mm^2 area. b) 100x dark-field optical images of fiber aggregates measured with our ScMOKE technique. The suspension-in-polymer and field-chaining process makes it difficult to focus the image at all heights. Note the large variation in individual fiber-alignment even as the aggregate is clearly linear. c) SEM image of field-aligned, non-Janus, CFO fibers, also created via electrospinning, suspended in methanol and dropped onto glass and coated with Au for imaging. The alignment process is of course different between the b) and c) because of the viscosity difference between PVA and methanol [11].

We conducted all measurements on fiber agglomerates resembling the images in Figure 2.5b. When picking fibers for magnetic investigations, we specifically look for fibers with diameters which are less than $\sim 5\mu$m.

CHAPTER 3

EXPERIMENTAL AND THEORETICAL OPTICS

3.1 AN IMPORTANT NOTE ON EXPERIMENTAL OPTICS

Laser optics are truly incredible instruments. The ability to direct light precisely with curved pieces of glass is truly amazing. The supporting theory of ABCD matrixes for predicting beam paths through non-ideal 'thin lenses', and even the effectiveness of the thin lens Equation to predict the path of collimated light through lenses which should clearly not be called 'thin' [12]. In this chapter, we will be talking all about experimental optics: useful experimental techniques, geometric considerations, where to accept 'good enough' as opposed to 'perfect', and how to make the best use of what is available.

3.2 RAY TRACING AND THE THIN LENS EQUATION

Just because we work with non-ideal "thick lenses" in a laboratory setting, does not mean that the basics cannot be applied in some capacity. The use of the thin lens Equation, shown below, and the ensuing ray tracing allows for a great deal of sketching when it comes to optical experimentation.

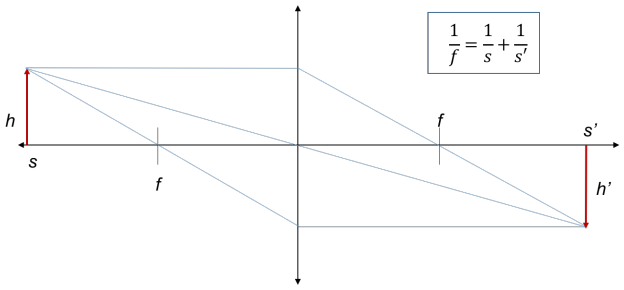

Figure 3.1 A basic ray tracing diagram following the thin lens Equation.

For the above thin lens Equation, f is the focal length, s is the distance from the lens to the object, and s' is the distance from the lens to the image created using the lens. This Equation is invaluable for quickly blocking out the geometry of an experiment since it can often get the experimenter within a few percent of the perfect location of an optical element. In most cases, we use a collimated laser source, which means this Equation is even easier to apply. By treating the laser as an object located at infinity ($s = \infty$), the focal length will simply equal the location of the final focused spot.

3.3 Gaussian Optics

Equation In optical experiments, one of the most important parameters that is considered is the spot size of a laser beam. Because the power distribution of a focused laser is Gaussian, the full width half max (FWHM) or the beam waist (w_0) will represent the effective spot size. For optical experiments involving nanomaterials, the goal is usually to focus the spot as small as physically possible. Using the thin lens Equation as a guide, we are led to believe, "if I just focus my plane wave at the focal length it will be infinitesimally small". Unfortunately, this will lead to frustration as you will see in the following discussion.

Starting from the thin lens Equation, if we assume that the a collimated laser can be treated as an object located at infinity, we are left with the following Equation, regarding the final size of the focussed laser spot size.

$$h' = -\frac{f}{s}h \tag{3.1}$$

The result of our geometric treatment of a beam is that the smallest final height will be achieved by having an object at a large distance s, using a lens with a small focal length f and having a beam a small starting height h, or beam radius in this case. Unfortunately for the modern laser experimenter, this is not true, and will lead to a spot size that is not quite as small as predicted by Equation 3.1. One must be fully aware that the treatment of a laser as an object located at infinity is incorrect when determining image size! A laser is actually a slowly diverging plane wave with a Gaussian distribution of intensity at the wave front. This slight difference makes a very large change in determining the final spot size. In fact, contrary to Equation 3.1, we do not, in fact, want a small initial radius of our spot. We instead desire a larger initial beam size, which will lead to a smaller final spot size. For a Gaussian beam, we see that the dependence on initial beam diameter, D, is inversely proportional to the final beam waist w_0, rather than directly proportional, as seen in Equation 3.1. For a full derivation of the Gaussian beam waist see reference [13]. The waist of a Gaussian beam is given by

$$w_0 = \frac{4\lambda f}{D}, \tag{3.2}$$

where λ is the wavelength of the beam, and f is the ordinary focal length of the 'thin lens'. As I said earlier, approximating a lens as thin is often good enough as an approximation. There are, however, obvious physical limits to some of these parameters. A wavelength is typically selected specifically for a particular experiment, the focal length can often be limited by the geometric considerations of the experiment, and the beam diameter can only expand so large before it can no longer fit through a given lens. In fact, in addition to potentially not fitting through a lens, a large beam can fall victim to an optical defect called spherical aberration. This can be shown using the ABCD matrices for a circular lens.

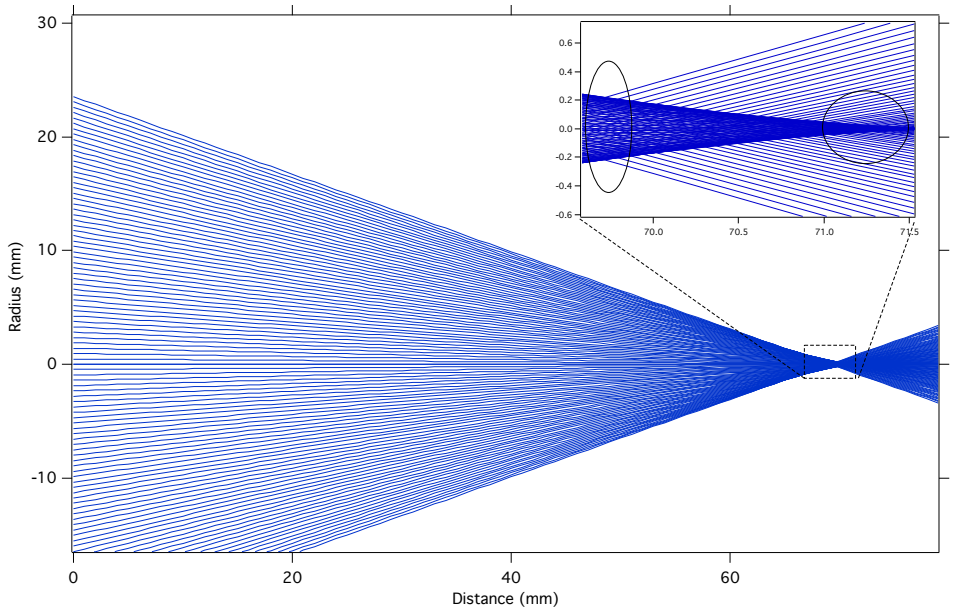

Figure 3.2 The above image is a simulation of parallel rays passing through a single spherical glass surface using the ABCD transfer matrices. The index of refraction is assumed to be 1.5, and the radius of curvature is 12.7 mm. The Y axis indicates vertical position on the lens in mm. The X axis is horizontal position in mm. We assume that the input to the left of the lens is a parallel set of rays. Note that the parallel rays (simulating a beam which completely fills the lens) diverge from their focal location (right circled region in the inset) as they strike initial positions farther from the center of the lens. Rather than achieving a point-like focus, spherical aberration will yield a focal spot much larger (left circled area in the inset).

Ideally, we seek to use Equation 3.2 to achieve the minimum spot size. It would seem as if this is simple, and one need only to use a beam expander to fill the focusing lens while finding a lens with the smallest focal length. Unfortunately, Figure 3.2 shows one of the most common aberrations which prevent easily achieving a tight focus. Fortunately, there are numerous types of lenses beyond the spherical lens which help to mitigate the effect of certain aberrations. Examples are doublet lenses, aspherical lenses, molded glass lenses, or microscope objectives. Specifically, a microscope lens is designed to magnify an image several hundred times with high resolution. Inside the objective there is a carefully designed lens stack to carefully focus light from all parts of the lenses within to a singular point of observation. These are just some of the solutions to the problem of excising

aberrations. However, when addressing these problems, the best solutions can often be financially unachievable, so we often must work with what is available. In our case, we decided to go for the most cost-effective route, which is the use of an Achromatic Doublet. The Achromatic Doublet is a pair of lenses (biconvex and plano-concave) which have been 'cemented' together. In addition to being a lens pair, each of the 2 lenses are have a slightly different index of refraction. Ordinarily, white light passing through a medium will split into its many separate monochromatic wavelengths each moving in a slightly different direction, following Snell's Law (chromatic aberration). Although this still happens when using a doublet lens, the negative effect is reduced by having each of the two lenses at a different index of refraction in addition to having an optical surface within the lens stack acting to slightly modulate the focusing mechanism through a slight dispersion of incident light. White light passing through will be refracted more or less depending on the wavelength, and will focus in a tighter range when compared to a traditional singlet. Fortunately, we are using a nominally monochromatic light source at $\sim$793 nm and the chromatic aberration will not have a large effect on our experiment. The doublet lens also helps to manage the spherical aberration from Figure 3.2. The combination of lenses assists in correcting the off-axis (rays which strike points far from the center of the lens) behavior. By selecting the correct lens for our experiment, we are able to easily reduce the size of our beam without having a too-complex optical focusing scheme. Picking the correct focusing mechanism is not a purely theoretical endeavor, however, as will be shown in the next section.

3.4 Knife Edge Measurements

Rather than relying solely on Equation 3.2, we can use it as a guideline and design a small experiment to determine the beam waist of our Gaussian beam. The particular experiment is referred to as a knife edge measurement [14]. A knife edge measurement is used to show the power distribution of an optical spot size in one or two dimensions. The experimental setup can be seen in Figure 3.3.

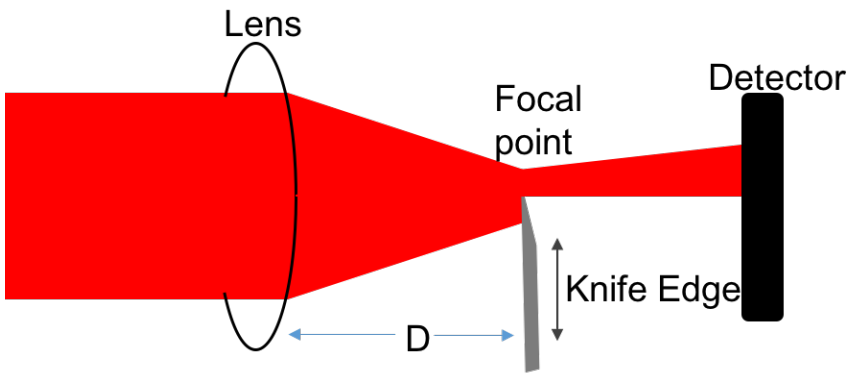

Figure 3.3 A very basic Knife Edge measurement. The ring on the left represents a convex lens. D is the distance from the lens to the location of the 'knife edge' where the measurement is taking place. The black bar on the far right is an optical detector capable of capturing the full diameter of the beam. The knife edge is incrementally moved so that the final measurement is fully unobscured by the knife.

The most important parts of an accurate knife edge measurement are a sharp edged object to fully or partially block a beam with minimal scatter, a laser source, the lens being tested, and an accurate way to translate the knife edge. Starting with a fully blocked beam, record the amount of signal being transmitted past the knife edge. The experiment is now as simple as translating the knife edge incrementally and recording the transmitted power at each increment. The resulting signal should resemble the graph in Figure 3.4 which is actually an error function.

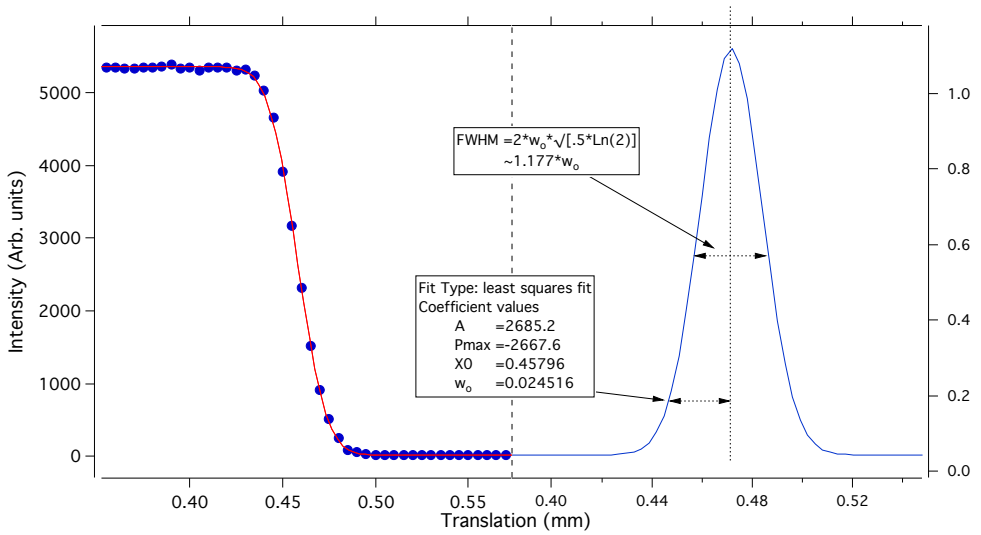

Figure 3.4 Visual representation of the how to extract the beam waist parameter from the knife edge measurement. The dots in the left graph represent the raw data and the line through it shows an error function fit. The specific fit parameters are listed in the lower box within the graph. The graph on the right is a differentiation of the fit function. The derivative of the error function yields a Gaussian, and in our case, the Gaussian profile of the intensity of our beam. On the Gaussian graph there are two widths marked. The lower of the two is the $\frac{1}{e^2}$ value, or the beam waist (w_0). Above that is the typically referenced full width at half max (FWHM) which can be found simply by multiplying the beam waist by $\sim$1.177.

By fitting this error function, we can extract a parameter from the fit which is actually the Gaussian beam waist! The Gaussian beam waist is a multiplicative constant away from the typically referenced FWHM of a beam (Figure 3.4). Repeating the measurement and changing the position of the focussing lens will yield a beam profile for each lens being tested (Figure 3.5). The process is slightly tedious, but with automation and a bit of coding, many of the intermediate steps can be processed with software (see Appendix A.1.1), and the result is an excellent representation of the realistic minimum waist size for different optical elements.

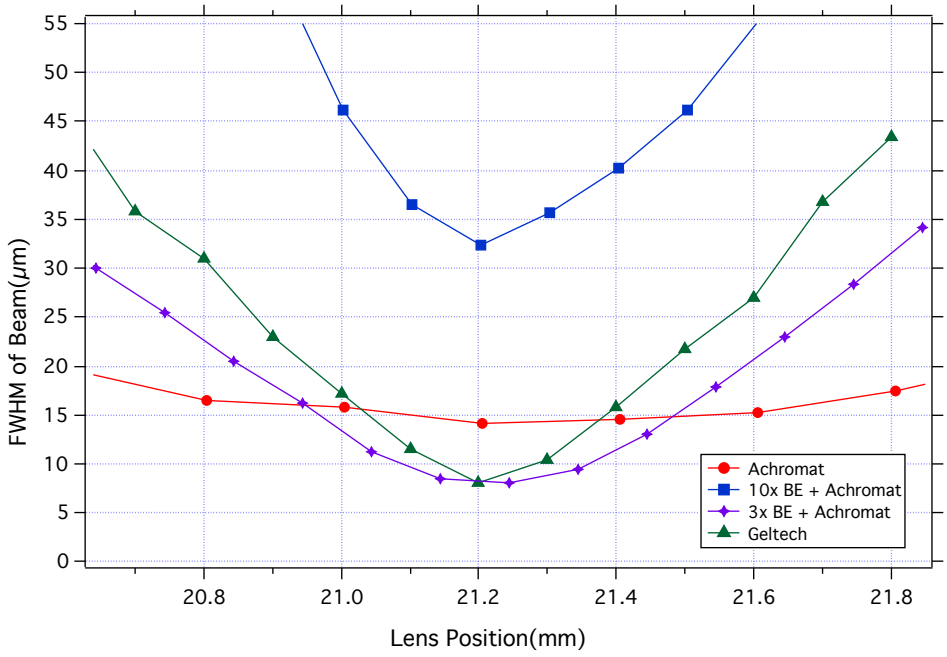

Figure 3.5 The result of a 4 different knife edge measurements. Each of these measurements was conducted with a different set of optics. The circle on the legend is simply mapping a 75mm focal length Achromatic double (AC508-075-B). The square is using the same doublet, but was placed in line with a 10x beam expander. The initial beam diameter was ∼2mm and the 10x expansion almost fully filled the 50.4 diameter lens. The diamond is a 3x beam expander in conjunction with the same doublet. The doublet was selected for all of these since it was the highest quality single lens that was available in the lab. Finally, the triangle is a molded aspheric lens (C280TMD-B), crafted to counteract the spherical aberration. While the geltech and 3x beam expander were able to reach approximately the same minimum beam waist of $\sim 7\mu$m, the depth of focus was extremely small.

The Knife-edge measurement data suggest that the minimum focus achievable with the available optics was approximately 7 μm with the geltech lens. Why did we settle on a single unmodified Achromatic Doublet then? Notice that the depth of focus for the 7 μm spot size to remain less than the unmodified doublet is only about 200 μm. In many experiments, this would be a perfectly acceptable parameter. As you will see in section 7.4.1, there are several experimental considerations that deterred us from the use of the low depth-of-focus lenses. Some of these considerations include a 45° incident angle to our substrate, a target surface that is an indeterminate depth within a transparent PVA solution, and the

low focal length causing an equally short beam divergence. Furthermore, we were able to modify our experimental data collection techniques such that an infinitesimally small beam size was no longer necessary to carry out the experiment. Instead, our beam just needed to be 'small enough', and the $\sim 15\mu$m spot size of the isolated doublet checked that box.

3.5 PLANE WAVE FORMALISM

In the previous section, we saw that treating a collimated source as an object located at a far distance will lead to trouble when it comes to applying optical expressions. We introduced the result of the beam waist of a Gaussian beam without discussing the reasoning why this applies. Here we will introduce the formal definition of a plane wave, the definition of polarization, and outline some of the ways a plane wave interacts with matter at different scales. As is the case in much of Electrodynamics, we begin with Maxwell's Equations,

$$\nabla \cdot \vec{E} = 0, \tag{3.3}$$

$$\nabla \cdot \vec{B} = 0, \tag{3.4}$$

$$\nabla \times \vec{E} = \frac{\partial \vec{B}}{\partial t}, \tag{3.5}$$

$$\nabla \times \vec{B} = \mu_0 \varepsilon_0 \frac{\partial \vec{E}}{\partial t}. \tag{3.6}$$

In empty space, free of charge and currents, the E and B components of Maxwells Equations decouple from each other and the result is a 2nd order differential Equation of $E(t)$ and $B(t)$ that each satisfies the wave equation,

$$\nabla^2 F = \frac{1}{v^2} \frac{\partial^2 F}{\partial t^2}, \tag{3.7}$$

with $v = c$, where c is the speed of light. This implies that an electromagnetic wave will propagate in free space with a speed of c. By solving the wave equation, we find that the solutions that satisfy it are of the form,

$$F(z,t) = F(\alpha z - vt), \tag{3.8}$$

where α has the units of 1/length and v is velocity. We choose to satisfy the wave equation with a sinusoidal function. We call these waves monochromatic as the frequency, ω, determines the wavelength and, consequently, the energy of the wave,

$$F(z,t) = Ae^{i(\omega t - kz)}, \tag{3.9}$$

which is the form of a plane wave travelling in the z direction in space, while oscillating at frequency ω in time. We call the direction of the oscillations, which are orthogonal to the propagation direction, the polarization of the plane wave,

From here, we can use Maxwell's Equations once again to investigate the mutual directionality of the E and B waves respectively. We find that $\hat{E}$ and $\hat{B}$ are perpendicular to the direction of propagation ($\hat{k}$), and likewise are perpendicular to each other. If we have an electric field derived from a plane wave

$$\vec{E} = \hat{E}E_0 e^{i(\vec{k}\cdot\vec{r} - \omega t)}, \tag{3.10}$$

Maxwell's Equations will show us that this electric field will require that the magnetic field take a form

$$\vec{B} = \frac{1}{c}\hat{k} \times \vec{E} \tag{3.11}$$

and both of these will be perpendicular to the direction of propagation $\hat{k}$. For a more explicit derivation of the plane wave formalism, see references [12, 15, 16]

3.6 Scattering and Optical experimentation

As I mentioned in the previous section, collimated light, or a laser, will take the mathematical form of a plane wave propagating in the direction $\hat{k}$ with polarization $\hat{E}$,

$$\vec{E} = \hat{E}E_o e^{-i(\omega t - kr)}. \tag{3.12}$$

We call the direction of propagation r, and the frequency of this monochromatic wave ω. In order to predict the outcome of an optical experiment, we must understand how light, in

the form of a plane wave, interacts with various surfaces. In most instances, surfaces can be broken down by shape and size. For example, we often will consider light interacting with flat planes at various incident angles, spheres of different radii, or cylindrical surfaces of varying radii. With these few shapes and the various sizes therein, one can approximate the majority of optical interactions. We will start with the most simplistic interaction, with the least amount of dependent parameters. We begin with a monochromatic plane wave striking a planar surface at θ degrees from the z axis.

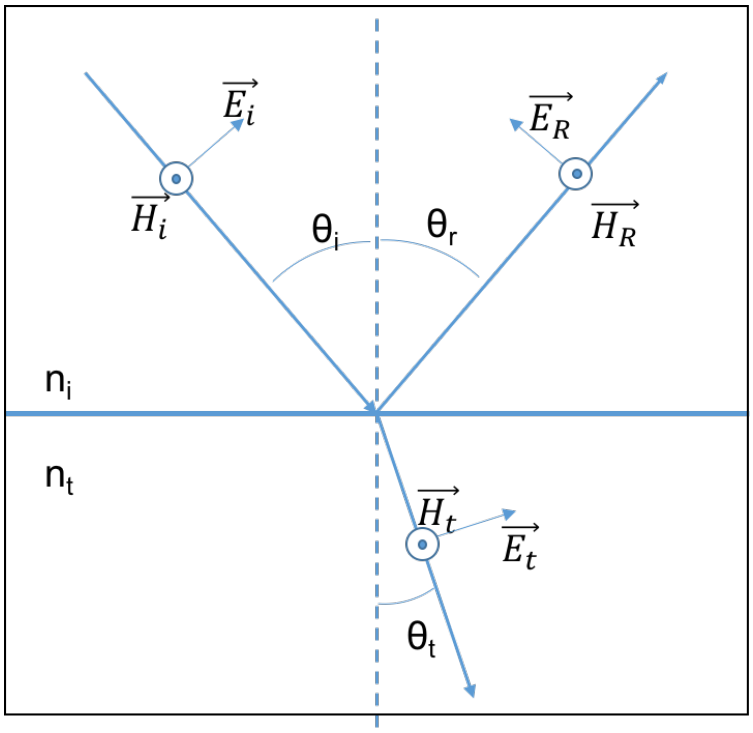

Figure 3.6 A demonstration of the laws of reflection and refraction for a polarized plane wave incident on a surface.

We assume that there will be a portion of the wave which is transmitted through this planar surface and a portion that is reflected back. With this assumption we can generate two more sets of E and B waves, namely, those reflected: E_R, B_R, and those which are transmitted through: E_T, and B_T. The values of these six plane wave can be determined by using the electromagnetic boundary conditions dictating continuity. The specific notation

is symbolically irrelevant for our discussion, and for a full derivation see reference [15]. One of the most interesting and useful conclusions is the so called the "law of reflection", which states that the angle of reflection θ_R will be equal to θ_i. Next we see "Snells Law", or the "law of refraction", wherein the light which is transmitted will change the angular pathway through a medium with a different index of refraction.

$$n_i Sin(\theta_i) = n_t Sin(\theta_t) \tag{3.13}$$

When considering the laws of reflection and refraction, we will often refer to the magnitude of reflections in orthogonal polarization directions as the Fresnel reflection coefficients, which will dictate the percent of light, initially in one polarization state, that will reflect into either the same or orthogonal polarization state. The propagation of light through a medium relies heavily on the index of refraction of the material, which is often treated as a constant. For a homogenous material, this leads to a relatively uncomplicated representation for each of the Fresnel coefficients which depend only on incident angle and the indicies of refraction of each material. We will see later, in Section 5, how the reflection matrix can be changed by simply changing the dielectric properties of the material.

The laws of reflection and refraction can generally be applied as long as your surface is much larger than the wavelength of light which is involved. By considering a circular surface as an infinite amount of flat surfaces, we can easily apply the law of reflection to an object which is much larger than the incident wavelength. This type of ballistic scattering, where light follows the law of reflection, is shown in Figure 3.7. This relatively simple simulation was made in Igor Pro to visualize our experimental optical geometry (see section A.1.2 in appendix).

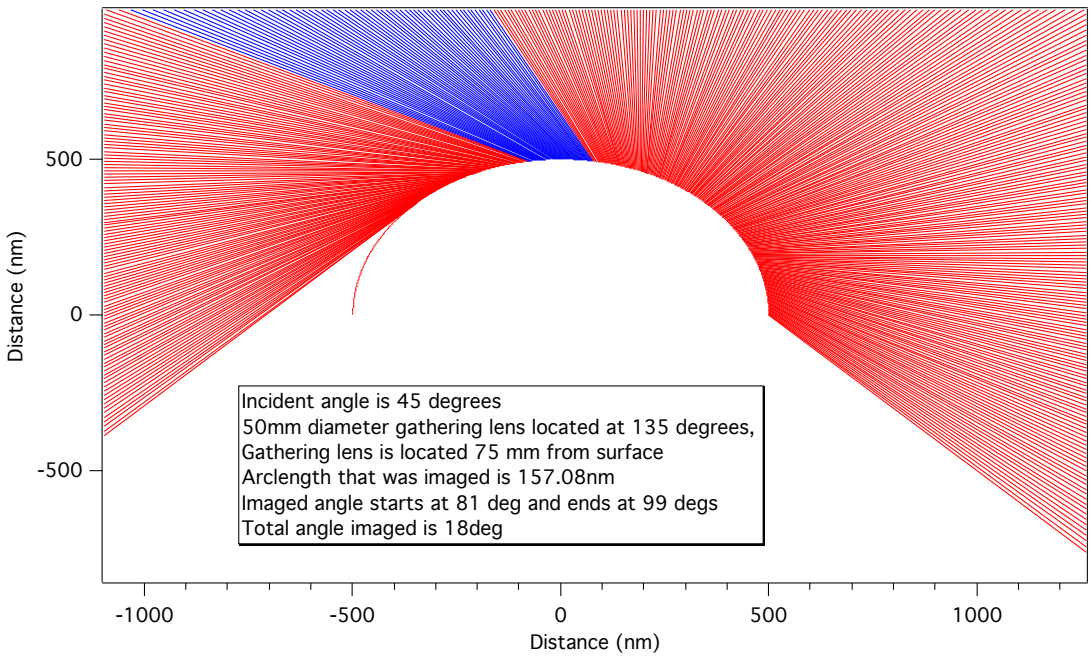

Figure 3.7 This simulation was an exercise to help visualize light reflecting from a circular surface. The light is incident at 45 degrees, originating from the top right of the frame. The diameter of the circle is 1000 nm. The model includes a location of a "gathering lens" where scattered light would be collected, indicated by the blue rays. The annotation within the frame indicates the specific range in which light would scatter, following the law of reflection, into the hypothetical lens.

This is an excellent approach when your object is much larger than the wavelength of light, as is the case for a macroscopic plane. When we begin the shrink the object, we encounter two other distinct formalisms which treat scattered light very differently. At the smallest limit, the wavelength of light $\lambda \gg R$, where R is a length representative of the size of the object. In most cases we consider ellipsoid-like shapes so R can refer to the smallest radial distance. For this case we must use the Rayleigh Formalism. By increasing the size of the object to where $\lambda \sim R$ we enter a new scattering regime: Mie scattering. Both of these require a more mathematical approach when compared to ballistic scattering, but the Mie scattering formalism is what will eventually dictate the experimental geometry of our optical experiment.

3.6.1 Rayleigh Scattering

We will briefly mention Rayleigh scattering first for completeness. A full mathematical derivation is available in refernce [16], and a particle-geometry-specific approach is available in reference [17]. Our research involves 1-5 μm diameter nominally circular wires. Instead of these wires being close to the wavelength of incident light, if we were to, instead, treat these as a very small, long, needles ($R \ll \lambda$), we can get a very smooth distribution of scattered light depending on the incident polarization. We assume that we have a wire which is infinitely long (in order to avoid complications from end effects) lying along the $\hat{x}$ axis. Our incident plane wave is polarized arbitrarily with $\vec{E} = E_{0\|}\hat{y} + E_{0\perp}\hat{x}$, traveling along the $\hat{z}$ direction. We establish a scattering plane in the $y-z$ plane, since we treat the length of the wire as infinite and the angle of incidence is normal with respect to the needle. Following the treatment of scattering from [17], we arrive at an expression for the electric field components of light scattered in directions parallel ($\|$) and perpendicular ($\perp$) to the scattering plane,

$$\vec{E}_{scat} = E_{\|}(Cos(\theta)\hat{y} - Sin(\theta)\hat{z}) + E_{\perp}\hat{x}, \tag{3.14}$$

where $E_{\|}$ and $E_{\perp}$ are electric fields parallel and perendicular to the scattering plane. The values for the two electric fields are dependent on the induced field (E_0), polarizability of the material (α_i), and the induced polarization ($\vec{p}$),

$$E_{\|} = [p_y(\alpha_y)Cos(\theta) - p_z(\alpha_z)Sin(\theta)k^2]\frac{e^{-ikr}}{r} \tag{3.15}$$

$$E_{\perp} = p_x(\alpha_x)\frac{e^{-ikr}}{r}. \tag{3.16}$$

From these we can calculate the intensity of each of these components at various scattering angles $0-\pi$, where θ starts at $+\hat{z}$ as 0 rads. By taking the magnitude of $E_{\|}$ and $E_{\perp}$ the result is similar to the radiating dipole, except this will have a slightly different trigonometric form as the object is not uniformly polarizable and therefore has directional dependance. Thus, the dipole moment term p_i, where the subscript i can be any of the or-

thogonal cartesian directions, will indicate the relative magnitude of polarization projected into the direction indicated by its subscript.

$$I_{||} = \frac{1}{r^2}[p_y^2 Cos^2(\theta) + p_z^2 Sin^2(\theta)k^4 + 2Sin(\theta)Cos(\theta)p_y p_z k^2] \tag{3.17}$$

$$I_{\perp} = \frac{p_x^2}{r^2}. \tag{3.18}$$

The important part of this result, to me, is the relatively elementary trigonometric result of the scattered intensities. Figure 3.9 shows the scattering intensities graphically compared to the same geometry for Mie scattering.

3.6.2 Mie Scattering

We begin the mathematical discussion of Mie scattering, as it applies directly to our experimental geometry. In this case, the wavelength of light is of the same order of magnitude with respect to the size of the object in which we are studying. There are many geometries available for this type of scattering, but only a select few have been fully solved. In the case of this research, one of the solved geometries, is an infinitely long ($L \gg R$) right cylinder. For the case of this discussion, we use references [17, 18, 19], which have solved for the scattering amplitudes of polarized incident light at arbitrary incident angles. The mathematics to reach the solution are non-trivial, so I will simply facilitate the discussion with relevant expressions. Unlike the limiting case of scattering, following the law of reflection, where light reflecting from an object can be mapped back to a specific surface of reflection (Figure 3.7), or Rayleigh scattering which produces a uniform and constant radiative "reflection" at all solid angles in a plane, Mie scattering seemingly combines both effects to a compromising degree. In the Mie regime, we will indeed will have light scattering in 2π directions in the scattering plane following both the short and long wavelength schemes. However, due to the fact that we are unable to easily reduce and simplify the treatment of Maxwell's equations via Taylor expansion or other means, we will end with a very explicit set of intensities that are highly directionally dependent in both incident angle, scattering

direction, and observation angle. Although the plane of scattering will now follow the law of reflection, the treatment of waves at the interface of materials becomes highly nontrivial.

To begin the Mie-scattering-from-infinite-wires problem, we start with a basic description of the geometry which can be seen in Figure 3.8, which has been adapted from the discussion in reference [19]. In the Mie scattering case, we consider two types of incident polarization: TM, and TE wherein the polarization is either parallel to the long axis of the cylinder (case 1) or perpendicular to the cylinder's long axis (case 2).

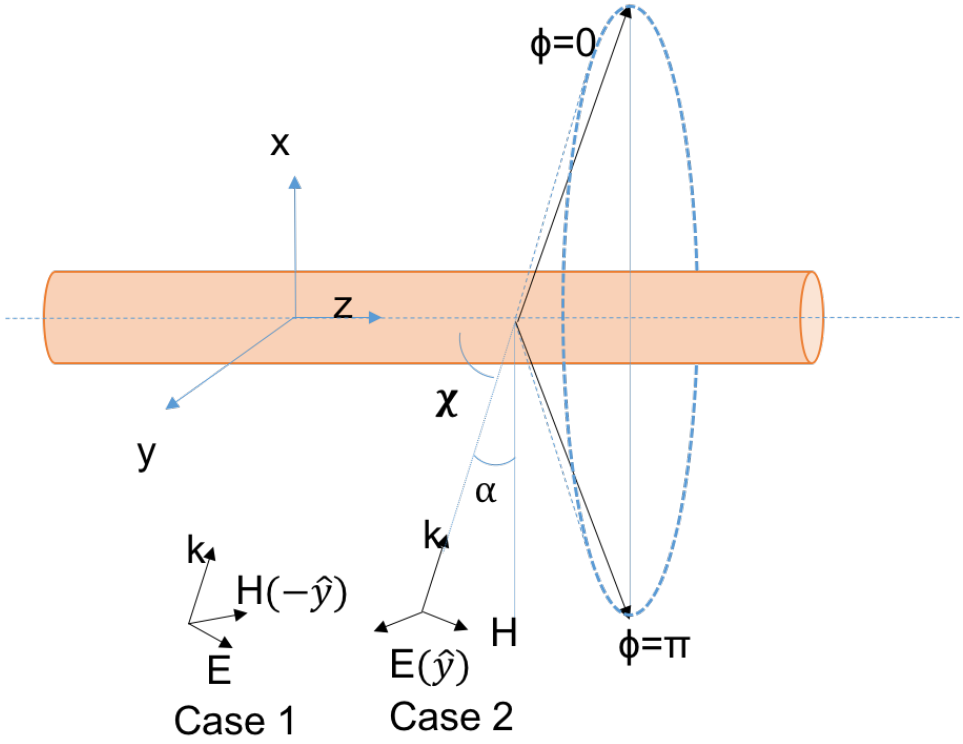

Figure 3.8 Shown here are the geometric specifics as outlined in [19]. The cylinder lies in the $\hat{z}$ direction. The incident wave propagates in the $x-z$ plane, with a variable incident angle α, which rotates about the y axis. E and H are indicative of the orthogonal fields of the plane wave. Case 1 represents incident polarization which is parallel to the long axis of the cylinder (along $\hat{z}$ if α is 0), while case 2 indicates polarization which is along the $\hat{y}$ direction, and is perpendicular to the long axis at all α. The cone of scattering is indicated in the image with forward scattering occurring at $\phi = 0$ and backscattering occurring at $\phi = \pi$. Normal incidence occurs at $\alpha = 0$ which reduces the cone of scattering to a flat plane.

We begin with Maxwell's Equations and create, via separation of variables, a vector and scalar wave equation,

$$\nabla^2\vec{A}+k^2m^2\vec{A}=0 \tag{3.19}$$

$$\nabla^2\Psi+k^2m^2\Psi=0, \tag{3.20}$$

where $\vec{A}$ and ψ are vector and scalar functions respectively, k is the wave number and m is the complex refractive index.

From Maxwell's Equations, the mathematical steps are highly non trivial, and for a full explanation, it is suggested to read a combination of references [16, 17, 18, 19]. Following the proper mathematical treatment for a plane wave polarized perpendicular to the long axis of our infinite cylinder, we have scattering intensities with the form

$$I_{22}=\frac{2i_{22}I_0}{\pi k(rCos\alpha+zSin\alpha)} \tag{3.21}$$

$$I_{21}=\frac{2i_{21}I_0}{\pi k(rCos\alpha+zSin\alpha)}, \tag{3.22}$$

with i_{22} and i_{21}, being the intensity coefficients,

$$i_{22}=|b_{01}+2*\Sigma_{n=1}^{\infty}b_{n1}Cos(n\phi)|^2 \tag{3.23}$$

$$i_{21}=|2*\Sigma_{n=1}^{\infty}a_{n1}Sin(n\phi)|^2. \tag{3.24}$$

Again, we end up with a non-illuminating solution, since the intensity coefficients b_{n1} and a_{n1} are infinite sums of various Bessel functions and their derivatives. Rather than focusing solely on the resulting mathematical expression, it is helpful to view this data graphically (Figure 3.9). For a given incident angle α we can produce $I(\phi)$ and look at how the resulting distribution of intensities compare to the small wire (Rayleigh) formalism. In this graph we have not completed the infinite sum as required by the intensity coefficients above. Instead we have taken the first 100 terms for computational reasonability.

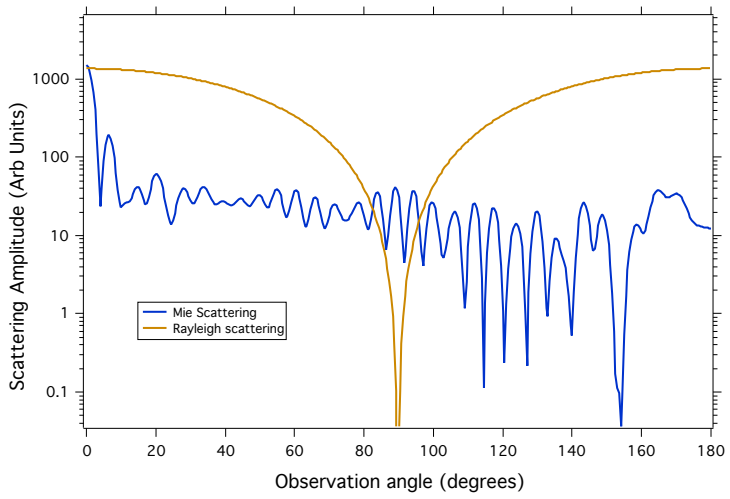

Figure 3.9 The Mie scattering amplitudes for the MOKE orientation considered in the experimental section. The data considers Mie scattering from a 5μm cylinder at normal incidence with a polarization aligned perpendicular to the long axis. The magnitude of scattered light changes sharply as observation angle moves through 180 degrees. Considering our experimental geometry, we are viewing the 120-140 degree range of angles

Looking at Figure 3.9 we can see that a significant amplitude of light polarized parallel to the incident light can be found any solid angle of reflection. These data are not to be taken as absolute numerical values. Following the geometry of Figure 3.8, we can see that each of these are a single input of results of i_{22} or i_{21} from Equation 3.22, and if we want the real intensity, there is a denominator that cares about where along the z axis (the length of the wire) the observation is occurring. In a realistic experimental setting we are using a lens to capture optical information which has a non-negligible area, and therefore a distribution of intensities will be collimated into a single optical signal. Additionally, from Figure 2.5b, we see that we do not have normal incidence, and instead the sub-micron structure of the aligned fiber agglomerate will deviate from a right cylinder at normal incidence. The Mie scattering intensities show us how dissimilar this regime of scattering is from the Rayleigh and ballistic scattering. We will be using the knowledge of these scattering intensities to modify a traditional MOKE geometry in the experimental chapters.

CHAPTER 4

MAGNETISM AND MAGNETIC MATERIALS

4.1 INTRODUCTION

Magnetism is too large of a topic to be covered in a document such as this. Instead of attempting to address magnetism from first principles, this section should serve to remind the reader of relevant magnetic terminology as well as introduce the reader to experiment-specific formalisms. For a full introduction to magnetism, see [20, 16, 21].

4.2 VARIETIES OF MAGNETISM

There are generally five types of magnetism which are commonly referenced: ferrimagnetism, ferromagnetism, paramagnetism, anti-ferromagnetism, and diamagnetism. The most distinguishing feature about these types of magnetism is their response to a reversing magnetic field. When all spins within a magnetic material point in a parallel direction, we refer to this as magnetic saturation. At this point, increasing an external field will no longer produce a change in the orientation of the internal magnetism of the material. When an external magnetic field is slowly reversed such that we want to fully saturate the magnet in the opposite direction, one of two things will happen. The first is that the decreasing field-induced magnetization will be identical to the increasing field-induced magnetization, as shown in the top image of Figure 4.1. This is the case for diamagnetic, paramagnetic, and antiferromagnetic materials. In this case, there is no dependence on the magnetic history of the material, meaning the magnetization is a function of applied field with a singular value of magnetization for each field value. Additionally, the magnetic permeability, μ, is just

the slope of the magnetization vs applied magnetic field. For the previously mentioned materials with a linear response to an externally applied field, this value is nominally constant. In the second case, instead of reversing magnetization by following the saturation path in reverse, the ferro/ferrimagnetic materials store a portion of the energy used to saturate and will follow a different magnetization path. As a result, a ferro/ferrimagnetic material has magnetic permeability which is a nonlinear function of applied field, rather than a constant value [20]. This phenomenon of irreversibility is referred to as magnetic hysteresis, and the specific shape of the hysteresis is of interest for researchers who study magnetic materials. In order to describe magnetic hysteresis data, scientists will often use the terms coercivity and magnetic remanence (bottom Figure 4.1).

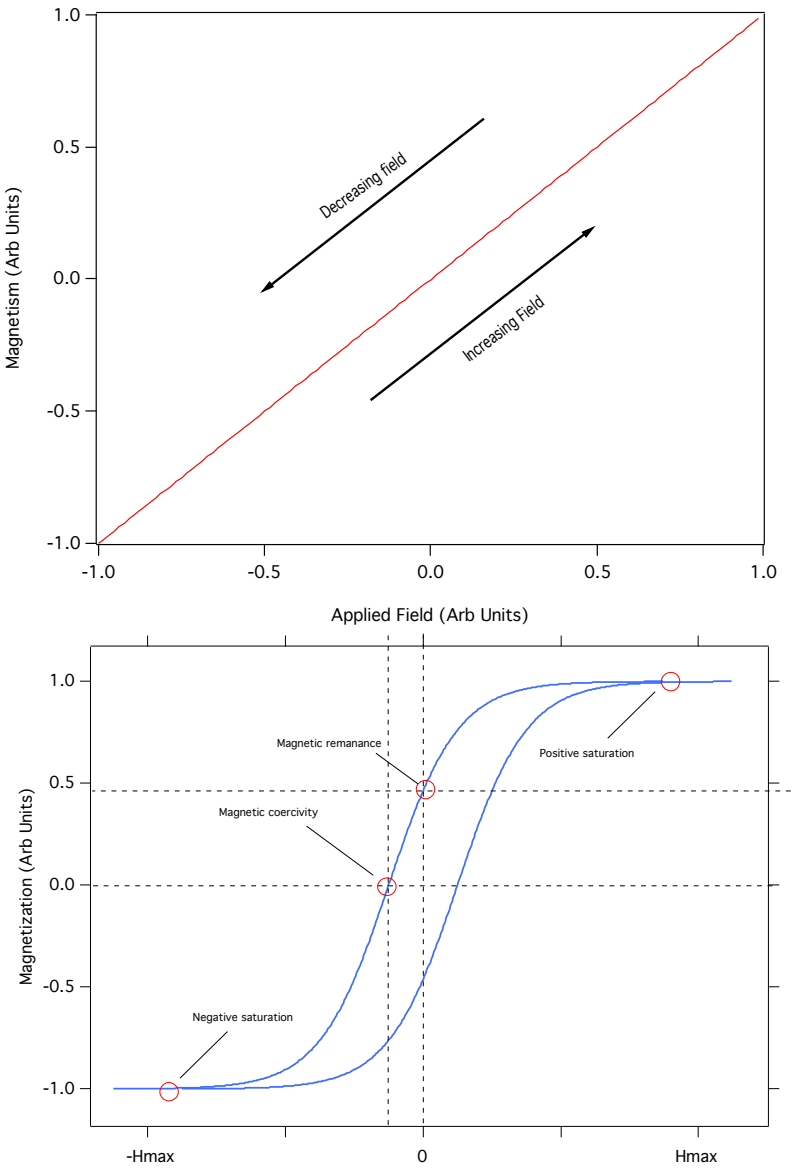

Figure 4.1 The top image shows a magnetization curve for a paramagnetic material. There is no hysteresis and the response to an external magnetic field is linear. The bottom image shows a representation of a ferromagnetic magnetization curve. Relevant field values are marked on the image.

There are numerous factors contributing to the actual values of magnetic coercivity and the remanant field. In the next section, we will specifically investigate magnetic anisotropies and how they affect the magnetization process.

4.3 Anisotropies and Domain Wall Motion

In general, the magnetization process is not directionally invariant. In other words, the magnetization path is highly dependent on the spatial direction in which the material is magnetized. All of the various things that contribute to this directional dependence are referred to as magnetic anisotropies. These will directly affect the shape of the hysteresis curve, and the corresponding rate at which magnetization occurs for a given applied field.

One of the most basic of the anisotropies is the crystal anisotropy, which directly corresponds to the structure of the crystal unit cell. A unit cell is defined by orthogonal directions $\langle 100 \rangle$, $\langle 010 \rangle$,and $\langle 001 \rangle$, each of which defines an axis within a crystal cell. As a basic example, we will consider a cubic crystal. A cubic crystal magnetized along different crystallographic directions will exhibit varying rates of magnetization along each of its orthogonal crystallographic directions, and likewise, in directions that are linear combinations of the orthogonal directions. We refer to directions which can be magnetized with very little field as 'easy' directions, and the crystallographic directions with a larger field required to saturate as 'hard' directions. The easy and hard crystallographic directions can be discovered by taking orthogonal slices of a magnetic material along carious crystallographic directions and observing the resulting magnetization curves. Crystal anisotropy affects the magnetization process by changing the way magnetic domains are formed. Magnetic domains are isolated "islands" of magnetic spins with equivalent magnitudes of magnetization per domain, but different directions of the net magnetization. Domains are not smoothly changing from one domain to another, but are instead separated by thin "domain walls", over which the magnetization rotates between the magnetizations of adjacent domains over a very short distance, as to almost be a discrete change. If we have a magnetic material which has a zero net magnetization, this does not mean that there is an absence of magnetic domains. Instead, each domain will spontaneously point in one of the easy axis directions so that the vectorial nature of the domains yields a net zero magnetization for the whole volume. Upon experiencing an external field parallel to one of these domains, the domains will re-

orient themselves in order to minimize the system energy. We can predict how the domains will be classified and consequently evolve by considering the equation for magnetic dipole energy,

$$E_{dipole} = -\vec{m} \cdot \vec{H} \tag{4.1}$$

$$E_{dipole} = -mHcos\theta, \tag{4.2}$$

where E_{dipole} is the energy from a single dipole, $\vec{m}$ is the magnetic dipole moment, $\vec{H}$ is the applied field, and θ is the angle between $\vec{m}$ and $\vec{H}$. In this case, the domain with minimal energy will be one which is parallel to the applied field. This domain will grow until it covers the entire magnetic surface/volume. At the point when this domain has overtaken all others, we have reached magnetic saturation. In the case when a field is applied in an easy crystallographic direction, a material can be fully saturated simply by domain wall motion, as described. However, when a field is applied in a non-easy direction, domain wall motion will still act to minimize energy until there are an equivalent number of minimum energy domain orientations. Since we were not acting along an easy crystallographic direction, there may be multiple domain orientations that are equivalent in minimum energy. Domain wall motion will act to eliminate all domains except the minimum two, and, at that point domain wall motion ends. Further application of the field causes the entire domain to rotate the direction of its net magnetization, and this requires much more energy, in the form of an external field, and is referred to as domain rotation. This is only one of the many anisotropies, but is the easiest to understand.

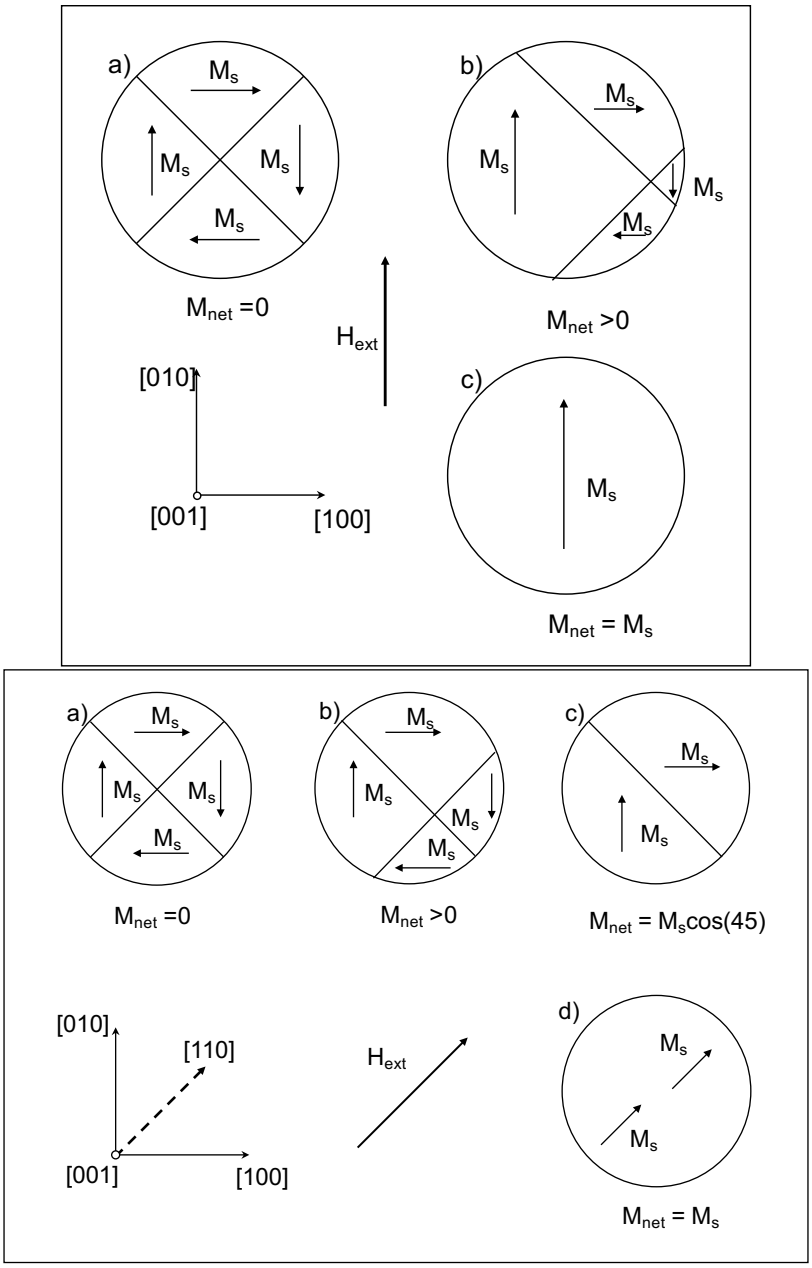

Figure 4.2 Demonstration of domain wall motion and domain rotation. In the top image the domains experience an external field indicated by H_{ext}. Each individual domain, is magnetized with M_s. Domains will grow to minimize energy, and therefore, will cause the domain parallel to H_{ext} to grow until it is the only domain. At that point M_{net} is equal to the magnetization of each individual domain. In the bottom image, the external field is applied in a direction that is not represented by any of the existing domains in a material. In this example it is represented as an applied field at 45° with respect to the [100] axis. When minimizing energy, we will end in a situation, shown in instance c) where both states share an equivalent energy state. With more applied field, both of these domains will rotate their net magnetization to align to the external field. This Figure has been adapted from reference [21].

The more relevant anisotropy to our experimentation is called shape anisotropy. The name of this particular anisotropy is indicative as to what the particular dependence is. In addition to considering the crystal structure of a unit cell, we also must consider the macro/micro/nano-scopic shape of our magnet. Shape anisotropy originates from the difference in the magnitude of the demagnetizing field within a material along different orthogonal directions. The example of a prolate spheroid is solved and is the easiest example. For a given geometry there exists demagnetization factors along each of the 3 orthogonal directions where the sum of these three factors is unity. From literature [21] we see that for a prolate spheroid with equivalent semi major axes, the demagnetization factors along each direction are.

$$N_c = \frac{1}{m^2-1}\left(\frac{m}{(m^2-1)}ln(2m)-1\right) \tag{4.3}$$

$$N_a = N_b = \frac{1-N_c}{2}, \tag{4.4}$$

where N_c and $N_{a,b}$ are demagnetization constants along the long and short axes of the ellipsoid respectively, and m is the ratio $\frac{c}{a}$. The demagnetization field in one direction in an infinite volume can be expressed as

$$\vec{H}_d = N_d \cdot \vec{M}, \tag{4.5}$$

where N_d is a generic demagnetization factor, similar to the specific values above. Continuing with our example regarding the prolate spheroid, we look at the demagnetizing energy for an arbitrary magnetization direction of the prolate spheroid, with M being aligned with an angle θ with respect to the long axis. The resulting energy will be

$$E_{demag} = \frac{1}{2}M^2(N_c) + \frac{1}{2}M^2(N_a - N_c)sin\theta^2. \tag{4.6}$$

By minimizing the above energy with respect to θ, assuming azimuthal symmetry, we find that the easy axis lies parallel to the long axis and the hard axes will be in the azimuthal plane of $\theta = \frac{\pi}{2}$. This treatment illuminates the varying factors on which the magnetization process depends. In section 7.4.1 we will use these facts as a basis of discussion.

CHAPTER 5

MAGNETO OPTIC KERR EFFECT

5.1 INTRODUCTION AND HISTORY

In order to characterize these aligned fiber agglomerates, we have chosen the Magneto Optic Kerr Effect (MOKE) as the magnetic characterization tool which will be used throughout all experimentation. MOKE is an optical phenomenon in which a plane polarized beam reflecting from a magnetic surface will experience a rotation of polarization direction and/or a change in ellipticity. The nomenclature comes from the founder of this particular effect, John Kerr, who reflected a polarized laser from the surface of a polished bar magnet [22]. The effect that he discovered is now called the "generalized Kerr effect", since the Kerr effect can further be broken into three distinct orthogonal geometries: Polar (PMOKE), Longitudinal (LMOKE), and Transverse (TMOKE) (Figure 5.1). Each of these three orthogonal geometries is defined by the direction of magnetization within the material with respect to the optical incident plane.

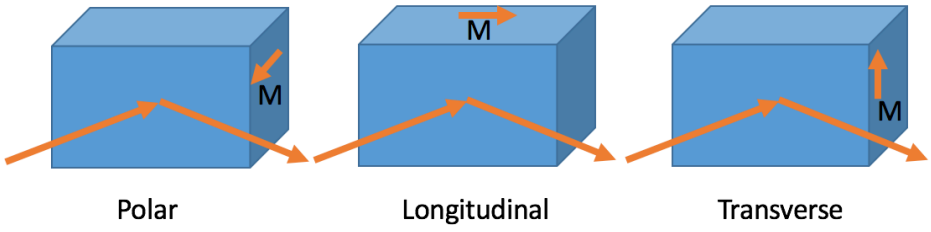

Figure 5.1 Figures A-C show the three orthogonal MOKE geometries. The difference between each is the direction of magnetization within the magnetic material. This can be an intrinsic direction of a permanent magnet, or the induced magnetization of the material as a result of an external field.

Each of these three distinct MOKE geometries can be further broken down in to two separate categories determined by the polarization direction of the incident plane polarized beam. These polarization directions are defined by the convenient orthogonal basis of a tabletop experiment: $\hat{s}$ and $\hat{p}$, which are polarized perpendicularly to the plane of incidence and parallel to the plane of incidence respectively. In our experimental setting, $\hat{s}$ is normal to the surface of the optical table, and and $\hat{p}$ lies parallel to the surface of the table, while also being perpendicular to the direction of propagation of the beam. For each MOKE geometry there are two different orthogonal "rotations", yielding a total of six different effects, resulting from the combinations of the three MOKE geometries and two orthogonal polarization states of the beam. The magnitude of these effects is not easily predictable for each of these different combinations of MOKE geometries and polarization, as they depend on the crystalline structure and magnetic properties of the material being measured. Knowledge of these parameters has yielded theoretical rotations matching experimental results, however [23]. It is simpler to predict the directions of the various Kerr effects and polarization states, by doing a Lorentz Force approximation,

$$\vec{F} = q(\vec{v} \times \vec{B}). \tag{5.1}$$

In the above equation we can replace $\vec{B}$ with the induced magnetization, $\vec{M}_{ind}$ as a result of the applied magnetic field $\vec{H_{app}}$. Next, we replace $q\vec{v}$ with the vector form of the polarization, which can be represented as the electric field of the polarized beam $\vec{E}$. Now, with these substitutions, our resulting equation is more transparent,

$$\hat{F} \sim \vec{E} \times \vec{M}_{ind}. \tag{5.2}$$

Note that we have replaced the vector value of the force with the "direction" of the force, and the equivalence with an approximation. This was done to comply with my earlier statement that this does not give a magnitude of the the rotation, just the direction in which the polarization will rotate. Using this equation, we can can show, pictorially, the six possible interactions below.

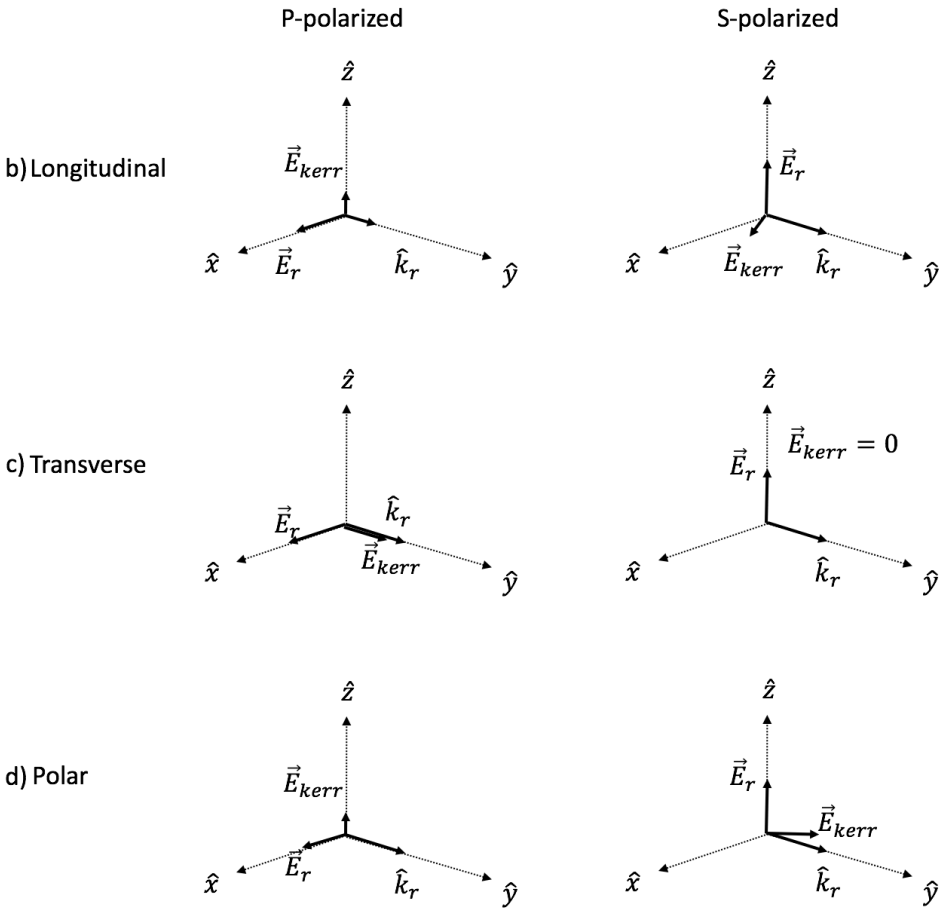

Figure 5.2 Each of two columns represents one of the orthogonal incident polarizations, while each of the rows shows one of the orthogonal MOKE geometries. In all six of these images, $\vec{E}_{Kerr}$ represents the component of polarization created through the Kerr interaction, approximated with the Lorentz force. The incident beam propagates along the -$\hat{x}$ direction with the $\hat{s}$ and $\hat{p}$ polarization directions being in the $\hat{z}$ and $\hat{y}$ directions respectively. $\vec{E}_r$ and $\vec{k}_r$ are the Kerr-independant polarization and propagation vectors of the reflected beam respectively. The new polarization state will be a linear combination of $\vec{E}_r$ and $\vec{E}_{Kerr}$. The direction of each of the $\vec{E}_{Kerr}$ vectors is computed with the above Lorentz treatment of the Kerr rotation, by taking the cross product of the incident polarization direction, and the direction of the applied magnetic field, according to the different LMOKE, TMOKE, and PMOKE geometries.

While the Lorentz force formalism is an easy way to quickly approximate the directional effect of different Kerr interactions, the origin of MOKE is a consequence of the dielectric properties within a given magnetic material. For an ordinary, linear, non-magnetic

material, an electric field vector will propagate through a material with directionally dependent diagonalized dielectric matrix ε,

$$\varepsilon = \begin{bmatrix} \varepsilon_{xx} & 0 & 0 \\ 0 & \varepsilon_{yy} & 0 \\ 0 & 0 & \varepsilon_{zz} \end{bmatrix}, \tag{5.3}$$

with ε_{aa} corresponding to orthogonal crystallographic directions within any given material. When considering a plane wave with polarization direction $\hat{E}$, incident at the interface of a material with dielectric matrix ε, we will have a resulting matrix of Fresnel coefficients which represents the relative intensities of the reflected magnitudes of orthogonal polarization states.

$$R_f = \begin{bmatrix} r_{pp} & 0 \\ 0 & r_{ss} \end{bmatrix}, \tag{5.4}$$

where r_{xy} represents the magnitude of the reflected polarization of incident direction x with reflected direction y. Note that there are no coefficients that have a mixture of incoming and outgoing polarization directions. Now we look at the dielectric matrix of a magnetic material.

$$\varepsilon = \begin{bmatrix} \varepsilon_{xx} & Q_z & Q_y \\ Q_z & \varepsilon_{yy} & Q_x \\ Q_y & Q_x & \varepsilon_{zz} \end{bmatrix}. \tag{5.5}$$

Instead of a diagonalized matrix with orthogonal permittivities corresponding to different crystallographic directions within a material, we have the existence of off-diagonal elements called Voigt magneto-optic coefficients [24]. These Q factors are highly dependent on the magnetization of the material being measured, and their existence will lead to a Fresnel coefficient matrix of values that has two addiditonal elements compared with the non-magnetic matrix,

$$R_f = \begin{bmatrix} r_{pp} & r_{ps} \\ r_{sp} & r_{ss} \end{bmatrix}. \tag{5.6}$$

There now exist elements for the mixed polarization states where a p-polarized incident beam will reflect with a component in the $\hat{s}$ direction and conversely, an s-polarized beam will generate a $\hat{p}$ component. This argument is the basis of Kerr rotations, and the same argument can be applied to Faraday rotations, but will use the transmission matrix rather than the reflection matrix. In plain language, the Kerr effect stems from the unequal propagation of orthogonally polarized fields through a magnetic material. Often, we talk about polarization in the orthogonal linear basis, though in this case, it is more advantageous to use the circular basis. Propagating plane waves are represented by an exponential function containing an amplitude, phase, and frequency,

$$\vec{E}(t) = A_o \hat{E} e^{-i(\omega t - kz - \phi)}, \tag{5.7}$$

where $\hat{E}$ is the direction of polarization, ω is the frequency of oscillation, k is the wavenumber, t is time, z is the propagation direction, and ϕ is the phase shift. By taking the real part of the Euler representation, we can form the typical orthogonal basis of linear polarizations

$$\vec{E}_x(t) = E_o \hat{x} e^{(ikz)} cos(\omega t), \tag{5.8}$$

for a linear $\hat{x}$ polarized component,

$$\vec{E}_y(t) = E_o \hat{y} e^{(ikz)} cos(\omega t), \tag{5.9}$$

and a linear $\hat{y}$ component. Adding these two orthogonal polarizations together would yield a 45° polarization. A simple $\frac{\pi}{2}$ phase shift applied to the $\hat{y}$ basis yields the polarization representation

$$\vec{E}(t) = E_o \hat{y} e^{(ikz)} cos(\omega t - \frac{\pi}{2}), \tag{5.10}$$

and

$$\vec{E}(t) = E_o \hat{y} e^{(ikz)} sin(\omega t). \tag{5.11}$$

Adding this phase-shifted $\hat{y}$ polarized wave to the $\hat{x}$-basis wave, the result is a sum which has a constant amplitude at every value of ωt,

$$\vec{E}_{ccw-circ}(t) = E_o e^{(ikz)}[cos(\omega t)\hat{x} + sin(\omega t)\hat{y}]. \tag{5.12}$$

The above expression represents a counter-clockwise circularly polarized plane wave with amplitude E_o. We can easily represent the clockwise circularly polarized plane wave by simply reversing the sign of the $\hat{y}$ component,

$$\vec{E}_{cw-circ}(t) = E_o e^{(ikz)}[cos(\omega t)\hat{x} - sin(\omega t)\hat{y}]. \tag{5.13}$$

Adding or subtracting these two orthogonal circular polarizations will return the orthogonal $\hat{x}$ and $\hat{y}$ polarizations respectively. Finally, we can apply a delay, in the form of the phase $e^{i\theta}$, to either of the circular polarization equations before adding or subtracting them to each other. Computing the sum of the phase delayed circular basis will still yield a linear polarization, however, it will be rotated by angle θ. If a plane wave propagating through a magnetic material experiences a phase delay in one of its orthogonal polarization directions, wether it be the clockwise (LCP) or counter-clockwise (RCP) direction of an initially linearly-polarized beam, the result will be a rotation of the initial linear polarization. Thus one of the orthogonal circular polarization directions is delayed as it interacts with the magnetic material, yielding the off-diagonal fresnel reflection coefficient. The existence of the off-diagonal elements in the permittivity matrix is a source of this delay in the propagation of the RCP vs LCP polarization, and as stated earlier, will yield a "Kerr Rotation". For a full explicit mathematical treatment of the Kerr effect see references [25, 26, 27].

5.2 Experimental Application of MOKE

In order to bridge the gap between MOKE theory and quantitative magnetic information, one must design a MOKE experiment that yields relevant magnetic data. While the phenomenon of MOKE yields a rotation of polarization from the surface of a magnetic material, we are interested in information regarding magnetization properties of the material in

question. Instead of simply looking at the rotation of polarization from a magnetic surface, which presumably is fixed at a constant magnetization, modern MOKE experimentation uses an externally controllable magnetic field in order to sweep the magnetization of the material from a negative minimum field to a positive maximum. During the reversal of the magnetic field, the rotation of the polarization is monitored and recorded continuously. Because the reflected polarization is proportional to the magnetization within the material, the data produced by a MOKE experiment is able to be identified as the magnetic hysteresis of the magnetic material. As stated earlier in chapter 4, magnetic materials have a type of magnetic memory called hysteresis, in which the magnetism within the material is highly dependent on the previous magnetic state of the material. MOKE implemented with an external sweeping magnetic field allows us to locate the remanence, which is the amount of magnetization when the the external field is reduced to zero, and the coercivity, the external magnetic field magnitude required to reduce the magnetization to 0.

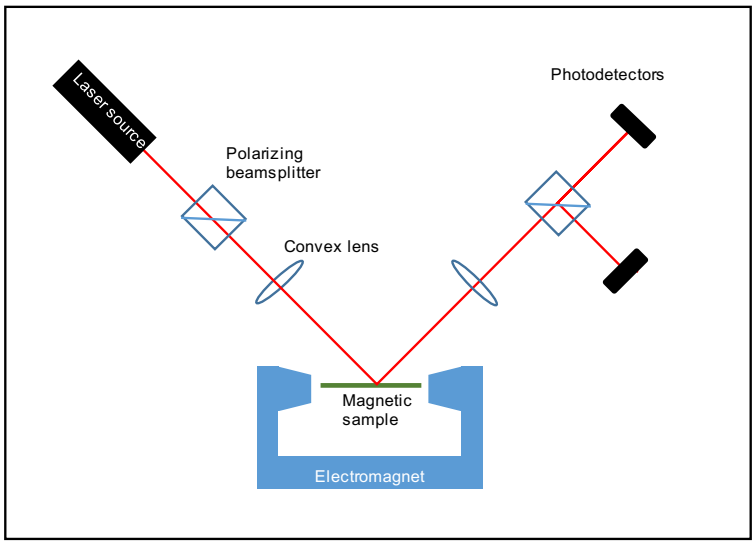

Figure 5.3 An example of a basic LMOKE experimental design. The most important features are included in this image. Starting from the laser source: a polarizer to ensure unidirectional polarization, convex lens to reduce optical spot size, a magnetic sample mounted between poles of an externally controllable electromagnet, a second convex lens to refocus the reflected beam, a second polarizer, and a pair of photodetectors.

At first glance the use of experimental MOKE is simple and straightforward. In chapter 7, we will demonstrate how the very basic MOKE geometry can be modified experimentally to do much more than just monitor the magnetization of a polished magnetic surface. Additionally, we will show how many small difficulties are involved in gathering high quality MOKE data.

CHAPTER 6

FIRST ORDER REVERSAL CURVES

6.1 INTRODUCTION

Before First Order Reversal Curve (FORC) experiments were widely used, an untitled experiment was carried out by Ferenc Preisach investigating the source of the magnetic aftereffect in a theoretical magnetic system containing a distribution of magnetic interaction fields and coercivities [28]. Notably in this experiment, he treated single domain particles as square hysteresis loops which had a distribution of coercivities and "pre-magnetizations" (non-zero loop centers). This type of treatment and interpretation would become the basis of many FORC experiments in the future. Since then, the method has been coined with the official title FORC, and his mathematical magnetic particles have been called "hysterons" (Figure 6.3). Numerous experiments have been carried out by various groups which have sought to quantify and explicity understand the results of these data that come from FORC [29, 30, 31, 32, 33].

6.2 THEORETICAL BACKGROUND

Ordinarily, when gathering magnetization data, as shown in previous sections, a material is magnetically saturated by applying a large external field. From the positive magnetic saturation, we can saturate the same material in a negative direction to see the remanence, coercive field and, negative magnetization values. In a material that is uniform and homogenous, the shape of the magnetization curve will smoothly change from $+H_{max}$ to $-H_{max}$. In the case of a multi-particle system with inhomogeneities, whether they be in-

terparticle spacing, variable sizes, inconsistent coercivities, or different interaction field strength, we will expect to see some sort of non-smooth transition between the minimum and maximum magnetization states. FORC offers an avenue to investigate the source of these non-smooth magnetization transitions, if they are visually obvious, or even discover interactions within a material originally thought to be magnetically homogeneous. FORC data is acquired by a modification of the typical magnetization curve measurement. Rather than simply ramping the applied field between $\pm H_{max}$, we increase the field to $+H_{max}$ but only decrease to a variable minimum field value, known as the reversal field, H_r. H_r can range, in nominally equally spaced steps, between $\mp H_{max}$. The result of this type of magnetic ramping will yield a family of minor hysteresis loops all enclosed within the major hysteresis loop, (MHL) and will resemble the image seen in Figure 6.1.

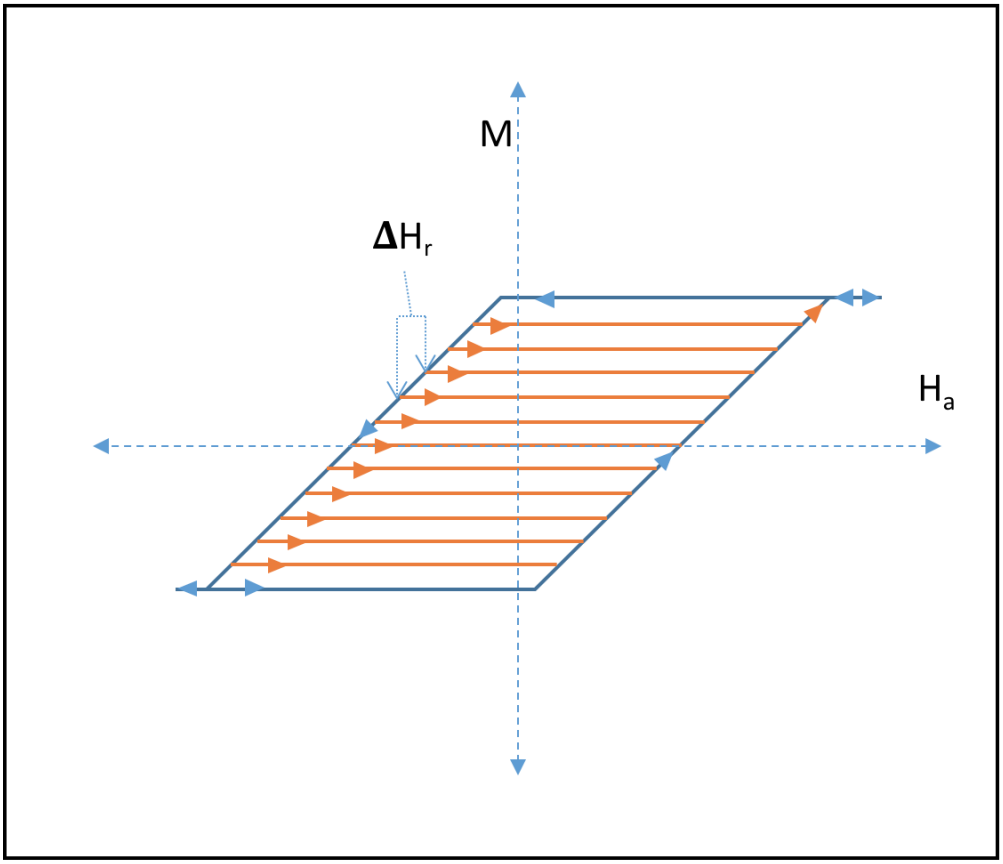

Figure 6.1 A representation of a family of FORC curves for a mixed magnetic state. The blue curve represents the outer major hysteresis loop, while the orange curves represent each equally spaced reversal. In the case of this schematic, the magnetization path of each reversal curve is identical to each other reversal.

Above we have the reversal path from $+H_{max}$ to $-H_{max}$ colored in blue, while the minor reversals are in orange. The number of reversals are limited by the resolution of the magnetic field used in an experiment, but more reversals will yield a higher resolution FORC diagram in future analysis. From the raw data, represented by the family of FORC curves, Mayergoyz [29] showed that given a system following the reversal process shown in Figure Equation6.1, we should be able to create a "Preisach Diagram" by executing the mixed derivative in Equation6.1. This assessment is contingent on two separate conditions being met: the congruency of minor loops, and the wiping-out property (minor loops should perfectly close after one cycle) [29]. This Preisach diagram would follow a certain type of

analysis as long as both of the above conditions were met. In most systems, one or both of these conditions are violated. Thus experimentalists have separated the FORC experiment from the classical Preisach model, and have begun to treat it as a separate experiment with unique interpretations [31]. Using a FORC distribution, there are no restrictions on which type of system this is applicable for, but this leaves much more room for analytical interpretation as to the results of a successful FORC distribution and the resulting implications therein. The mathematical representation of the FORC distribution is

$$\rho_{FORC} = -\frac{1}{2}\frac{\partial^2 M}{\partial H_a \partial H_r}, \tag{6.1}$$

where H_a is the applied field. Equation6.1 is a double derivative, where the first is taken as a typical derivative of the hysteresis along the applied field direction, while the second is taken with respect to the reversal field direction. Even though this is identical to the classical Preisach diagram, the result of the mixed derivative is referred to as a FORC distribution, resulting in a FORC diagram. In plain language, the result of this derivative will reveal "how does the slope at any applied field point change if we start from a different reversal value". As a visual example, we use Equation 6.1 on the cartoon representation of FORC curves in Figure 6.1 which will yield a FORC diagram represented in Figure 6.2. In order to illuminate some of the difficulties visualizing the transition from a traditional magnetization set of coordinates to that of a FORC diagram, the image from Figure 6.1 is shown mapped to the new coordinate system which is used when viewing FORC diagrams.

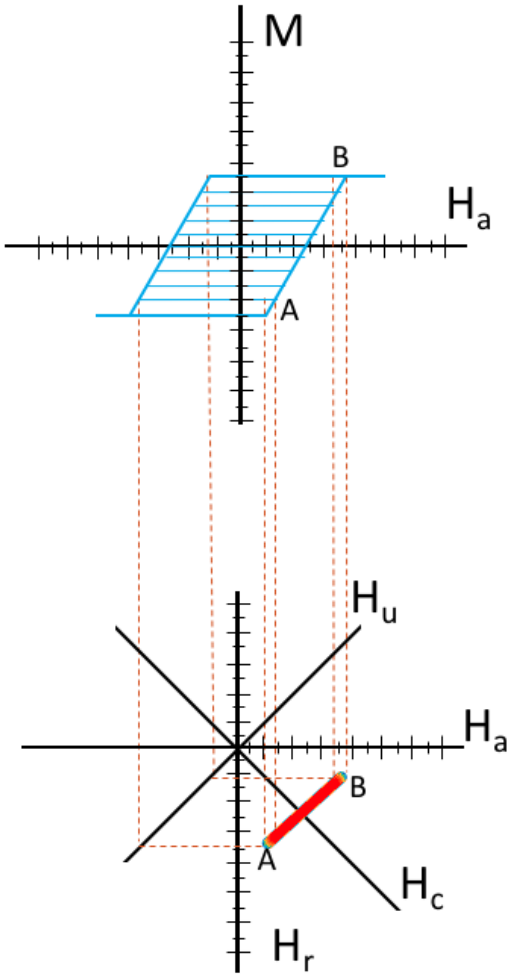

Figure 6.2 The top Figure shows a typical family of FORC curves, similar to Figure 6.1, while the bottom curve shows how the data changes following the analysis of 6.1. Derivatives are taken in coordinates H_a and H_r while the FORC data is represented in a rotated coordinate system H_c and H_u. At instance 'A' in the top graph, we see how the slope of the lowest reversal changes, with the same applied field, with respect to the next reversal above it. This curvature change is mapped to the new coordinate system as a sharp spike, with the color gradient indicating the 3D nature of a FORC diagram. At point 'B', we have a similar situation, where all reversals beyond a certain point will no longer leave the flat part of the hysteresis loop, and, thus will no longer have the slope behaviour seen in previous reversals. The first time this happens, at B, will map the final signal spike to the new coordinate system.

From Figure 6.2, we can see the complexity of even a simple magnetization process. Rather than manually looking for changes in slopes as we change reversal field values, this type of derivative will reveal sharp discontinuities where changes in the reversal process

occur. Translating this mixed derivative to experimental data introduces a significant magnitude of noise if these derivatives are taken literally. Originally introduced by Robert Pike [31], a variable numerical method was introduced to extract the value of Equation6.1 for every point in a family of FORC curves without the need for any actual differentiation. By creating a matrix representation of the full family of FORC curves, we can fit each point, centered within an $N \times N$ subset of neighboring points, with a polynomial surface:

$$K_0 + K_1 H_a + K_2 H_r + K_3 H_a^2 + K_4 H_r H_a + K_5 H_r^2 \tag{6.2}$$

For each point in the matrix representation of the family of FORC curves, the K_4 coefficient will represent $\frac{\partial^2 M}{\partial H_a \partial H_r}$. By increasing or decreasing the size of the $N \times N$ window of points, one can effectively increase or decrease the resolution of the fit by including more points contributing to the polynomial surface fit at the point of interest. Different modifications have been made which have improved resolution by introducing weighting functions when selecting a window of $N \times N$ points, effectively allowing non-integer window sizes [34]. When transforming our raw experimental data into a FORC diagram, we choose an alternative method which uses the literal definition of a derivative with some numerical smoothing in order to extract the the FORC density and compare it to previously pioneered methods. Completing the second derivative is only the first step towards interpreting the results of a FORC diagram. Following formalism introduced by Preisach, we pick a new symmetric coordinate system which is simply a rotation from H_r and H_a in which to present and analyze our data,

$$H_u = \frac{1}{2}(H_a + H_r) \tag{6.3}$$

$$H_c = \frac{1}{2}(H_a - H_r), \tag{6.4}$$

where we have two new coordinates, H_u the interaction field, and H_c the coercive field. These coordinates were already shown, but not introduced in Figure 6.2. We note that the coercive field cannot be negative as the coercivity in a magnetization curve is a non-negative value and H_u can be positive or negative, indicating the direction of interaction

field value. FORC data has historically been very difficult to interpret and there have been various experimentally proven interpretations of the resulting contour plots.

One interpretation of a family of FORC curves is that it is composed of fundamental magnetic particles called hysterons. In earlier FORC interpretations, hysterons were taken to be rectangular hysteresis loops of differing coercivities. From the early interpretation, more fundamental hysterons have been introduced in order to model and characterize FORC diagrams (Figure 6.3).

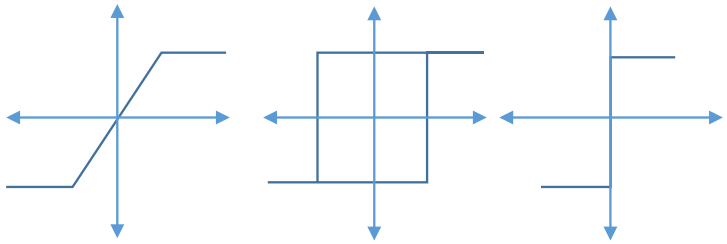

Figure 6.3 Left, middle, right are linear reversible, irreversible, and fully reversible vertical hysterons respectively. Originally, only the middle hysteron was used in describing a system of FORC curves.

Reference [33] treats these hysterons as single domain magnetic particles, following the Stoner-Wolfarth model. These particles are used to simulate large assemblies of like-hysterons. FORC analysis is carried out on these assemblies of hysterons in neutral, non-interacting conditions, and the results are compared to FORC diagrams under the influence of mean interaction fields aligned parallel or antiparallel to the magnetization direction of the simulated structure. The results of these simulations can be summarized with the general rule that a negative interaction field will cause a spreading of the FORC signal in the H_u axis, while a positive interaction field will cause a shift of the FORC peak along the $+H_c$ axis (Figure 6.4). Unfortunately, as many FORC conclusions tend to be, these effects only work for structures created by a singular type of hysteron, and a linear combination of these three hysterons will not yield a FORC diagram which is also a linear combination of the individual FORC diagrams from each of the hysterons. However, this data is illu-

minating for systems where the magnetization process closely resembles that of one of the fundamental hysterons, such as nanowire arrays or thin films [35, 36, 37].

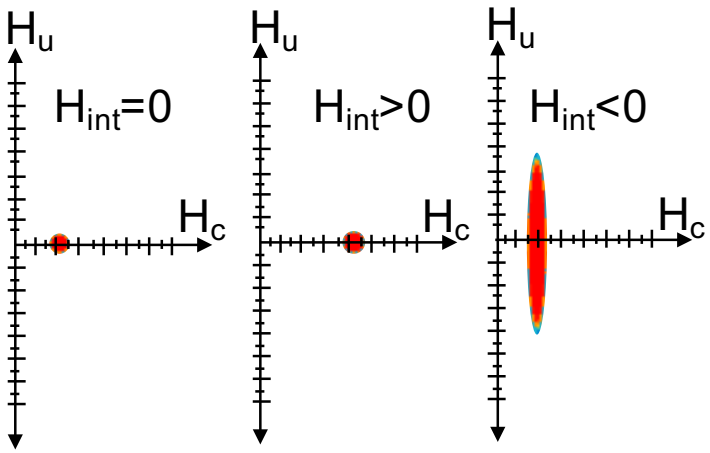

Figure 6.4 Figures adapted from reference [33]. Simulated experimental results of different interaction fields acting on a system of identical, irreversible square hysterons.

One of the modern motivators of the resurgence of FORC investigations is Robert Pike, As stated previously, his contributions cemented FORC as its own measurement rather than being an extension of the classical Preisach model which is bound by certain criterion found in reference [29]. Pike mathematically used the hysteron treatment of multi-particle systems to predict the resulting FORC curves for different types of particle interactions and distributions [38, 31]. Following this mathematical prediction, the authors of reference [39] sought to qualify the predictions made by the mathematical analysis of Pike *et al.* Using samples with known morphology, the authors compared FORC diagrams with the predicted distributions. Figure 6.5 shows an adaptation of some of the relevant results.

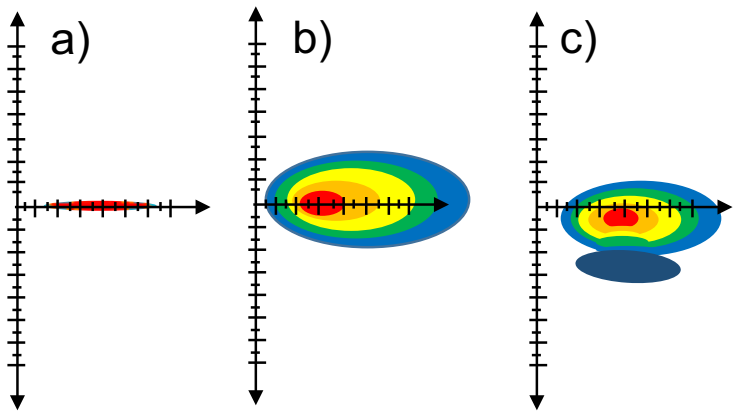

Figure 6.5 Figures adapted from reference [39]. The color indicates the value of the contour, with red being the highest and deep blue being a negative value. All simulations were compared to magnetic sedimentary rocks in which certain distributions of magnetic particles were known to appear. a) single domain noninteracting particles b) single domain particles with randomly assigned interaction values and c) single domain particles with randomly assigned fields and a mean interaction field parallel to the net magnetization.

The investigations from reference [39] were all carried out on naturally occurring magnetic sediment as a means to characterize these materials, and thus the types of features are more general when compared to arrays of nanoparticles/nanopillars or complex interacting magnetic systems. Further features arise when investigating more complex interactions. In many investigations into arrays of nano materials, a characteristic "wishbone" shape was identified in FORC diagrams [40, 37, 35, 32]. A representation of this wishbone shape is seen in Figure 6.6. It is often seen with the vertex of the wishbone off of the $H_u = 0$ axis. Through simulations using the hysteron model, it was found that this shape could be reproduced by a family of FORC curves which were created from a collection of hysterons with a gamma distribution of coercivities and a negative mean field interaction. This may be unsurprising considering that the samples where this characteristic shape is found are arrays of magnetic antidots and nanopillars [40, 35, 37].

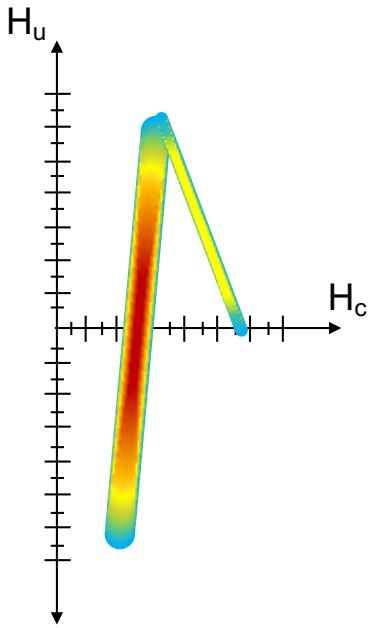

Figure 6.6 Characteristic wishbone often seen in FORC data representing arrays or collections of nanoparticles or nanopillars.

An interesting feature that has been observed in FORC measurements has been the crossing of the MHL by minor reversals, viewed in raw FORC data. First pointed out in reference [40], it has been explicitly identified at least one other time [41]. The former related a similar behavior to that shown in the appearance and disappearance of out-of-plane (OOP) magnetization components where the loops cross, and subsequently cross back over.

As stated earlier, much is still unknown regarding the interpretation of FORC results, but most, if not all, of the more complex features seen are identified experimentally and then artificially created via simulation to confirm the behavior.

CHAPTER 7

EXPERIMENTAL ENDEAVORS

7.1 INTRODUCTION TO THE LAYOUT

This Chapter is intended to cover all of the physical work that was completed and that I have considered relevant. I will lay out my experimental process in an an approximate chronological order of things being completed to illuminate some of the difficulties of the experimental journey, and the necessity and importance of having as much information as possible. The summation of my work culminates with a new measurement technique which is subsequently adapted to magnetic measurement capable of lending insight on a complex magnetic multi-body system.

7.2 THE QUARTER WAVE PLATE AND ITS APPLICATION

Most of my first few months working in the lab was spent understanding basic experimental optics. This includes polarization between cross polarizers, aligning and walking a beam, and most importantly, understanding what a quarter wave plate is and how exactly it works. In theory, a quarter waveplate is not an over complicated optical element, and most descriptions of its function reads similarly to this : A quarter wave plate induces a phase shift in one of the two orthogonal polarization states of the plane wave, thus resulting in a circular polarization state which cannot be extinguished with one polarizer and has an equal amplitude at all polarization angles. Mathematically, as we have seen before, this is as simple as adding a phase shift of $\frac{\pi}{2}$ to one orthogonal state,

$$E_{linear} = E_0[\hat{p}e^{-i(\omega t)} + \hat{s}e^{-i(\omega t)}] \tag{7.1}$$

$$E_{circular} = E_0[\hat{p}e^{-i(\omega t)} + \hat{s}e^{-i(\omega t - \frac{\pi}{2})}]. \tag{7.2}$$

One orthogonal state now lags $\frac{\pi}{2}$ behind the other, so that the magnitude of $E_{circular}$ is a constant in time. A quarter wave plate is an invaluable tool when ensuring linear polarization, due to its ability to induce and remove ellipticity from the polarization state. When an experiment requires a high order of linear polarization, a simple inclusion of a quarter wave plate is a wise choice.

7.2.1 Differential Detection Using a Quarter Wave Plate

When first introduced to experimental MOKE, I was also introduced to the idea of differential detection. Differential detection is used when detecting a very small change in a relatively large signal. In the specific case of MOKE, the effect we observe is a very small rotation of the polarization state of an optical signal. Each of the two differential detectors is calibrated to detect only one orthogonal polarization state and the resulting signals from each detector are subtracted to nominally yield an initial difference of zero. Figure 7.1 shows two ways in which this can implemented in a MOKE system.

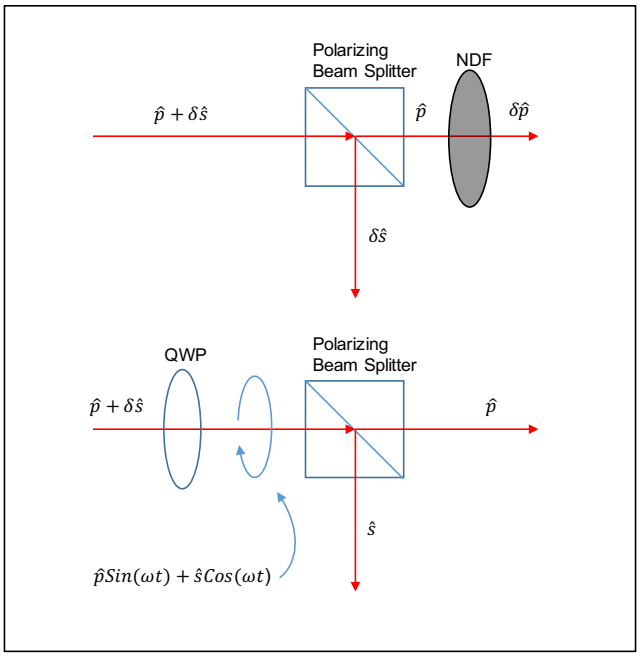

Figure 7.1 Two ways in which differential detection could be applied to our MOKE experiment. In the top image, an initially polarized beam has a small component in the $\hat{s}$ direction and the beam splitter will pick it off into its own detector. The $\hat{p}$ signal will go into a second detector. To balance these, one must use a neutral density filter (NDF) to reduce the $\hat{p}$ signal until it has the same initial amplitude as the $\delta\hat{s}$ signal. this can be particularly difficult if the difference between them is large. Additionally, the NDF will also reduce the effect of the rotation because it reduces all signal passing through it. In the bottom case, we pass the same initial polarization state, but use a quarter wave plate (QWP) to FORCe circular polarization from this linear state. Now the magnitude of the two orthogonal states will be the same due to the nature of circular polarization. A change in $\hat{s}$ or $\hat{p}$ will not be reduced by the NDF.

When a rotation does occur, the originally balanced detectors will see a reduction of signal in one detector and a proportional increase in signal in the other. By subtracting the signals in the detectors we will observe a total signal increase which is double the total change in polarization. If we have an initial state

$$\hat{E}_0 = E_0[\hat{p}+\hat{s}], \tag{7.3}$$

and the effect of our experiment causes the new state to be

$$\hat{E}_f = (E_0-\delta)\hat{p}+(E_0+\delta)\hat{s}, \tag{7.4}$$

then the difference between the two signals (I_s and I_p) will be $\sim 2\delta$. Additionally, any noise source in both detectors will be ideally removed due to the process of taking the difference of signals.

When work initially began on this project, I was instructed to use the lower scheme from 7.1, and was given a QWP to use. As a new experimenter, I assumed that I had the ideal piece of equipment, and that experiments worked as well as theory. Naturally, the first step to making a circular polarized beam was to simply place the QWP at normal incidence with the polarized beam and rotate it until the signal through an initially crossed polarizer is at a relative maximum.

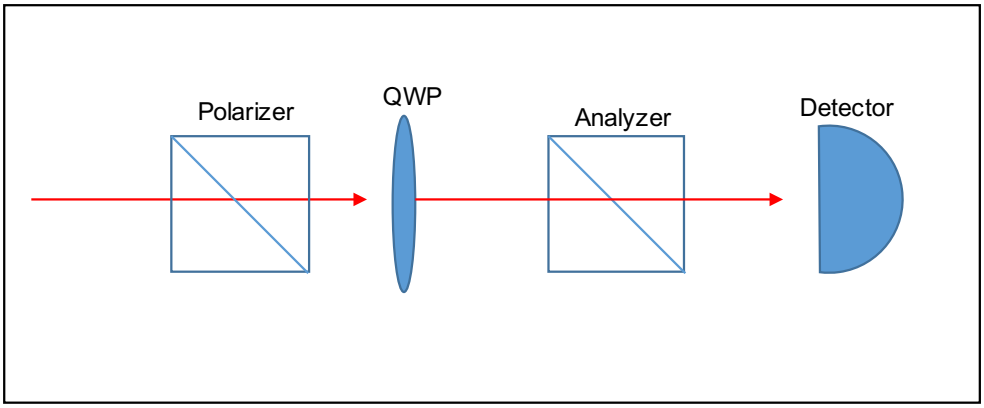

Figure 7.2 The basic initial QWP experiment. Plane polarized light was passed through a polarizer oriented in the $\hat{p}$ plane. Next, the $\hat{p}$-polarized light was passed through a QWP in order to induce circular polarization. The analyzer is set to pass $\hat{s}$-polarized light. As the QWP is rotated toward circular polarization, the initially cross polarized signal reaches a maximum. When a maximum signal is found, the analyzer is rotated to find the degree of circular polarization (equation 7.5).

Following this step, we then rotate the polarizer, and celebrate as the magnitude of the signal remains constant due to circular polarization. This was not the case, however. At best, with this approach, I was only reaching $\sim 85\%$ circular polarization by using an equation of my own design.

$$DoCP = \frac{S_{min}}{S_{max}}, \tag{7.5}$$

where *DoCP* is the degree of circular polarization and $S_{min/max}$ are the maximum and minimum signals observed upon rotation of the initially crossed polarizer. Various repetitions of this same micro-experiment continued with similar results, while I mentally insisted that I was not being careful enough with my alignment, even though optical orthogonality was meticulously ensured. Reading some literature about how the actual piece of physical optics worked, I understood that each QWP is crafted specifically for one optical wavelength and the induced phase shift is achieved by picking a specific thickness of a birefringent material that will pass one polarization undeterred (fast axis) and retard the other polarization state (slow axis). The thickness of this birefringent material is specifically chosen so that the delay is equal to one quarter of a wavelength of light, hence the name quarter wave plate. From here, it was discovered that the QWP was designed for an 780 nm (Thorlabs WPQSM05-780) wavelength beam, and our laser has a peak at $\sim$790 nm with a FWHM of $\sim 25nm$. With a bit more knowledge about the function of the QWP, I was able to diagnose the source of my problems. Furthermore, after an investigation into Hecht's excellent optics textbook, I found that if a QWP is not designed specifically for the wavelength of light being used, a simple tilt of the QWP about one of its own axes (fast or slow) will increase the path length that must be travelled, and thus will increase the amount of retardation induced by the QWP. Following this discovery, I switched the optical source from femtosecond pulses to a continuous output, thus reducing the FWHM of the optical wavelength to a mere few nm from $\sim$ 25 nm. With an optical source that was much closer to a singular wavelength, I used my original experimental techniques and found that with the addition of tilting, I was able to acquire a 95% circular polarization from an initially plane polarized beam. Furthermore, I extended this approach to also include a second detector. Instead of checking the minimum and maximum signal passed by the analyzer, I simply added a detector placed to detect the orthogonal signal as in Figure 7.1b. While this may not ensure perfect circular polarization, the ultimate goal of this approach was to balance two signals for a MOKE differential detection scheme. With this less strict qualification of

circular polarization, I was able to easily balance the two signals with the addition of QWP tilting. The direct application of the quarter wave plate will be described in full context in section 7.4.1.

7.3 Finding the Correct "x" Axis

When viewing magnetization data, the Y axis of a hysteresis loop is typically the magnetization of a material and the X axis is the applied field which induces that magnetization which was shown earlier in Chapter 4. In an experiment such as MOKE, where the instrumentation is not calibrated to output a Y axis of Magnetization, it is often taken to be in arbitrary units with the maximum and minimum values being ± 1 respectively. Since the Y values are magnetization in arbitrary units, the implication is that the X axis must be relevant and accurate in order to convey any amount of correct information. This section will outline the importance and subsequent process of acquiring a properly tracked magnetic field in the context of MOKE, and, more importantly, in the more dynamic case of FORC magnetization.

As the reader will notice in this document, some of the MOKE curves have current as the horizontal axis. Early in experimentation, the importance of the nonlinear relationship between the applied current and resultant magnetic field was not considered. Rather than a paramagnetic field response, our electromagnet displays a ferromagnetic magnetization process with hysteresis and saturation. Figure 7.3 shows the typical response for the electromagnet that we have used.

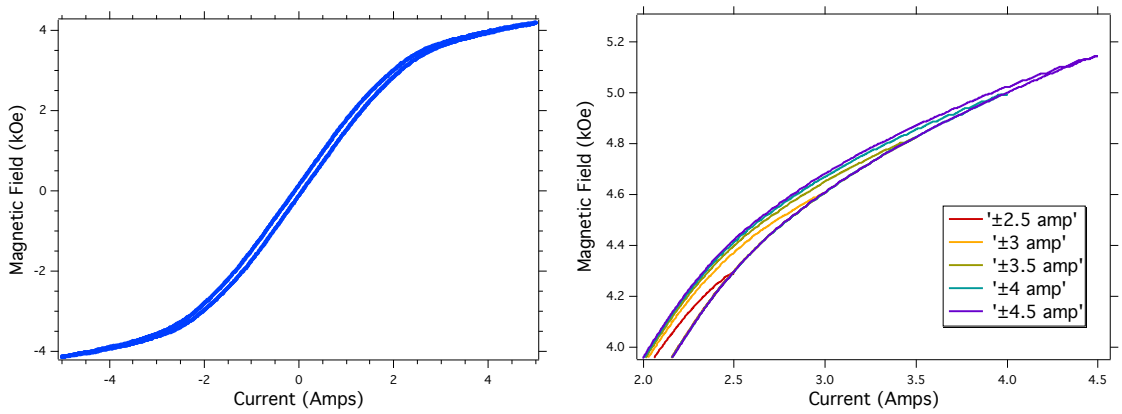

Figure 7.3 Left) The nonlinear relationship between the applied current and the resulting magnetic field. If we use "small" fields (± 1 kOe), we note a fairly linear relationship between H and I. For all other field values the relationship is nonlinear. Right) The minor loop behavior of the magnet itself. Starting and ending at different symmetric current values will yield a different curvature near the start of each respective curve. Using just one lookup table will result in an incorrect field assignment due to the existence of minor loops in the magnet.

Using data which maps I to H, we create different "lookup tables" for each different regularly used current range. For example, with liquid cooling, our electromagnet can run between ± 4.5amps, so we create lookup tables for $\pm 4.5, \pm 4, \pm 3.5, \pm 3$ and ± 2.5 amps. The resulting I to H graph is shown in Figure 7.3b. Due to the ferromagnetic nature of the magnetic core, each of these different endpoints will result in a different minor loop, and thus a different magnetic field for the same values of current. Retroactively plotting data which has been taken with evenly spaced linear current steps will result in a nonlinear set of field steps. The nonlinear relationship is visually obvious, as can be seen in Figure 7.4. Even though the first set of data has evenly spaced steps in current, we see that when we use a lookup table to convert the input current to the realistic field, there is a heavy concentration of points near the saturation field values and a comparatively lower density of points near the zero field value, where curvature is likely to change and remanence takes place. The implication of this graphical representation is that we must strive to take even steps in magnetic field rather than evenly spaced steps in current.

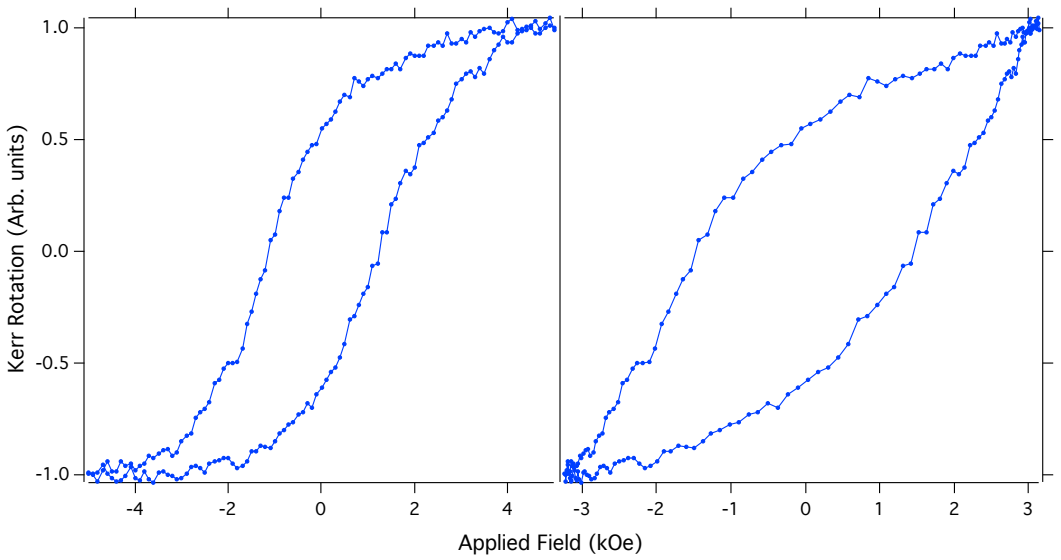

Figure 7.4 Two representations of the same MOKE data. The left hysteresis loop is plotted vs.current, while the right was plotted vs.field, using the same current values and a lookup table.

Functionally implementing these lookup tables into our LabVIEW code was not overly complicated. The method to achieve linear spaced field values is the same for all ranges of current, so this description will be general. A lookup table will have a maximum field (H_{max}) associated with the maximum current (I_{max}). The LabVIEW interface allows selection of "maximum applied field" from a preset dropdown menu, and thus that dictates which lookup table to build a linear set of field steps from. From this point we use the maximum allowable field and the desired number of steps to create a field spacing.

$$\Delta H = \frac{H_{max} - H_{min}}{N} \tag{7.6}$$

where ΔH is the step size, H_{max} is maximum field value from the lookup table, H_{min} is the min field from the same lookup table, typically a sign reversal of the max field, and N is the number of steps intended to be used between max and min field. With the ΔH selected, we then construct an indexed list of field values incremented by ΔH running between H_{max} and H_{min}. For each field value, the lookup table is searched for a field value which is closest

to the value determined by $i \cdot \Delta H$, where i is the iterating index. Because our lookup table has finite points, it is unlikely that we will ever find a desired field value which is actually on the table. Instead, we will often find a point that is "very close" to the point we want. At this point we have two options: use the point closest to the desired field value, or interpolate between small field spacings in order to have nominally evenly spaced field points. If we choose to use the closest point, our job is easily completed with a few simple steps. The desired field value is subtracted from every indexed value within the lookup table of field values. The resulting value from this subtraction which is closest to zero is selected as the field value which is "close enough" to evenly spaced. The corresponding index for that value is recorded and the current of the same index is recorded to build a nonlinear current table which will yield an approximately linear set of field values. If we choose, instead, to insist on evenly spaced field steps, we must do some interpolation. The method used is not difficult to do for a single point by hand, but it ends up being more arduous to create a functional piece of software that works in all cases. The interpolation assumes that the step size between neighboring field points in the lookup table is small enough such that the relationship between current and field is linear. For the following set of equations, we assume that we are looking for a field value, H_0, between bounding field values from the lookup table, H_i and H_{i-1},

$$\Delta H_i = H_i - H_{i-1} \tag{7.7}$$

$$\Delta H_{0i} = H_0 - H_{i-1} \tag{7.8}$$

$$S = \frac{\Delta H_{0i}}{\Delta H_i} \tag{7.9}$$

$$\Delta I_i \cdot S = \Delta I_{0i} \tag{7.10}$$

$$I_{i-1} + \Delta I_{0i} = I_0, \tag{7.11}$$

where ΔH_i is the field spacing between indexed field values on the lookup table, ΔH_{0i} is the difference between our desired field value, and the lower bound of the indexed lookup

values, and S is the ratio which determines in which fraction of the spacing between lookup table values, H_i and H_{i-1}, lies our desired value. ΔI_i is the spacing between the current values which share an index with the field values, ΔI_{0i} is the difference between the smallest current and the desired current, as determined by S, and finally, I_0 is the interpolated current value which will yield the desired field value H_0. Both of these methods will yield a nominally linear set of field points which can be seen in Figure 7.5.

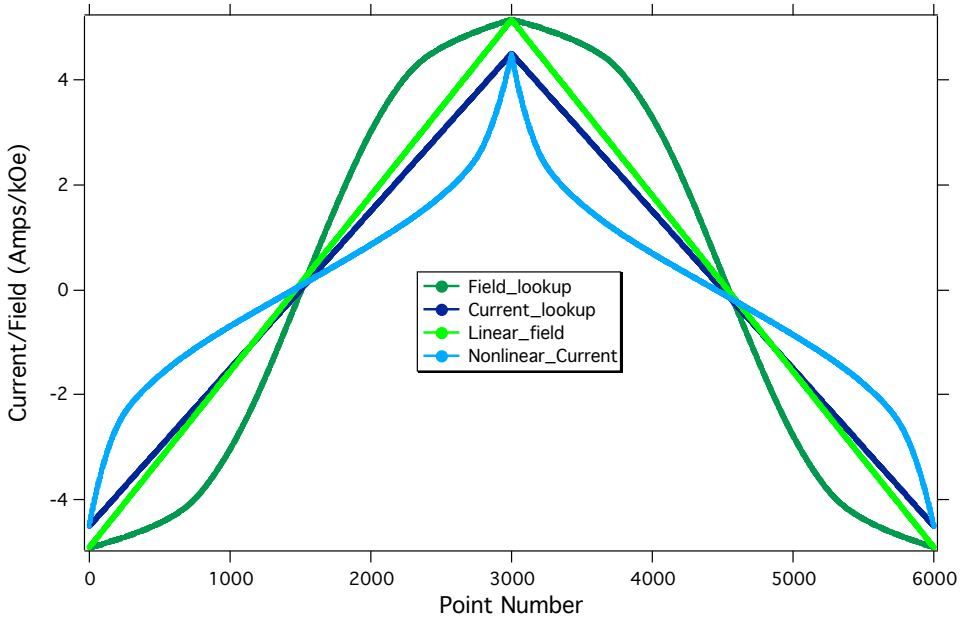

Figure 7.5 Data from a lookup table yields linear current steps corresponding to nonlinear response in magnetic field. Using previously mentioned methods, current becomes nonlinear in order to create linear steps in magnetic field.

Ideally our job is done, and the implementation of the interpolation function can be added to the LabVIEW program which controls the MOKE experimental data acquisition. In order to verify the linearity of our field steps, the calculated field steps were compared to the actual measured steps for different step rates ($\frac{time}{step}$). From these rate tests, we found that there was a nonlinear dependance on the deviation of our expected value from the actual measured value of field. The ideal rate and the deviation from the expected value can be seen in Figure 7.6

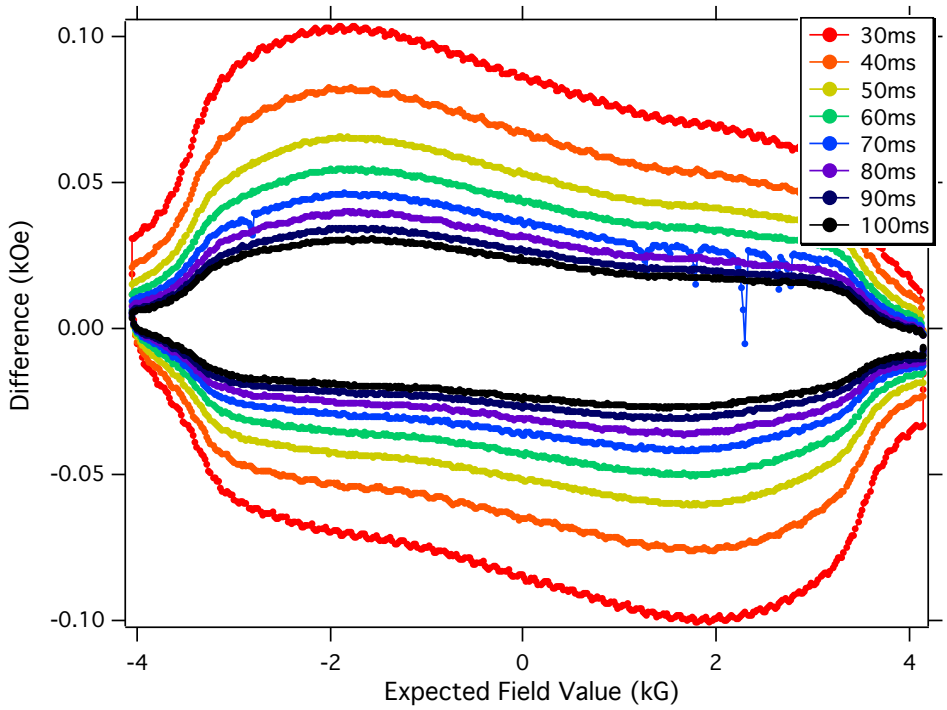

Figure 7.6 Each of the lines above represents the difference between the expected and measured magnetic field values for different wait time between data points. At the highest ramp rate of 30ms/point, the deviation from the ideal field value reaches $\pm$100 Oe at the max and min respectively. As the wait time between points increases, the difference between actual and measured points decreases.

The difference between expected field and measured field, ΔH_{EA}, increased exponentially at each point, yet the exponential dependance was not the same for each different point. We see in Figure 7.7 that we have good exponential fits, but no singular fit function in which we can use to predict this offset for any given single point.

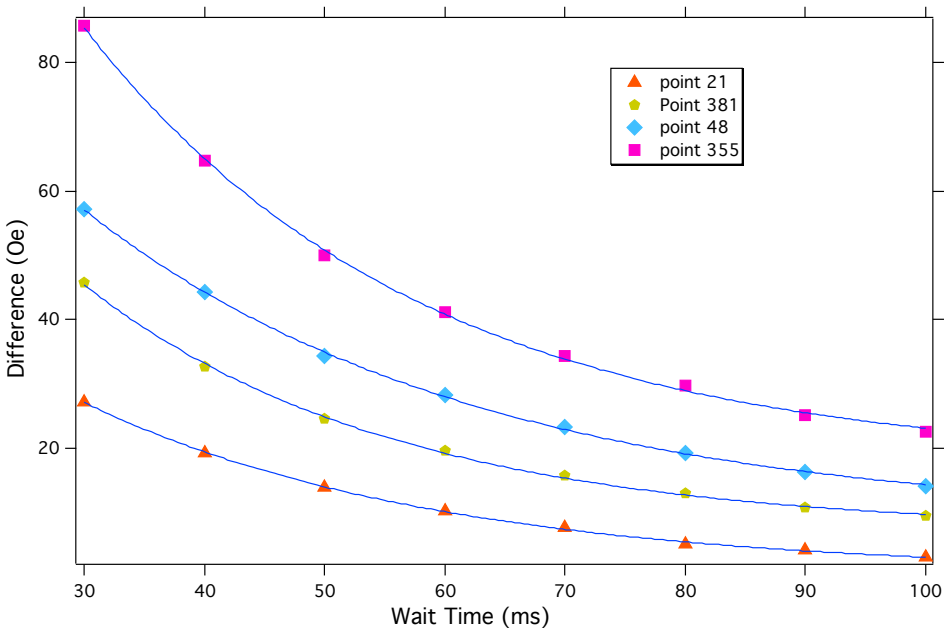

Figure 7.7 Each of the four lines above represent ΔH_{EA} at a single data point, each taken from one of the different lines from Figure 7.6. The horizontal axis is simply the different wait times in which the data points were taken. We see here that the offset from the expected difference of 0 scales exponentially with wait time.

It is clear that there is a nonzero rise time of the current-generated magnetic field produced by our magnet. The important question that must be asked is "Can we simply wait longer to reduce ΔH_{EA}?". Unfortunately in the context of experimental MOKE, as will be revealed later, the answer is "No". The experimental endeavors involving MOKE require several to hundreds of averages in order to produce data with high enough SNR to be useable. If each data point takes $\sim 2-3$secs to complete, and a data set has, at minimum, 100 points, then the total time for a data set of 100 averages will take no less than 5.5 hrs. If we wanted to increase the point density, i.e. the resolution of the magnetization curve, the time increases accordingly. As in many cases of experimentation, the goal is to acquire data that is both quick and relevant. Minimizing the offset by simply waiting longer per point will not satisfy the "quick" condition. With the hope of predicting the rise time of the magnet, we begin to monitor magnetic response, given a discrete step in current. Figure 7.8 shows the magnetic response and two different fits. For an air-gap electromagnet, the

expected dependance on the rise time is exponential [21], but instead we observe that the magnetic rise is fit much more closely with the use of a sum of two exponential functions. Fitting of these magnetic responses were carried out using a home made Igor Pro procedure which can be seen in Appendix A.1.3. According to literature, this type of effect, known as the magnetic after effect, is common in iron core magnets, which have a large amount of magnetic material to rotate, though it is not exclusive to physically large magnets [21, 28].

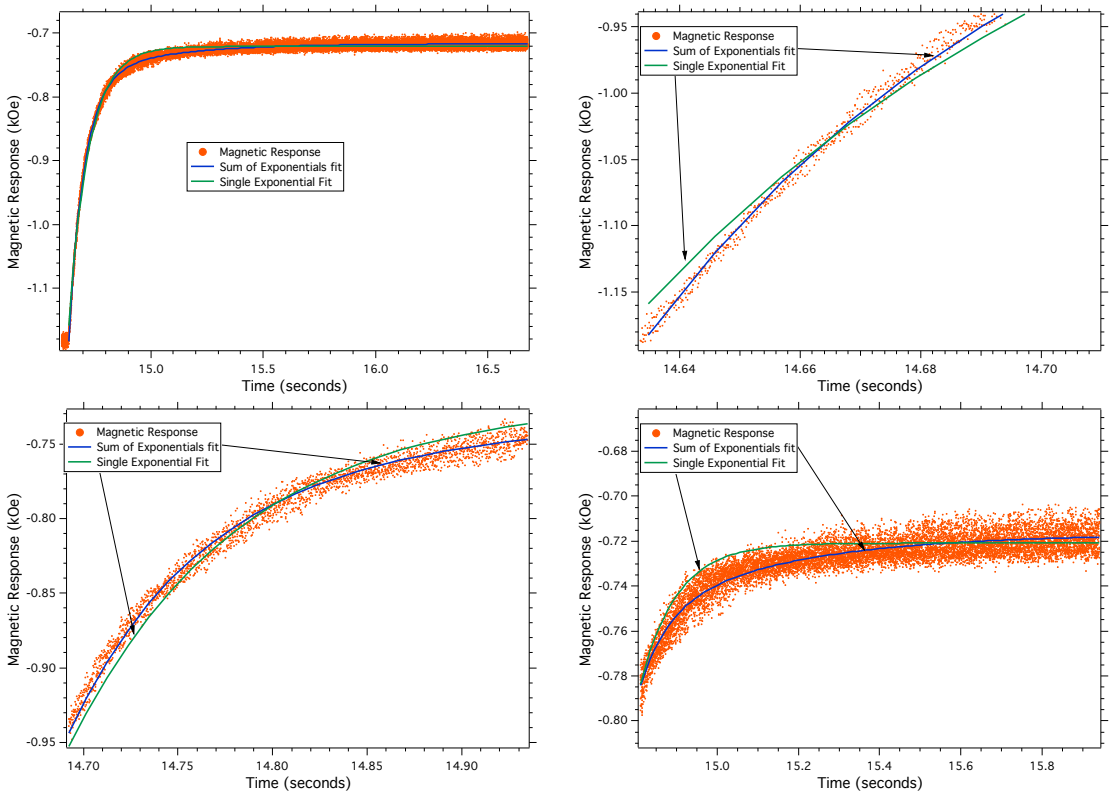

Figure 7.8 Shown in the largest image is the magnetic response to a discrete change in current. The magnet has an exponential-like response to this discrete change in current. The expected fit would be a single exponential. We see that the resultant fit of the single exponential fails to fit the raw data at several parts of the graph.

At this point we are able to verify that the particular curvature of the magnetic rise time is indeed due to the magnetic after effect, and according to the graphs in Figure 7.8 the time to reach $\sim 95\%$ of the intended value is $\sim$2 full seconds. This means, that if we want to continue to use lookup tables to find linearly spaced field values, we must wait two seconds

per data point, which is unacceptable given the need for improved resolution via averaging.

This method of axis calibration seems to be a dead end due to exponential rise times and inconsistent time dependent deviations from expected values. Beyond having an accurate field axis for the sake of being complete, we seek this type of precision for the facilitation of FORC measurements. This particular measurement requires differentiation of raw data in both the X and Y directions. Ideally, points in X lie on an evenly spaced array to facilitate this differentiation. If we take fewer data points between $\pm H_{max}$, the slight offset of our actual field values is negligible, but this also lowers the resolution of our resulting density function (as described in section 6). We seek to have a high density of evenly spaced field values with corresponding MOKE responses at each curve, and therefore, we must change how we construct our field axis.

For all lookup tables, we use a Gauss probe capable of outputting 1000 samples/sec. Initially field data was acquired using (F.W. Bell Model 6010 Gauss/Tesla Meter), but after a mishap which led to the demise of the first probe, it was replaced with a different model (Meggit 5180 Gaussmeter). The probe has a BNC analog output which is able to represent magnetic field with equivalent voltage. For example, 1k Oe would output 1V or a different voltage scaled by a factor of 10. The rate in which the probe outputs is much faster than it is ever acquired when constructing a lookup table because the faster speed would just cause a larger offset due to the magnetic after effect, if we were to make a lookup table very quickly then use it later at a different rate. As it stands, we measure Kerr data as a function of ideally linear field steps determined from a pre-made lookup table. Instead, if we could monitor both vs. time, we can remove time as a common axis and acquire MOKE vs. H_{app}. The Lecroy Waverunner 64xi oscilloscope is capable of 5 gigasamples per sec acquisition. The capabilities of this instrument makes the 1000S/sec output of the magnetic probe seem comparatively slow. Using the analog outputs of the magnetic probe and Lock-in amplifier, we begin to monitor both magnetic field and Kerr signal through the oscilloscope. The only consideration that must be accounted for at this

point is the integration time of the lock-in amplifier. Ideally, the integration time should be at least 3x less than the rate of data collection. Lowering integration time increases noise, but increases point density and speed, so it it is a necessary step. The time constant for MOKE via the oscilloscope is changed to 300μs rather than 30ms, which was the previously used time constant for MOKE using lookup tables. Using a 10 second window, we are able to input a large volume of simultaneous MOKE and magnetic field data through the oscilloscope (Figure 7.9).

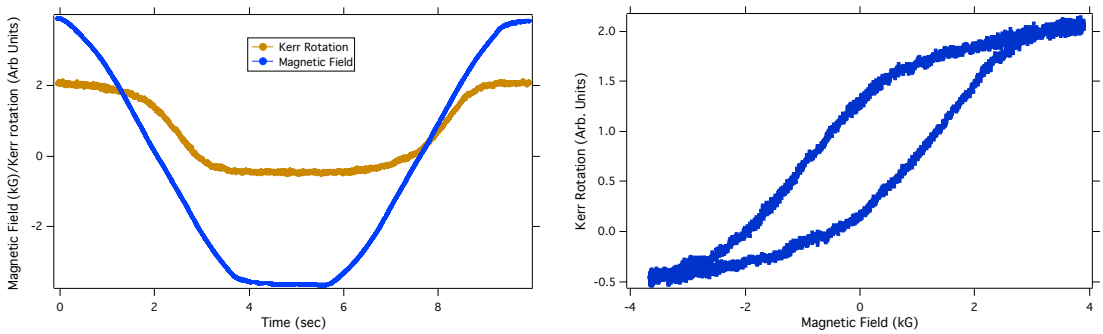

Figure 7.9 Data acquired through the oscilloscope for both magnetic field and Kerr rotation. The first graph shows the data as seen on the oscilloscope, which is plotted vs. time. For each point in time, Kerr rotation is plotted against magnetic field yielding the graph in the right image.

With the pivot to using the oscilloscope to input and save data, the function of LabVIEW is simplified to only control the magnetic field values with no function of data recording. By changing from a pre-made lookup table to an *in-situ* measurement of field and Kerr rotation, we are able to take advantage of the density of points to produce the above graph with a full 10,000 data points. This is relevant when seeking points with nominally evenly spaced field values. With such a small change in field per point ($\sim$15 Oe), we can now employ similar interpolation techniques as we had done for the lookup tables before (starting with equation 7.7).

With every new solution come additional experimental problems to overcome. The magnetic probe does not produce noiseless output. Just like our MOKE signal, we employ

the same number of averages to the signal output from the magnetic probe. Rather than simply reducing the noise of the magnetic probe, the averaging revealed some artifacts of bad programming within the probe itself (Figure 7.10).

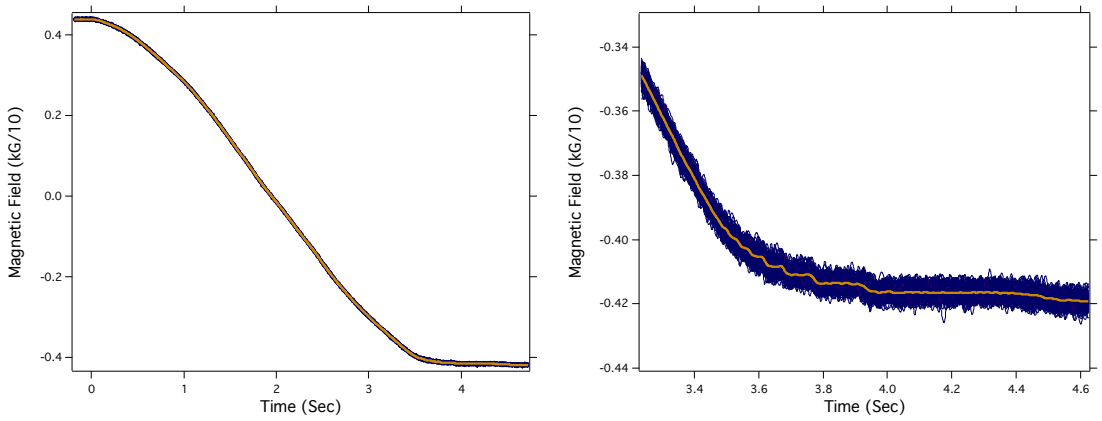

Figure 7.10 Due to the fact that there exists noise in the magnetic probe, we also average the output of the field, as well as the Kerr rotation. On the left is a typical set of magnetic field values starting at $+H_{max}$ and descending to $-H_{max}$. The blue data is 200 identical runs, while the orange line within represents the average of those runs. The Figure on the right shows that upon averaging, this data becomes less noisy, but shows discrete steps near the negative minimum, where $\frac{\partial H}{\partial t}$ is low. These steps are consistently 25 Oe.

Investigations into the origin of these discrete hidden-by-noise steps reveal that it was, "built into the firmware for an unknown reason", according to the supplier. The existence of these steps means that we are now, once again, unable to trust the values of our magnetic field, but this time, it's just at the endpoints where the change in field is small with respect to time. With the knowledge of the functional form of field rise time, the decision was made to replace the slowly changing portions of the magnetic field curves with an appropriate fit (Figure 7.11). Initially the double exponential fit was used, but observation of the fit coefficients showed little to no dependence of the second exponential in this slowly changing subsection of the graph. Fitting was facilitated with an Igor Pro procedure which allows the selection of a portion of the graph to replace using three cursors (section A.3.5 in Appendix). The first is the start, the second is the end of the fit, and the third is the additional points beyond the fit, in which we will replace points according to the fit function.

The choice to include a 3rd cursor comes from a suspicion of how the discrete steps in the field graph originate. Since they are discrete, it is likely that when the magnitude of the field passes a certain threshold between consecutive 25 Oe steps, it is placed into the next 25 Oe field value. This can be seen in the right image of Figure 7.10, where there are large flat areas and then a discrete 25 Oe step.

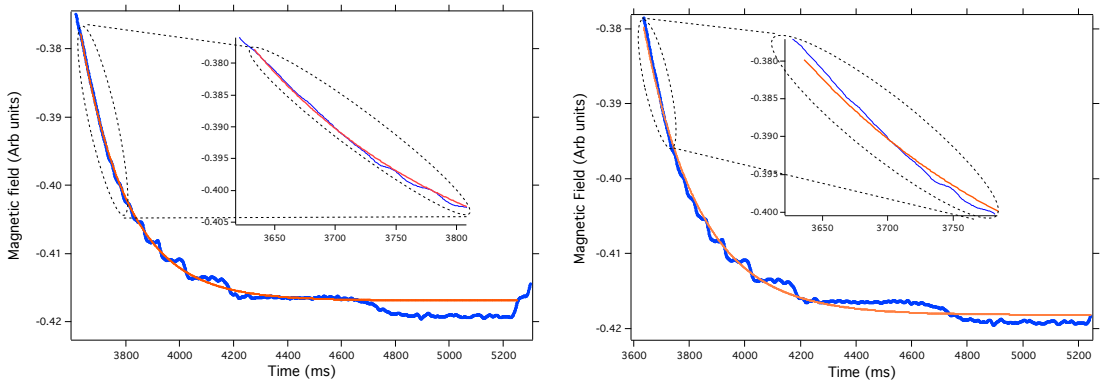

Figure 7.11 Using a single exponential fit, we are able to approximate the behavior of the magnetic field as it approaches saturation. The graph on the left is fit between 3700 and 4700 ms, with the fit function extended to 5250 ms. The graph on the right has a fit between 3650 and 5200 ms, with the fit extended only a few ms further. Although it is less apparent from a macroscale view, the insets show how well the fit matches up to the smoothly changing data.

We see via the insets that the areas where the data is nominally flat are likely due to some conditional statement within the magnetic probe software, and not representative of an actual 25 Oe step to that value. By selecting a point near the final discrete step (~4700ms in Figure 7.11), rather than at the end of it, we see that the fit flows seamlessly into the real data. We use this technique in conjunction with our knowledge of the exponential behavior of the magnetic response and LOESS smoothing of the averaged magnetic field to correct the errors caused by the questionable Gauss probe software. With these methods we are now able to realistically represent the applied magnetic field and, due to the large density of points, we are even able to extract fewer points in order to have equally spaced magnetic field steps. Additionally, using the an oscilloscope allows us to speed up an individual

MOKE run by now being limited by the size of the window on the oscilloscope, which was often 10 seconds with a 2 sec wait to allow the trigger to reset. For regular MOKE applications we have improved both the accuracy of the field values and the rate in which data can be acquired.

7.3.1 Preparing Asymmetric Reversals

In future experimental sections we see the success of a novel MOKE method which is able to facilitate the acquisition of low-noise magnetization curves on aligned Janus fiber agglomerates. From low noise magnetization data acquired using MOKE, we decide to use MOKE to acquire FORC, as described in Chapter 6. As a reminder, the first step of a FORC analysis is gathering asymmetric magnetization curves. In fact, the steps of applied field tend to look like the image shown in Figure 7.12.

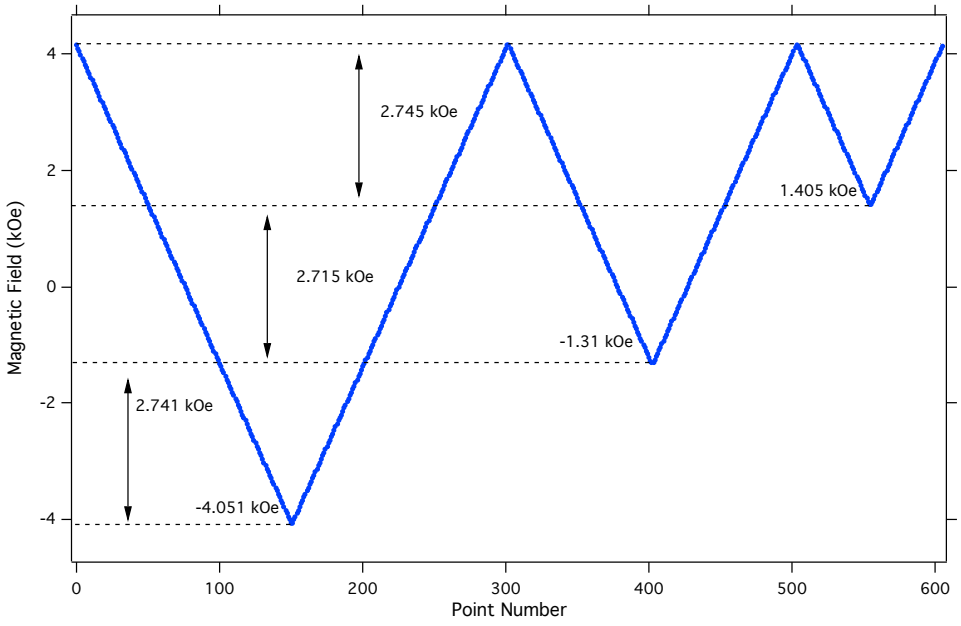

Figure 7.12 The shape of a typical X axis for a single full FORC with only 3 reversals. From start to finish, the field reaches a reversal point (H_r) and returns to H_{max}, with each subsequent reversal reaching a less negative value. Nominally the H_r values are evenly spaced to facilitate a set of derivatives on the future FORC data. Magnetization data is recorded thoughout the process.

Part of the future FORC analysis requires that two separate derivatives are taken on the data, as stated in Chapter 6. The way we parametrize FORC data is with the two field values H_r and H_a, where the former is reversal field, and the latter is applied field. For example, in the graph above, if we were to refer to the point with $H_r = -1.31$ and $H_{app} = 0$, we would be referring to point 440. Basically, the reversal field values serve to separate each of these "minor" loops. Our future derivative will involve the typical derivative with respect to the applied field (the typical x axis for a MOKE loop), but the second derivative is with respect to the reversal field value, comparing how the typical change of magnetization with respect to field, $\frac{\partial M}{\partial H_a}$, changes between different minor reversal loops. In order to facilitate these derivatives with minimum interpolation (we end up having to do quite a bit of it anyway), our goal is evenly spaced values in both of the differentiable variables. As shown in Figure 7.12, the reversals are fairly evenly spaced, but not exactly, while the linear field steps in this image are calculated from a lookup table to be evenly spaced. From the previous section, we know that the sole use of a lookup table will result in varying rate-dependent offsets. The accuracy of the evenly spaced field points is corrected with the same method as the non-FORC applied fields. Extra care, however is used in programming FORC data acquisition, which will be discussed further in the FORC experimental section 7.6.

7.4 MOKE Measurements on Permalloy

As described in the MOKE section, we can use MOKE to measure the magnetization process in a magnetic material. The final goal for this measurement technique is to measure the aligned multiferroic Janus fibers mentioned earlier. Following the tremendous difficulties experienced with the QWP, I decided that a slow, methodical approach was wisest. The first sample to be measured with a MOKE experiment was 200 nm thick permalloy films. These permalloy films had in plane easy axis magnetization and a relatively large Kerr response. The geometry of the experiment can be seen in Figure 7.13

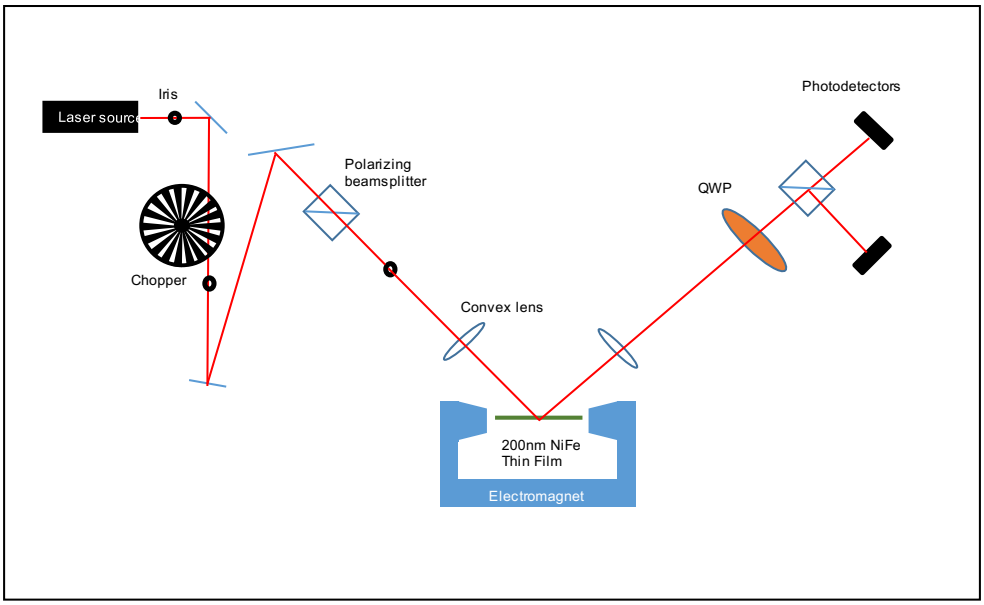

Figure 7.13 The original MOKE geometry for measuring NiFe thin films using the differential detection scheme from Figure 7.1b. The optical chopper is used as a reference for a lock-in amplifier. The various irises are in place to shape the beam profile to ensure the cleanest wavefront possible. The first and second polarizers are both set to transmit $\hat{p}$ polarized light. The two convex lenses are used to focus the beam spot and subsequently gather the diverging beam. The electromagnet is controlled via LabVIEW software, which sweeps the magnetic field between $\pm H_{max}$. The QWP FORCes the initial polarization state at the detectors to be circular so that the difference between the signals of both detectors is negligible.

Initial hysteresis loops were acquired by using a long time constant of one second on the lock-in amplifier. The time constant essentially allows averaging for that duration before resolving the value for output. The MOKE geometry being used falls into the LMOKE scheme, and thus we are probing the magnetization in the plane of the thin film. Since the thin film has a very strong easy axis in plane, due to shape anisotropy, all of the full thin-film results were very rectangular. At this point in the experimental MOKE process, a reader will note an X axis in units of current, rather than magnetic field. This topic is addressed in detail in section 7.3.

Throughout this early experimentation, using differential detection, Two different brands of detectors (Thorlabs DET36A, HINDS DET-200) were used with a different electronic

gain associated with each one. As an extra step, before each experiment, using a variable neutral density filter, the signal into the detector with the larger gain was attenuated to match that of the detector with the smaller gain. After doing this process numerous times, the problem was circumvented by simply searching for two identical detectors.

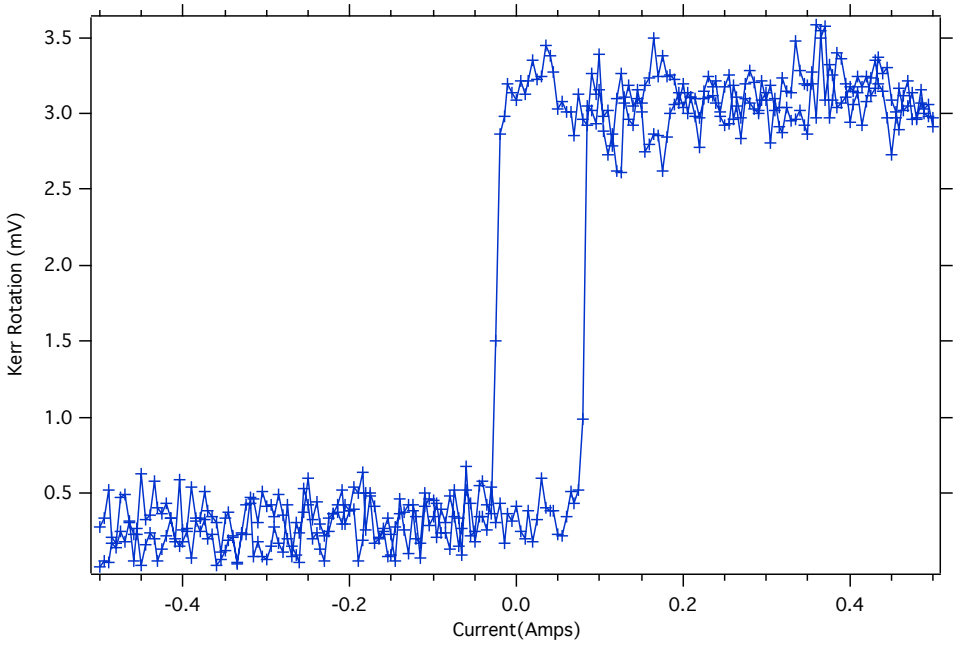

Figure 7.14 A representation of typical MOKE using NiFe as a sample. Note the easy axis magnetization due to the fact that we are doing LMOKE, and thus, probing the in-plane magnetization. The levels of noise are quite significant even when the magnetization has saturated.

MOKE was chosen as a viable magnetic measurement technique based on literature, specifically reference [42]. In this particular journal publication, the author uses MOKE to measure hysteresis of 200 nm wide, flat permalloy wires. After many hours of using the optical scheme outlined in Figure 7.13 to measure the permalloy thin films, the contents of reference [42] were revisited, and it was noticed that we were implementing the quarter wave plate incorrectly. Rather than inducing circular polarization, the ideal use of the QWP was to correct any reflection-induced ellipticity. This paper also argued that, rather than cross polarizing a polarizer and analyzer, the best signal to noise ratio (SNR) would

come from an $\sim 97\%$ initial extinction of the reference signal. With the polarizer and analyzer oriented such that they are almost fully cross polarized, one detector is receiving a much larger signal than the other. The author suggests using a neutral density filter to remove excess signal in order to balance the two detectors. The experimental design was modified to implement this SNR maximization (Figure 7.15a) and the resulting data was visually much higher in SNR (Figure 7.15b). At this point, a MOKE loop would take approximately 5-10 minutes to complete, depending on the step size. For future, greater SNR MOKE, this time would have to be reduced in order to employ averaging with the intent of improving the SNR.

To ensure that signals were nearly equivalent, the signals from both detectors were passed through a dual channel oscilloscope (Lecroy Waverunner 64xi). When one of the signals was magnitudes larger than the other, it was helpful to have a visual representation of the signal as the lock-in amplifier only had a finite signal range. Additionally, using the math functions within the oscilloscope, it was discovered that there was a 'small' ($\sim \frac{\pi}{9}$) phase difference between the two signals, which prevented perfect cancellation of the signal when considering the signal has a magnitude and frequency. I was overzealous about small things like this, and decided to ensure that these signals would fully cancel each other in amplitude and phase. This was done by building a variable high pass filter. The relevant experimental optical Kerr signal was modulated by an optical chopper, and therefore the form was sinusoidal. This high pass filter would delay the phase of one of the signals depending on the variable resistance(0-5kΩ). This eased my peace of mind, although in future experiments, we will see that it was ultimately unnecessary, as the disruption caused by the very slight phase difference was less important than the noise gained by putting this circuit in-line with my signal. Thankfully, when using permalloy thin films, the SNR was simply so high that single Kerr loops still had a relatively high SNR without the use of additional add-ons.

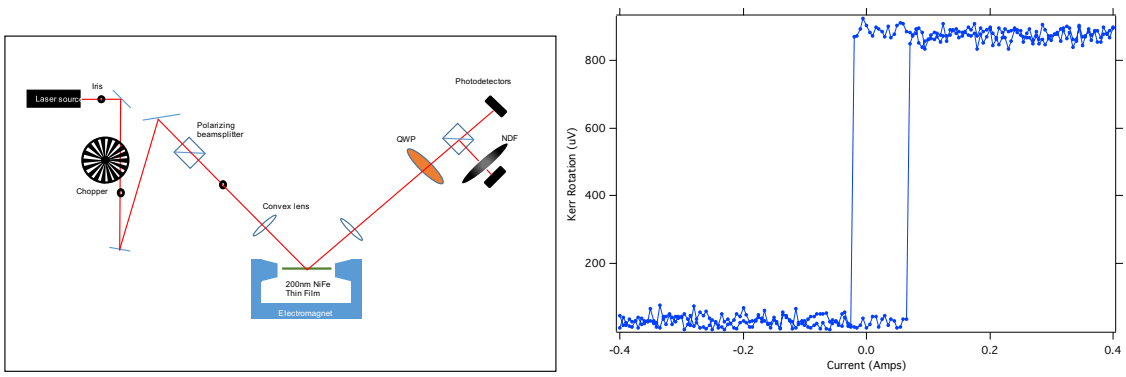

Figure 7.15 The modified MOKE geometry. The inclusion of the neutral density filter (NDF) fundamentally changed how the QWP functions as part of the experiment. The result is a graph possessing a visually larger SNR when compared to previous data in Figure 7.14.

We will take a slight aside here to discuss the motivating factors that led me to create and measure permalloy microwires using MOKE. Following the excellent success of permalloy thin films shown above, MOKE measurements were attempted on nanofiber agglomerates. As a cautious experimenter, pure CFO nanofibers were used initially, rather than the Janus vairety, with the hope that a material that was fully magnetic would be easier to extract signal from, rather than one that is nominally 50% magnetic [43]. Using the same methods for our excellent SNR on permalloy, I attempted many unsuccessful CFO MOKE runs. Noise levels were large enough that no Kerr rotation was obviously detectable. The few successes were not reproducible and will be fully discussed in 7.4.1. These CFO were aligned with a uniaxial field with methanol as the alignment medium. Fibers were left in the external field until all methanol was evaporated. Compared to a thin film, there are obvious differences in geometry. A thin film is "infinite" in the plane of the substrate, and a thin film has negligible surface variation. An aligned nanofiber agglomerate is nominally 2-10μm in diameter and is the formed through the aggregation of many individual fibers. Were these experiments not working because the width of these fiber agglomerates were too small for my MOKE? Were they failing to produce a Kerr rotation because the aligned fibers formed such a topographically diverse surface? In order to test one of these two

potential sources of experimental failure, NiFe wires with dimensions similar to that of aligned CFO were created.

Permalloy wires of various diameter and edge quality were created to eliminate the suspicion that the few μm diameter was the cause of low SNR of CFO measurements. Using photolithography and physical vapor deposition NiFe wires were created and deposited onto glass substrate. Atomic FORCe microscopy was performed on these wires to verify the surface quality. Figure 7.16 shows optical and atomic FORCe microscopy images of NiFe wires

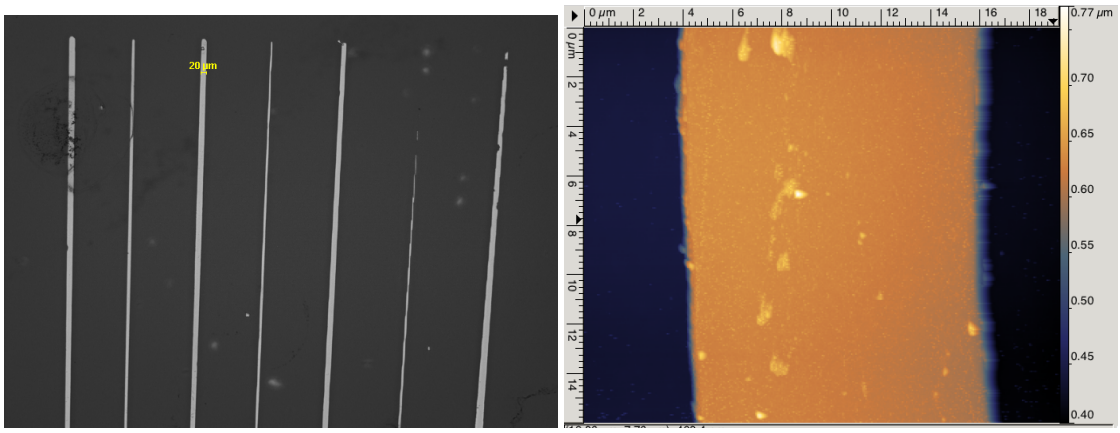

Figure 7.16 Left) Dark field optical microscopy of NiFe wires created with photolithography. Right) Atomic FORCe Microscopy (AFM) on the surface of one wire to confirm width, height, and surface smoothness.

Glass was chosen as the substrate for the NiFe wires for the simple reason that the nanofibers are also aligned on glass. Throughout the entire experimental process using glass as a substrate has been a boon and a curse. Glass will transmit $\sim 90\%$ of light at most incident angles and even more if the specific polarization is chosen. This helps tremendously to reduce specular light reflected from non-magnetic sources. On the other hand, glass has two surfaces of reflection: the front pane and the back. Due to the fact that the magnetic sample surface only occupies a small portion of the optical spot size, we gain a relatively large reflected signal from light incident on the sample substrate. When

considering the reflected beam of light from a magnetic material, it is helpful to consider the column of light as many individual rays, which are originating from different surfaces. Those originating from the glass substrate will nominally have the same polarization as the incident light, and those originating from the magnetic material will have a polarization state which is rotated slightly with respect to the initial state. I am not the first MOKE researcher to encounter this particular issue [44]. The MOKE-specific equation, which quantifies a relative Kerr rotation, adapted from reference [44], is,

$$\Theta_{Kerr} = \frac{\Delta I_{Kerr}}{I_{sub} + I_{mag}}, \tag{7.12}$$

where Θ_{Kerr} is the Kerr rotation, ΔI_{Kerr} is the total signal change from the Kerr rotation, I_{sub} is the intensity of light reflected from the substrate, and I_{mag} is the intensity of light reflected from the magnetic material. In general, we seek to minimize the I_{sub} term, so that the whole Kerr rotation increases. This fraction can be maximized in a few ways. The first, and most obvious of which is the reduction of the optical spot size. By reducing the optical spot size, the FWHM will ideally approach the area of the nanomagnet such that there is no substrate reflection. One can also apply an antireflective coating to the sample substrate. Lastly, one can increase ΔI_{Kerr} by applying a specific dielectric coating to a material so that there are multiple internal reflections that serve to multiplicatively increase the numerator of this fraction. Since this detour to NiFe wires was not the ultimate goal of our research, the dielectric coating, while seemingly easy to use for NiFe, was not feasible with the sample preparation used nor the inhomogeneity of the fibers. The option of AR coatings to the sample substrate was not available in our lab, and as I have already stated earlier, our optical spot size was at its reasonable minimum limit. For the future goal of measuring aligned fibers, it seemed like none of these particular improvements would be worthwhile for both the current problem of improving SNR for wires and the future goal of measuring magnetization of aligned nanofibers. Separate from these specific MOKE-related signal improvement techniques, it is known that the SNR of raw signal can be improved by $\sqrt{N}$, where N is the number of repeated and averaged trials that have been

processed [45]. Until now, SNR has been high enough that the shape and parameters of MOKE data has been easily distinguishable from a single MOKE acquisition. Figure 7.17 shows the Kerr rotation of a permalloy wire. The SNR is much lower than that of the NiFe thin films, probably due to the fraction of optical spot size ($\sim 20\mu$m FWHM) compared to the few micron diameter of the fiber shown below. The LabVIEW code handling all of the MOKE-related measurements, at this point, is modified to allow continuous collection of consecutive MOKE data. This data is post-processed in Igor Pro (section A.2.1 in the Appendix), in which data is averaged and normalized (Figure 7.17). It is apparent that SNR is tremendously improved by a simple averaging scheme.

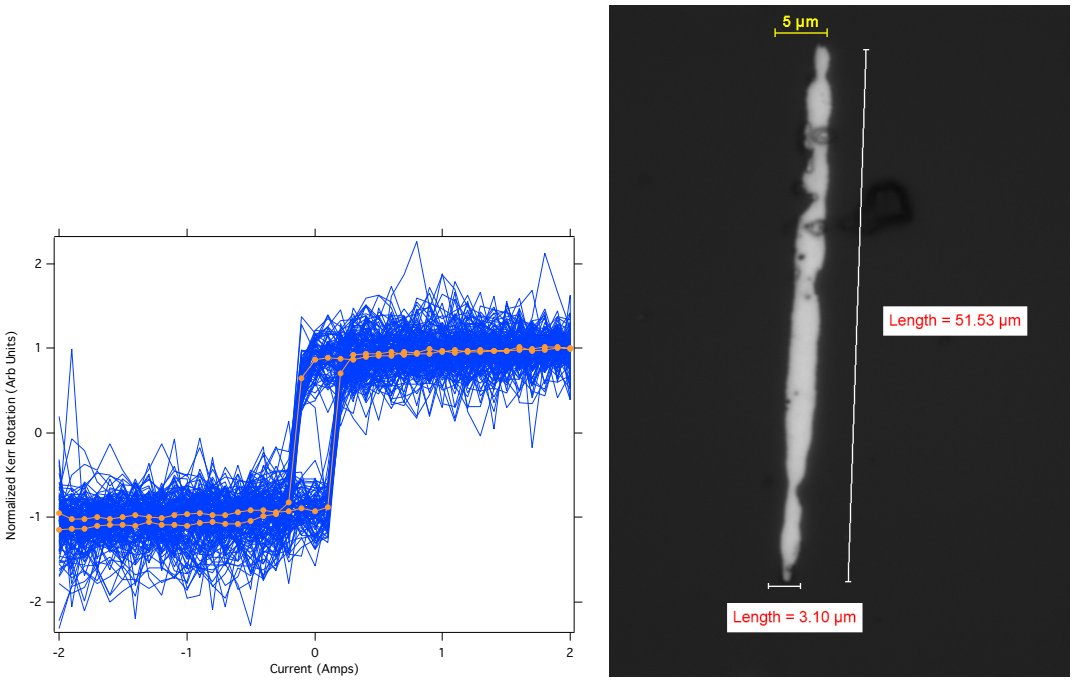

Figure 7.17 Left) Kerr Rotation data taken from the wire shown on the right. The blue data are single Kerr rotations, while the orange data are the average of the blue data. Note the vertical axis has been normalized such that the data runs between ± 1. The average standard deviation (noise) from the individual runs is initially $\sim$0.183 arb units. After averaging, the noise is reduced to $\sim$0.0174 arb units. This is a $\sim$10.5x reduction of noise. The expected reduction was $\sqrt{82}$ which is $\sim$9.1x. Right) The actual wire from which this data is taken. The dimensions and edge profile are comparable to a small aligned fiber agglomerate

After acquiring a significant amount of consistent MOKE data on the NiFe wires, I was confident that the size of nanofiber agglomerates should not be a problem, given the quality of averaged hysteresis loops. Note that the signal quality returned to that of the NiFe films only after averaging. Moving forward, averaging will be continuously implemented to have the possibility of seeing relatively low-noise hysteresis loops. Even with the significant back reflections from the glass substrate, the process of averaging multiple data sets was able to reduce the noise to a point where noice can be easily distinguished from internal magnetization processes. Averaging, normalizing, and noise measurements were completed via Igor Pro procedure (section A.2 in Appendix).

7.4.1 MOKE ON CFO NANOFIBER AGGLOMERATES

When given the option to use a fully magnetic or a Janus magnetic material in a highly unrefined magnetic measurment experiment, the careful experimenter will choose the fully magnetic one. Compared to Janus nanofibers, which are nominally 1/2 magnetic (1:1 BTO:CFO), CFO nanofiber agglomerates are pure CFO [43]. These externally aligned agglomerates were prepared on glass slides according to the alignment process in section [11]. As stated earlier, the glass substrate produces back reflections from multiple surfaces, which can comparatively drown out a Kerr signal which originates from the sample surface. The option to prepare fibers in a transparent PVA solution was avoided, as the addition of cured PVA would simply add another layer of reflections which could further reduce SNR. To avoid these additional back reflections, samples were prepared in methanol and placed into an external alignment field. While on the sample substrate, methanol would act to facilitate alignment due to its low viscosity and vapor pressure. Post sample fabrication, fibers were oriented with their long axis in the $\hat{p}$ direction in between the magnetic poles on the glass substrate for MOKE measurement. With the same methods outlined in the NiFe section, MOKE acquisition of these fibers began. Compared to the NiFe wires, relatively no Kerr rotation was detected.

Here, the experimental MOKE geometry was once again changed. Considering that these fibers were nominally cylindrical, and not thin film-like surfaces as seen with the NiFe wires (Figure 7.16), the light detected in the specularly reflected beam would not contain the same magnitude of signal as it would in a case where the law of reflection applies (large cylinders or flat surfaces for example). Using the Mie scattering interpretation, and some visual information regarding the surface structure of the CFO agglomerates (7.18), it seemed likely that there would be significant Kerr rotation information in the light which was scattered rather than specularly reflected.

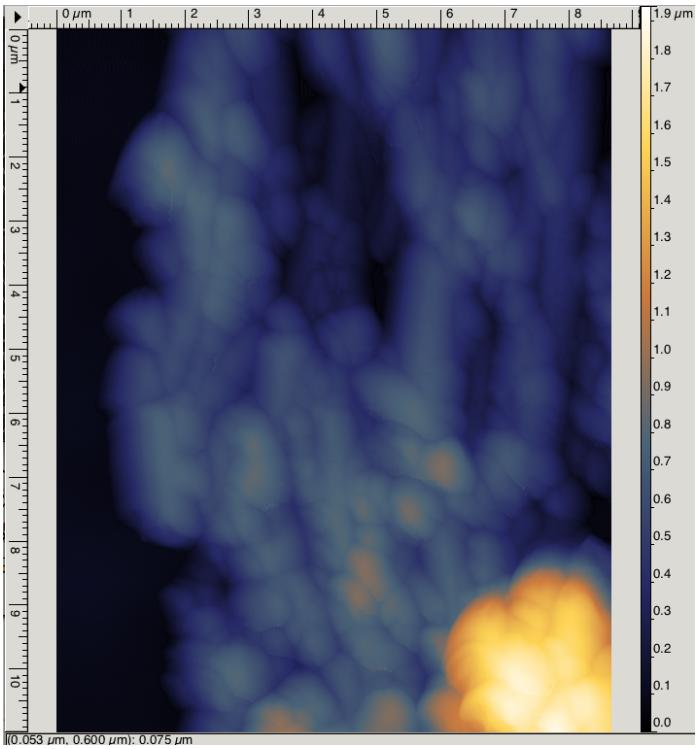

Figure 7.18 AFM data taken on methanol-aligned CFO fiber agglomerates. Note the amplitude of the topography is roughly.8 μm.

In order to have a better chance at reading a signal from large quantities of non-specularly scattered light, instead of a single lens after the reflection from the sample, a lens pair was placed with the intended function of collimating the scattered light (with the first of two lenses) and then focussing the newly collimated light (with the light colli-

mated from the first lens) on the detectors. With the addition of this significant amount of scattered light, the linearity of the beam was low enough that it could not be easily extinguished between polarizer and analyzer. We return to an original use of the QWP, and now use it to circularly polarize the signal in order to continue to use differential detection. We had originally moved away from using the QWP to induce circular polarization in order to really ensure the linearity of specularly reflected light, but now because of the highly topographically diverse surface, the use of a single QWP was unable to yield linear polarization. Figure 7.19 shows how the experimental geometry has changed since the previous step.

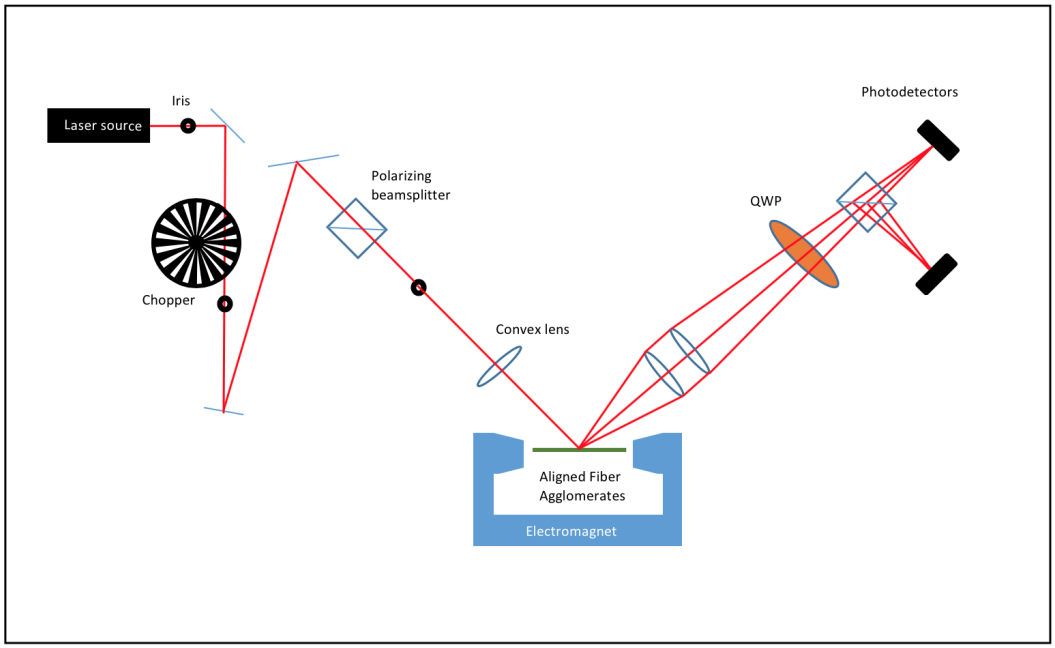

Figure 7.19 The previous experimental geometry was once again adjusted by including a pair of 50.8 mm lenses immediately after reflection. The purpose of which was to collimate and focus the scattered light. The NDF was removed and the function of the QWP was returned to enFORCing circular polarization to balance the signals between the two photodetectors.

With the adjusted geometry shown above, the sample substrate was searched for a fiber agglomerate that would have a large amount of scattered light. What I eventually found

should be called a mat of aligned fibers, rather than a chain. The fiber mat and resulting data are shown in Figure 7.20. When observing the anatomy of the reflected beam, the majority of the reflection was scattered light. The incoming specular beam was not reflected, and instead, seemed to be fully scattered.

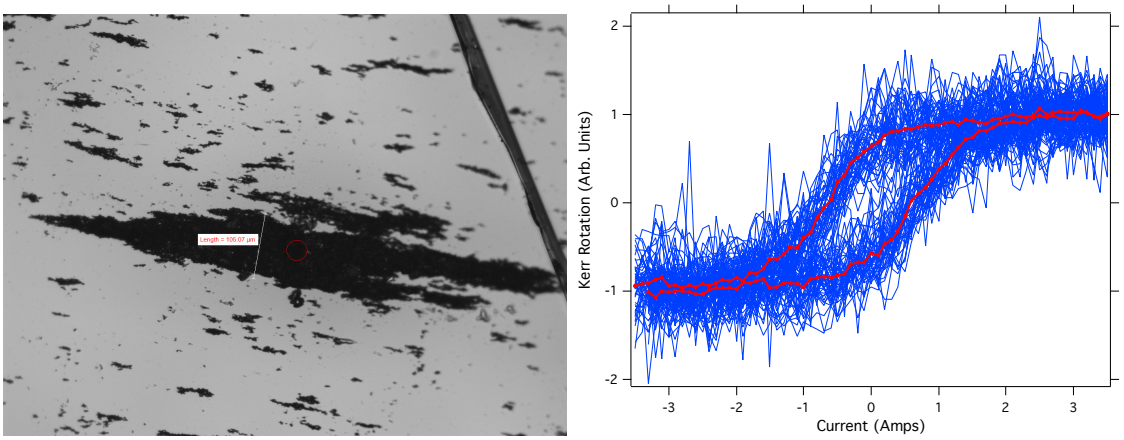

Figure 7.20 Left) Aligned nanofiber agglomerates which have formed a relatively large island when compared to the optical spot size, which is represented by the red circle. Composed of fibers which are 1μm in diameter, this agglomerate is much larger than we are interested. Right) Hysteresis data for the red circled spot of this large clump. The 92 blue datum were averaged to create the red line within. Note the pinching of the hysteresis loop at the point where current is increasing at $\sim$0.5 amps. This wasp-waisted shape is often seen in CFO samples [46]

This was the first success of using MOKE from chained and aligned fibers. As the first successful fiber measurement, we notice that there is a change in the curvature of the hysteresis loop at approximately 0.5 amps in the increasing field direction. It has been shown that cobalt ferrite will display wasp waisted curvature that is dependent on the synthesis process, specifically heating and ion transfer [47, 46]. Due to long time per run ($\sim$10-15min) and the sub 100 consecutive runs, the the intensity of light was hot enough to burn a physical hole in the sample which was too deep to image with the AFM. It is likely that in doing MOKE on this large mat of fibers, the surface melted to be a thin film-like surface or melted briefly and re-aligned in the probing field. Regardless of the reason, this particular run served 2 purposes: the verification of scattered light as a source of Kerr rotation, and

motivated the need to expedite the time in which data is acquired. Repeatability of this particular method was low due to the back reflections still dominating the reflected signal in all cases except when the aligned fibers were a large mat.

With the newly discovered information regarding the potential to use scattered light as a source of MOKE signal, the experimental geometry is once again adjusted. Rather than extracting the scattered signal from the annulus surrounding the specular reflection, I took the bold step to simply move the lens pair so that no specular light will be detected. Now, light scattered from the cylindrical nanofibers will be collimated by the first lens, and just like before, it is focused at the detectors. Unlike the previous geometries, however, there are no more substrate reflections. We have unintentionally improved Θ_{Kerr} from equation 7.12. By reducing the signal reflected from the substrate (I_{sub}), (all non-fiber reflections will reflect in the specular direction determined by the incident angle) with the modified geometry, the specular reflection is stopped.

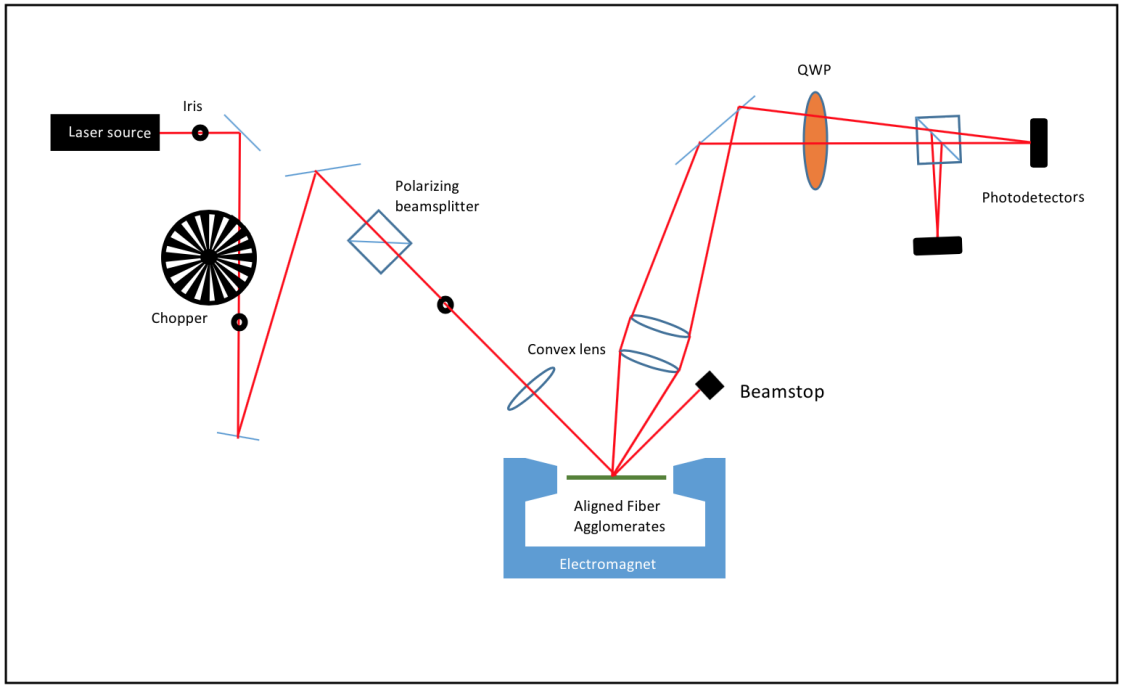

Figure 7.21 The updated MOKE geometry which detects light scattered in a direction which is not specular with respect to the incident beam.

Compared to previous MOKE measurement geometries, this geometry will become the most robust for studying topographically diverse Janus fiber agglomerates. This is a highly relative term, as previous methods showed little to no consistent success. However, with intermittent success came new sources of experimental failure. In the context of exploratory experimental MOKE, a failure to acquire a signal usually involves a low SNR rendering a Kerr rotation to be indistinguishable from the noise in which it lies. In the case of this new geometry, the low SNR would display itself as a signal fluctuation after balancing the two differential detectors. Ideally, when the detectors have been balanced to zero, the maximum range of the Lock-in-amplifier can be adjusted to view small changes of signal around 0. However, the particular issue was that the random noise amplitudes were higher than frequently observed Kerr rotations. This problem would appear consistently but with no apparent pattern, as it would be low noise for some fibers, and large noise amplitudes for others with no apparent dependance on size or length. As these noise issues continued to present themselves, it was suggested to me by colleagues to use the PVA-suspended CFO samples. The alignment quality was higher, and the average diameter of aligned fiber agglomerates was smaller. The specifics of the sample preparation can be seen in reference [11]. With the change to PVA, we were seeing an increase in narrower/longer fibers (Figure 2.5), rather than large islands (Figure 7.20). As a reminder, using these samples for MOKE was initially an issue because the PVA acted as yet another specular surface which would drown the specular signal using the previous MOKE geometry (Figure 7.19). With the new, purely scattered light geometry, the only signal added from the cured PVA solution would just be small bubbles which have been trapped in solution. Compared to a specular signal, the signal contribution from the bubbles will be negligible. As there was now no detriment to using PVA-coated fibers, we began MOKE using these particular samples. Fibers are oriented parallel to the applied magnetic field ($\hat{p}$), and a large field of scattered light is focused on the pair of detectors, being careful to balance both detectors. As a good omen, the first CFO MOKE run in the scattered geometry was a success (Figure

7.22). When attempting to balance the two signal inputs from the differential detectors, the PVA suspended fibers showed a lower noise floor when compared to non-suspended fibers. A likely reason is that fibers simply aligned and placed on glass have very little FORCe keeping them anchored to the substrate. It is likely that the uniaxial magnetic field being swept induced small oscillations or small physical rotations of isolated fibers sitting on the glass. This motion could potentially propagate to the signal detection and produce large noise amplitudes which have originated from mechanical motion of the fibers.

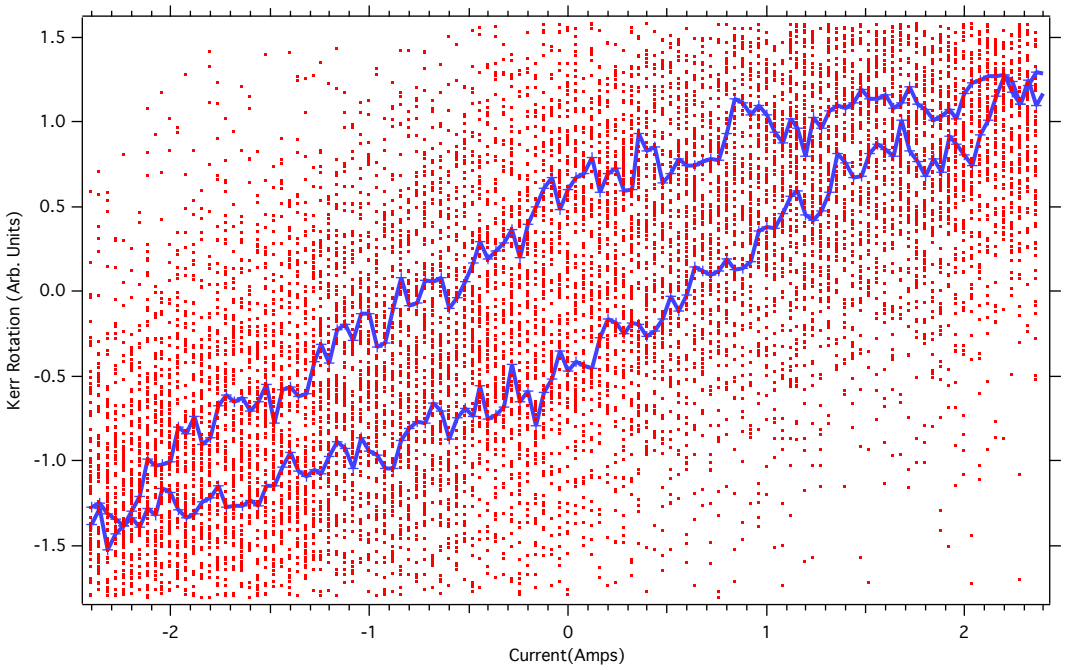

Figure 7.22 The first set of data acquired from a pure scattered MOKE geometry. Fibers are oriented parallel to the field direction. The large quantity of dots in the background are each of the individual runs which make up the blue averaged line. Unfortunately for this particular bit of time, the cooling system for the magnet was being repaired, and the range of currents was restricted between $\pm$2.5amps which is the likely preventing saturation of the magnetization of the fibers.

The repeatability of measurements of the new MOKE geometry with pva-suspended fibers was very high. Unfortunately, SNR remained average at best. In order to improve SNR at this point, I considered the question, "how do I acquire more scattered light?". Noting the cylindrical geometry of the individual fibers and the mechanism in which Mie

scattering from cylindrical surfaces occurs, it was clear that a larger signal could be acquired by rotating the aligned agglomerates perpendicular to the direction of the external magnetic field ($\hat{s}$). Figure 7.23 shows how the scattered rays will differ in direction based on the angle of the cylinder.

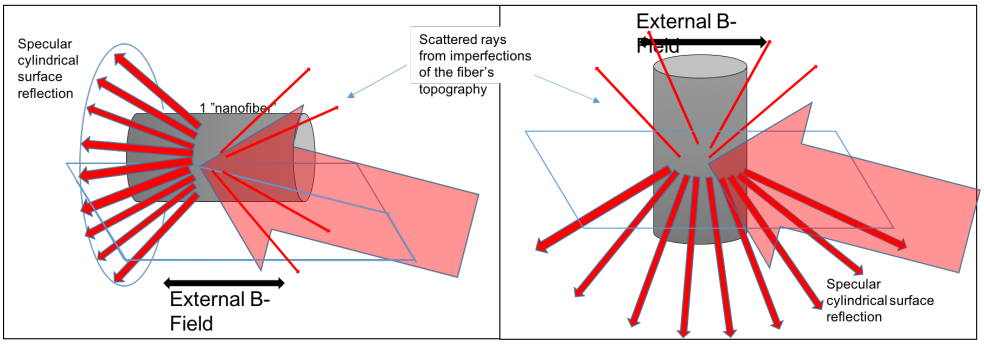

Figure 7.23 Following the geometry outlined in section 3.6.2, light scattering from a non-normal incidence will produce a cone of reflected rays. The intensity of light varies at different azimuthal angles with respect to the long axis of the cylinder, but will travel according to the law of reflection. Combining this conical scattering view with the geometry shown in Figure 7.21, it is clear that at a 45° incident angle, much of the scattered light will indeed be in-line with the specularly reflected substrate reflections. A rotation of the entire substrate so that the fibers are now oriented in the $\hat{s}$ direction allows the scattered light to exist in the $\hat{p}$ plane, theoretically making a larger signal available for detection.

This simple rotation of the sample improved the SNR of the overall signal (Figure 7.24). In addition to the obvious benefit of lower noise, the integration time per data point was also deceased. With the improvement of SNR per individual run, less preprocessesing noise reduction was needed, so that overall speed could be improved, which would help to avoid noise contributions from heating (Figure 7.29). With the reduction in acquisition time, and the improved SNR for single runs, we benefit even more from the automatic continous acquisition of MOKE data via LabVIEW. Now, since the averaging is so critical, we begin to consider number of averages needed as a single entity since they will end up representing a single datum. Figure 7.24 shows the 100 automated average runs that contributed to the MOKE for $\hat{s}$ oriented fiber agglomerates. Although we now acquire data

with a higher SNR, we are also fundamentally measuring a different magnetization mechanism. Considering shape anisotropy, which was discussed earlier, the easy axis should nominially be along the long axis of a cylindrical wire, while the hard axis is in any radial direction normal to the surface of the cylinder. We are no longer measuring the presumed easy axis magnetization, but we are now able to acquire much higher SNR. Keep in mind, that the fibers measured are not perfectly aligned at the scale of an individual fiber, and it is likely that every constituent fiber has a nonzero angle with the intended alignment direction, yielding a mixture of easy and hard axis magnetization. Due to this fact, it is unlikely that the easy and hard axis magnetization can be fully separated.

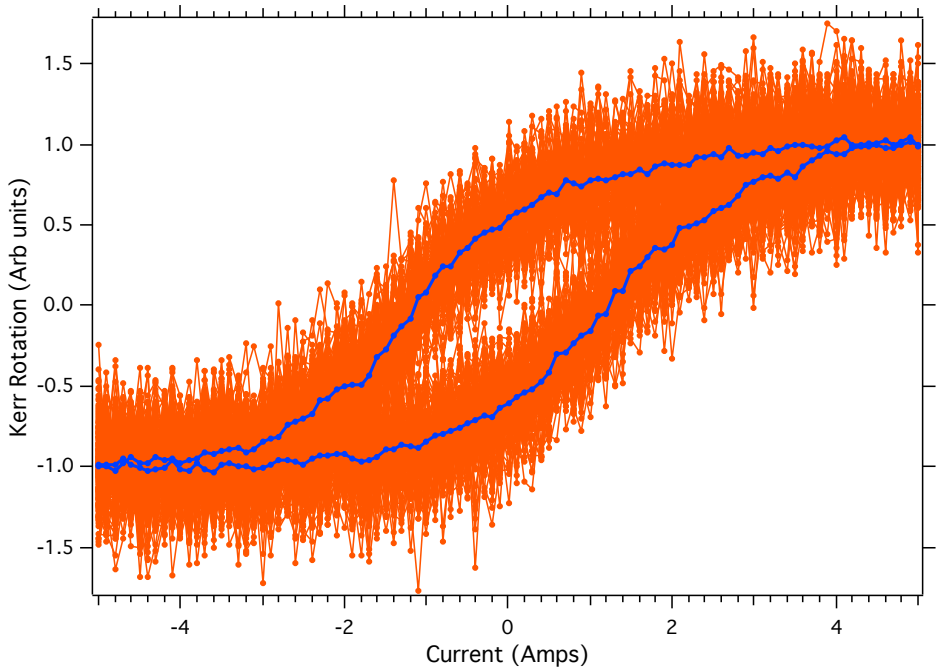

Figure 7.24 The first set of data acquired using continuous MOKE data acquisition of fibers oriented perpendicular to the applied magnetic field. The blue line is constructed from the average of 100 orange runs. Notice that the SNR is low enough that we are able to see the middle of the graph even through all of the relatively noisy individual runs.

7.4.2 MOKE MEASUREMENTS ON JANUS FIBERS

With a much more robust method for acquiring magnetization data on aligned nanofiber agglomerates, the MOKE experimental design is finally prepared for Janus fibers. Until now the use of Janus fibers was avoided because, by definition, they are nominally half as magnetic as the pure CFO fibers. This was confirmed by our collaborators who synthesized the fibers [43] as well as within our own lab via vibrating sample magnetometry, where we observed that the saturation magnetization of the pure cobalt ferrite is $\sim$2x that of an equal volume of Janus fibers. Fortunately the experiment did not care about my reluctance to measure these fibers because initial results seemed promising. Figure 7.25 shows the first successful magnetization curve on an aligned Janus fiber agglomerate.

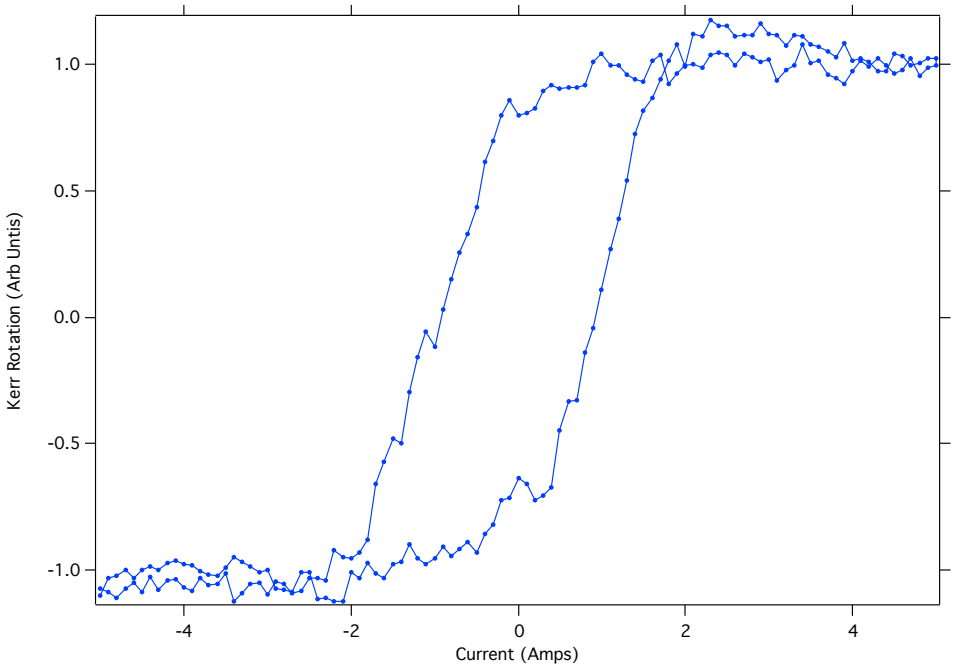

Figure 7.25 Janus fibers agglomerate oriented in the $\hat{s}$ direction. This particular graph was created by averaging 300 consecutive runs.

From this initial curve, many more Janus fibers were measured. As the experiment was fine-tuned via more careful placement of optics, blocking of back reflections, and measuring more well-aligned fiber agglomerates, one of the most interesting aspects of

measuring Janus fiber agglomerates was the appearance of non-symmetric features in the magnetization curves. These features show up as sharp dips or changes in curvature at various field values. Without averaging, these could potentially be written off as noise. The repetition of these behaviors in agglomerates of different shapes and sizes was an indication that they were not, in fact, noise, but were a real magnetization processes. These effects can be seen in the various images in Figure 7.27. The detection of these irregularities and the result of averaging can be seen in 7.26.

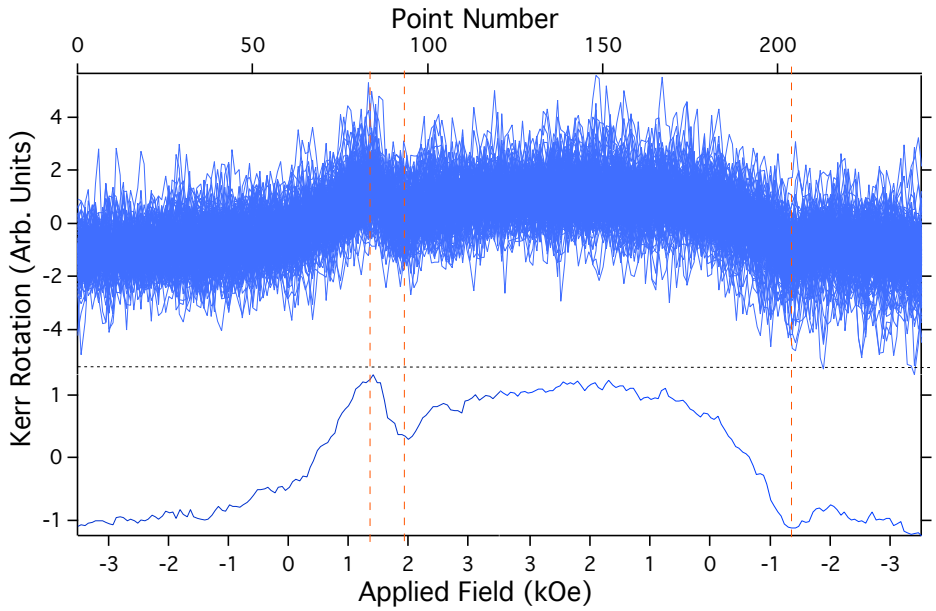

Figure 7.26 The top image shows a total of 100 MOKE runs, while the bottom graph shows the result of averaging those individual runs. The vertical dashed lines draw the eye to the assymmetric features that are commonly seen when measuring the magnetization curves on aligned Janus agglomerates.

We can see in the top image of Figure 7.26 that without significant averaging the noise amplitude of a single run would have a magnitude as large as the large dip in magnetization seen between 1 and 2 kOe. Magnetization curves from Janus fiber agglomerates show a variety of features, including the least interesting feature of a smooth, featureless magnetization curve.

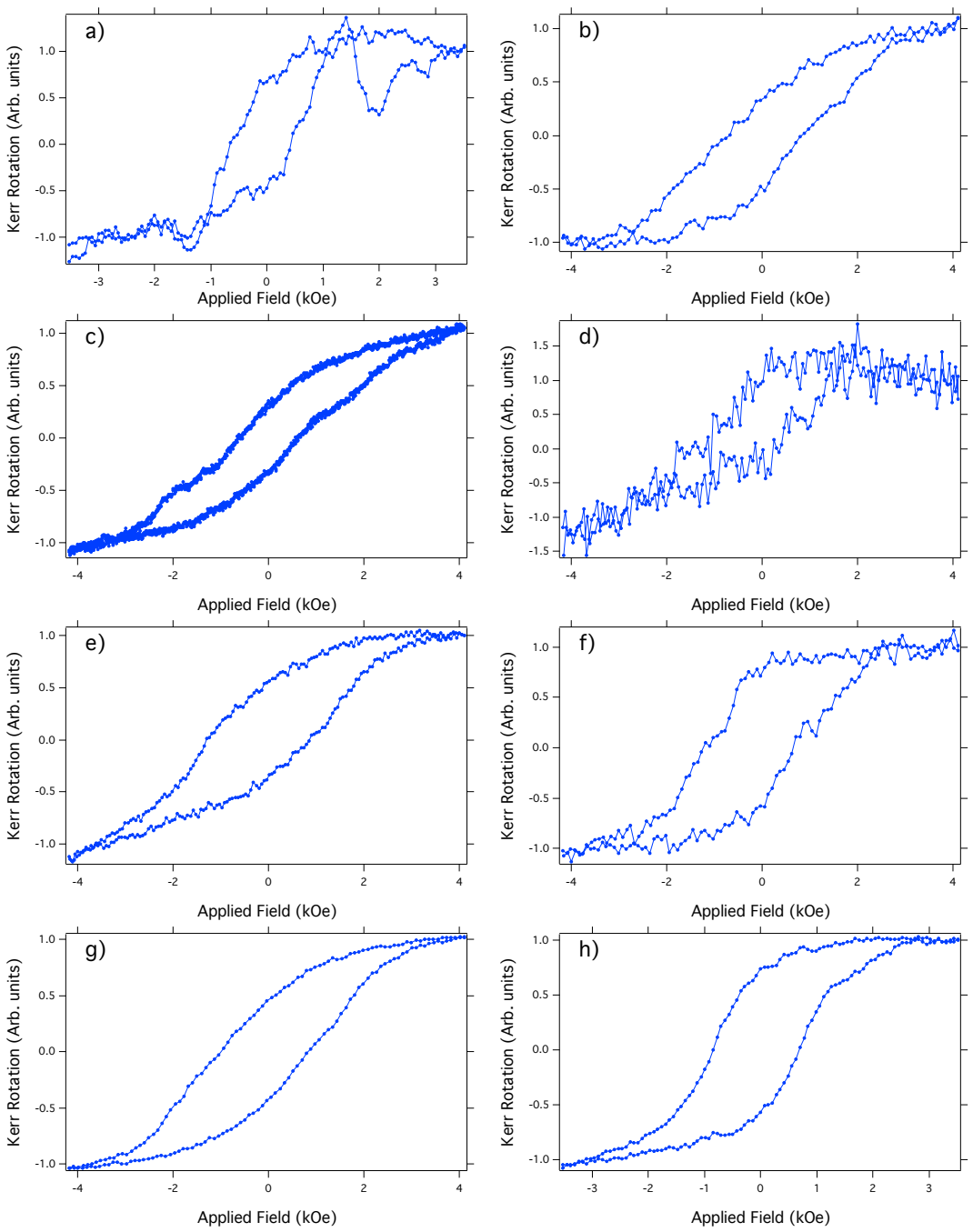

Figure 7.27 First half of Janus magnetization curves are results of averaging between 100-400 consecutive runs. Each curve is taken from a different aligned Janus fiber agglomerate. A variety of asymmetric features can be seen throughout this collection of magnetization curves.

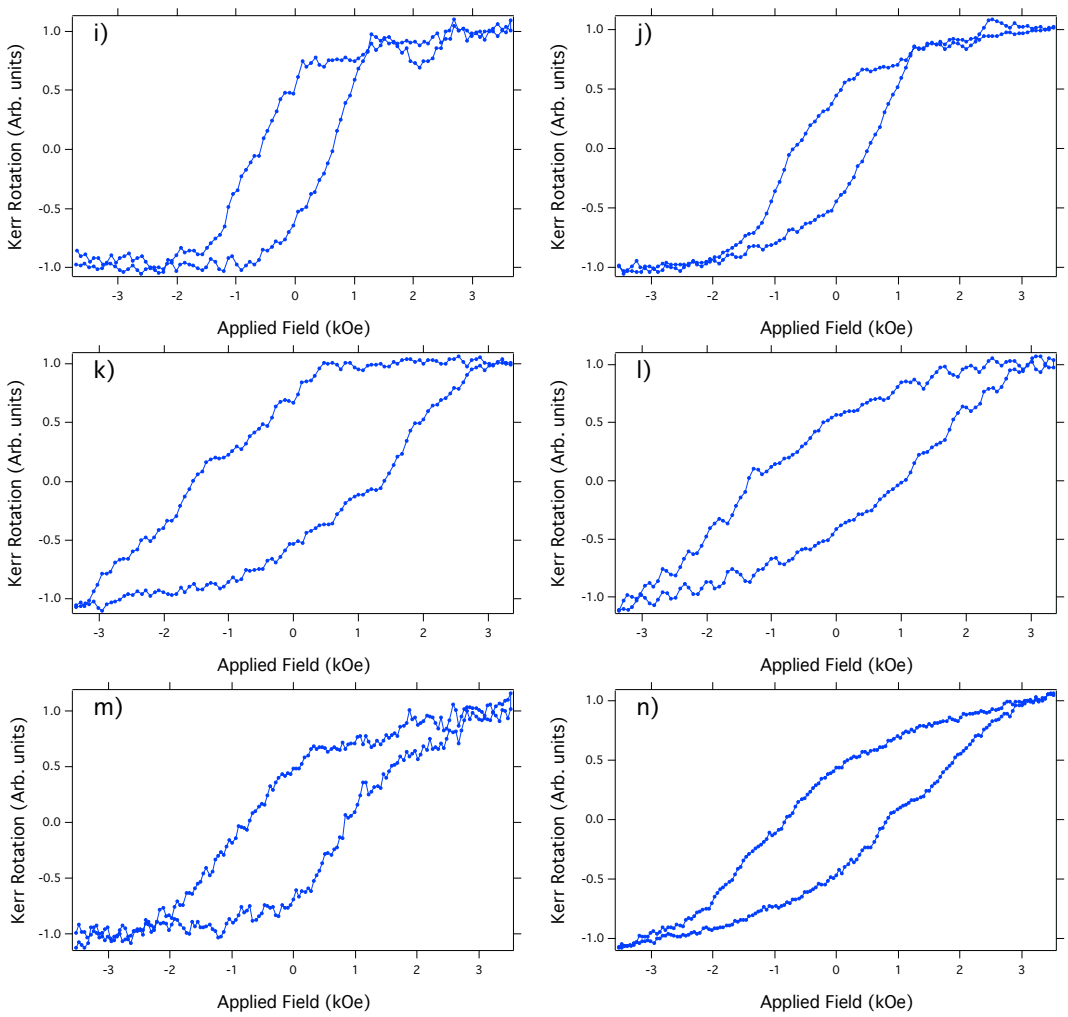

Figure 7.28 Second half of Janus magnetization curves are results of averaging between 100-400 consecutive runs. Each curve is taken from a different aligned Janus fiber agglomerate. A variety of asymmetric features can be seen throughout this collection of magnetization curves.

The data shown above were collected over many months. Before switching to the oscilloscope for data collection (section 7.3), each single run could take up to a full minute, and if that is repeated 100-400x, a full set of MOKE data can take up to 7 hours to build. In order to create a normalized set of MOKE data, the Kerr rotation of each individual MOKE run must be recorded (section A.2.1 in Appendix). This is carried out with a homemade Igor Pro function, and in certain data sets we see that the Kerr rotation will be decrease over the duration of a multiple hour MOKE run. We see this effect in Figure 7.29.

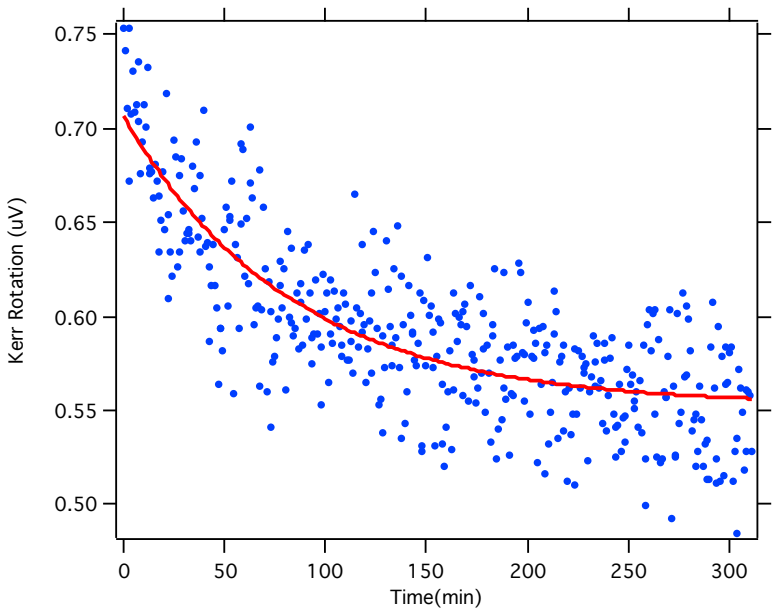

Figure 7.29 The Kerr rotations taken from the Igor Pro normalizing program plotted against time. There are 400 data points here collected over approximately 5 hours. The fit type is an exponential decay.

Even with the initial MOKE measurements on aligned fiber agglomerates, we saw that excessive power would result in the destruction of the aligned fibers. Unlike the highly reflective permalloy where the incident power was not really considered as a limiting parameter, there was a definite hard limit on the the power used for measuring aligned fiber agglomerates. This was easily identifiable as there would be an apparent hole where there was once an agglomerate. Because power (W) and intensity (W/m^2) have different dimensions, an equivalent power would impart more energy to a larger fiber than a smaller one. Rather than risk burning the fibers, we determine how/if laser power is related incident power. It is found in literature [42, 45] that SNR scales positively with increasing power. This was verified in Figure 7.30.

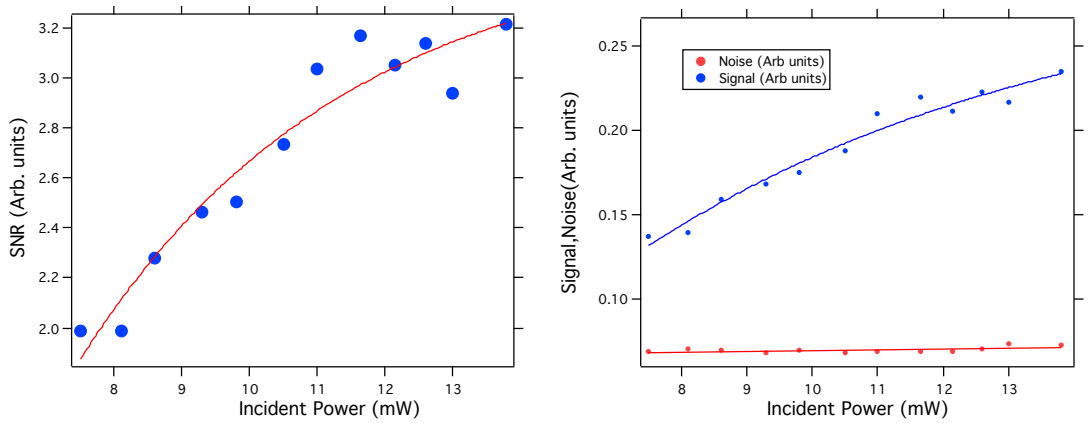

Figure 7.30 The improvement of SNR with the increase of incident power. SNR was taken from a set of 10 average MOKE runs at each different incident power.

The "signal" in "signal" in the above image is taken as the Kerr rotation, while the "noise" is simply the standard deviation in areas where the signal saturates, where it is nominally unchanging. We see an exponential-like dependence on power, suggesting that there is a limit to how much one can improve SNR by increasing the power. Fortunately for us, we don't have to wonder about when further power increase will no longer improve SNR, since we have the hard limit of sample destruction. Typically, measurements are taken with a power of ∼9-12mW.

7.5 Analysis of Scattered MOKE

As shown in Figures 7.27 and 7.28 , we observe field-dependent variations in the individual hysteresis loops from different agglomerates. These features in the averaged magnetization loops presumably offer insight into the micromagnetic properties of these structures and ultimately the dynamics of the chaining process that creates them, which is an obvious potential source of variability. In an external field, our fibers are seen to assemble both end-to-end and laterally, depending on the distribution of fiber lengths and the concentration of fibers in the PVA solution [11]. Finally, the individual fiber properties depend on the Janus electrospinning process, which is known to produce variations in wt.% of the

composite materials between individual Janus fibers [43]. To compare the magnetic behavior of the CFO within the Janus fibers to other common CFO geometries, Table 1 shows average and standard deviations of the coercive field (H_c) and Remanent Field (R) from our measurements, compared with magnetization data found in the literature. The values used in Table 1, are acquired from 14 sets of averaged hysteresis curves, each taken from ScMOKE measurements on a different fiber agglomerate. Figure 7.27 shows the hysteresis loops used to calculate the ScMOKE values in Table 1, while Figure 7.31b shows the VSM data representing the entire suspension of aligned agglomerates. Magnetic properties measured with the VSM differ from MOKE for thick magnetic samples due to the fact that VSM probes the entire magnetic volume of all agglomerates lying on the sample substrate, while MOKE only measures magnetization from within skin depth on the order of 10 nm. The images in Figures 7.31 and 7.27 serve to illuminate the differences between a single surface measurement of an aligned agglomerate using ScMOKE when compared to a bulk volume measurement of VSM.

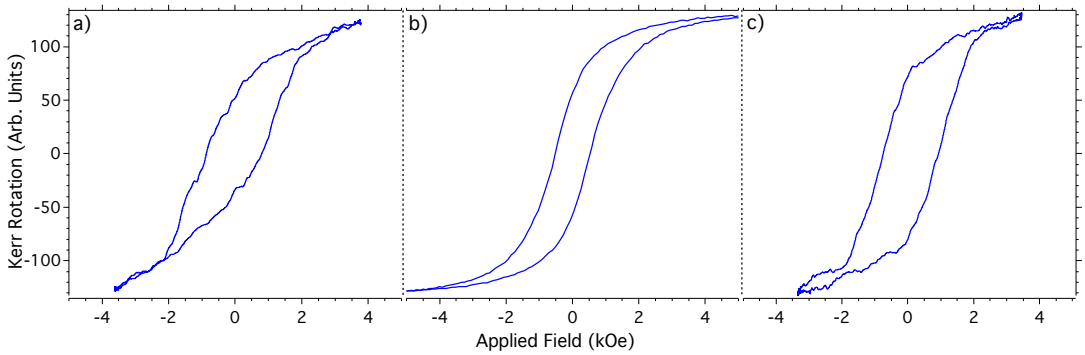

Figure 7.31 a) A single averaged magnetization curve taken via ScMOKE with fibers oriented parallel to the substrate surface, and pointed along the $\hat{s}$ direction with respect to the incident plane. b) VSM data taken for an entire sample of fiber agglomerates aligned perpendicularly to the magnetic field. c) Same as a) except the agglomerate is pointed along the $\hat{p}$ direction with respect to the incident plane. Table 7.1 shows that the coercive field for single fiber agglomerates is approximately 70% larger than the average of an entire agglomerate-covered substrate taken via VSM. Panel c) shows a slightly more rectangular magnetization curve, suggesting a possible easy axis along the fibers' long axis.

	ScMOKE fibers	VSM fibers	Solgel NPs [48]	Wet Chem NPs [49]
H_c (kOe)	0.85± 0.2	0.495	0.65	0.2-0.6
R (%)	0.5± 0.12	0.46	0.4	0.4

Table 7.1 Comparison of averaged hysteresis data obtained from ScMOKE and VSM measurements of CFO-containing Janus fibers with representative literature results for Solgel precursor synthesized CFO nanoparticles and wet-chemical mixing prepared nanoparticles. Referenced values are approximated by Figures found in corresponding literature. Coercivities for ScMOKE and VSM measurements are determined by fitting each curve with a hyperbolic arctangent and extracting the field value where the fit line passes through zero magnetization (y-axis).

Hysteresis data from CFO thin films along the easy and hard crystallographic directions show sharp transitions along the easy axis with little to no magnetic coercivity along the hard axis [50]. This is unsurprising as the magnetization transition along an easy axis is facilitated by rapidly switching domain wall motion, while magnetization along a hard axis relies on domain rotation. Under the influence of a slowly changing magnetic field, domain rotation yields a slow transition from one saturation magnetization to the other, while domain wall motion is typically a sharp transition, associated with the pinning and unpinning of many domains simultaneously. Both ScMOKE and VSM from perpendicularly oriented agglomerates, Figures 7.31a and 7.31b respectively, show a smoother transition that suggests a mixed easy/hard axis state. We also see similar behavior from an agglomerate measured along its shape anisotropy-determined easy axis in Figure 7.31c. The VSM-determined R $\sim$.46 and H_c of 0.495 kOe from aligned fiber aggregates are shown in Figure 7.31b. Table 7.1 compares the fiber agglomerate results with solgel precursor-created CFO nanoparticles[48] and nanoparticles formed with wet chemical mixing[49]. The data show large variations in magnetic properties depending on the CFO synthesis procedure, including temperature(s), geometry, and size.

Figure 7.27 shows several distinct and unique loop shapes acquired via ScMOKE on different fiber agglomerates. There are several different effects that could cause these variations, including domain wall reorientation and propagation, the mixing of Polar and Lon-

gitudinal MOKE geometries, relative proximity and periodicity of aligned fibers within an agglomerate, and the purity of CFO phase.

One of these effects is domain wall reorientation caused by different local energy minima. It is known that a smooth hysteresis loop will be observed if magnetic moments undergo coherent rotation following the direction of the external field [21]. Discontinuous changes in the moment at a specific field arise from motion and/or pinning/release of domain walls. Since we apply the external magnetic field parallel to a hard axis in terms of the shape anisotropy of individual fibers, we might expect purely coherent rotation. However, it is still possible for local domain walls to nucleate and jump across the diameter fiber, for example in fibers that are aligned such that the field is at an angle to both easy and hard axes [21]. Domain wall pinning can also give rise to sharp magnetization discontinuities, which have been observed with MOKE, but typically in mono-few layer thick samples [51]. Complex domain walls have also been observed in "thin" cylindrical magnetic wires, including transverse, vortex, and Bloch walls. Depending on the type of domain wall formed, the magnetization vector on the sample surface rotates through different orthogonal directions as the domain wall propagates across the surface [52], a phenomenon that has been seen in both experiments on and simulations of sub-100 nm diameter cylindrical magnetic wires [53]. The diameter of the Janus fibers are an order of magnitude larger, and the agglomerates are not isolated wires. Without knowing the micromagnetics of the Janus agglomerate, complex domain wall motion cannot be ruled out. SMOKE (Surface MOKE) measurements taken on crystallographically frustrated surfaces have been shown to exhibit similar loop behavior at certain vicinal angles with respect to the crystal axis. Notably we see very similar magnetization curves to Figure 7.27d) for Fe thin films grown on Co substrates with a 7° vicinal angle [54]. We could be seeing similar effects at the BTO-CFO interface if there is a significant lattice mismatch, leading to crystallographic frustration.

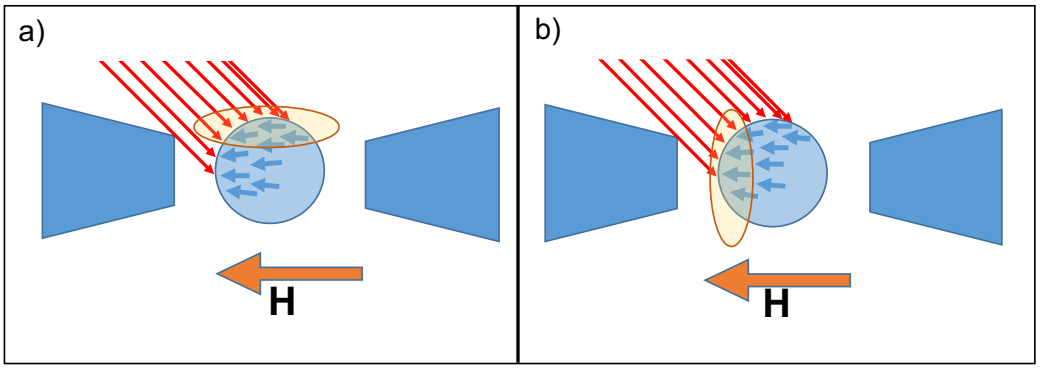

Figure 7.32 a) shows a representation of our experimental geometry from the top view (along the $\hat{s}$ direction). the arrow labelled "H" shows the direction of the applied magnetic field generated by the poles. The highlighted portion of a) shows the portion of the circular cross section that would qualify as LMOKE, while b) shows the portion of the cross section that qualifies as PMOKE. At the non-specular Mie scattering limit, the light scattered from a cylinder will contain signal from all illuminated parts of the cylinder agglomerate.

Another possibility to consider is the mixing of Polar (PMOKE) and Longitudinal (LMOKE) MOKE signals as a potential source of hysteresis loops with minor-loop behavior. When defining LMOKE and PMOKE, the easiest visualization is by using a thin film geometry. MOKE variations are differentiated by the direction of the applied magnetic field with respect to the plane of incidence. Light incident on the surface of a 2D or quasi-2D sample will produce a coherent reflection from a planar surface, and qualify as a single orthogonal MOKE geometry. The mixing of P and L MOKE can be linked to the $\hat{s}$-orientation of the fibers. Light incident at $\sim 45^{\circ}$ will strike the front surface of our fiber agglomerate, as expected (Figure 7.32a) but will also strike the surface of the fiber which faces a magnetic pole (Figure 7.32b). During any given experiment there is an illuminated surface that lies parallel to the magnetic field (LMOKE), and one which is perpendicular to the magnetic field (PMOKE). The difference in the magnitude of the demagnetizing field at the field-parallel surface (Figure 7.32a) and the pole-facing surface (Figure 7.32b) could yield a magnetization curve which will be a superposition of two different magnetization processes.

Previous research with diffraction MOKE (DMOKE) show a wide variety of hysteresis loop shapes that are similar to what we observe in ScMOKE. DMOKE is sensitive to motion of an individual magnetization vector in a patterned element that is the same in all elements of an array where each element has the same spatial separation, similar to diffracted light from a grating [55]. Several features of our magnetization curves appear to match those seen across the different orders of magneto-optical diffraction. However, we have a rough agglomeration of nano-sized magnetic fibers and not a well-ordered array of elements [56][57]. The most obvious similarity between ScMOKE and DMOKE is that both detect off-specular signals, i.e., light scattered from a material. In the ScMOKE case many small fiber "building blocks" are assembled into a larger agglomerate, all of which are scattering light. It is possible that they exhibit common magnetic reversal features that occur at similar fields, and thus show similar behavior in scattered light to observations of periodic elements with DMOKE. A DMOKE experiment yields multiple orders of detectable signal, which of course are not observed in ScMOKE, but an agglomerate of multiple fibers may cause an interference of signals from multiple fibers undergoing similar magnetic behavior at the same external field. We do not, however, observe distinct intensity maxima as would be expected in a DMOKE orientation.

Finally, cobalt ferrite has long been known to show unique magnetization curvature [46], most commonly a "pinching" of the hysteresis loop about the zero field point. We observe a similar effect in various graphs in Figures 7.27 and 7.28. Until recently, the specific reason why, and to what degree, the curvature of CFO magnetization would change was unknown [47]. It has been discovered that the curvature of CFO samples is highly dependent on the sample preparation method. The investigation by Zhang et al. studied the origin of CFO pinching by creating various pellets of CFO and controlling the heating time and temperature, as well as the precipitation time of the CFO. After observing various changes in curvature under different synthesis conditions, the group attributed these changes to the mixing of phases between Co and Fe ions. Although the electrospinning

process is inherently different, it is possible that there is a similar phase mixing between the two ferroic phases of BTO and CFO.

7.6 FORC

In theory, measuring raw FORC signal is just a natural extension of a magnetization curve measurment. Rather than taking symmetric magnetization curves, FORC data is built from assymetric magnetization curves that sweep between a fixed maximum field, H_{max}, and a changing minimum "reversal field", H_{r_i}. After completing a single magnetization curve which ends at an arbitrary reversal value H_{r_i}, the next reversal value will be a value $H_{r_{i+1}}$, following the relationship

$$H_{r_i} - H_{r_{i+1}} = \Delta H_r, \tag{7.13}$$

where ΔH_r is a constant. Thus each reversal value is equally spaced from the previous value. Because we have developed MOKE into a powerful tool for measuring magnetization of aligned fiber agglomerates, we seek to adapt it for FORC data acquisition. Modifying and analyzing FORC data from our last orientation of ScMOKE requires little physical geometrical changes, but requires a large amount of modifications in automation and analysis. A typical FORC data set requires 100-150 reversals, and MOKE wants at least 100 averages per reversal to reduce noise to a level at which non-noise magnetization features can be observed. In order to facilitate ScMOKE FORC, we first need to create a proper applied field with reversals. A consideration that must be made is the way in which data is acquired and processed. All is acquired via continuous triggering of 10 sec windows within the oscilloscope. This eliminates the ability to simply feed our MOKE program a set of field inputs which look like Figure 7.12 due to the fact that $\sim$100 reversals would take at least 1000 sec (10s·100revs), which is much larger than the available 10sec window. Instead, each reversal is treated as an independent MOKE run which is then run the appropriate number of times to build a sufficient average (typically 100-200) before moving on

to the next reversal field. The next issue revolves around magnetic viscosity in general. We have sought to eliminate the dependence of field value on the magnetic rise time by switching it to an *in-situ* field measurement. We must now confirm that there is no dependence of the Kerr rotation as a function of rate. This will have direct consequence on the LabVIEW field control. If there is no dependence, the parameters needed are simply H_r, H_{max} and n_{max} (number of steps). Where we use the combination of these three to calculate the step size of the applied field ΔH_a,

$$\frac{H_{max} - H_r}{n_{max}} = \Delta H_a. \tag{7.14}$$

On the other hand, if it is determined that the Kerr rotation is rate dependent, we must modify the steps taken to occur at a fixed rate ($\Delta H_{a_{max}}$). We fix the rate by the very first, most outer, reversal curve. For subsequent reversals, we use $\Delta H_{a_{max}}$ to calculate a new number of steps, n', needed to reach each new H_{r_i} from H_{max}, according to

$$\Delta H_{a_{max}} = \frac{H_{max} - H_{r_{min}}}{n_{max}} \tag{7.15}$$

$$n' = \frac{(H_{max} - H_{r_i})}{\Delta H_{a_{max}}}. \tag{7.16}$$

While mathematically this is not difficult to grasp, the implementation will require that the other parts of the 10 second window to be filled with idling rather than field ramping. To test whether or not we need to implement the more complicated approach to the applied field steps, MOKE data is acquired for the variety of steps (Figure 7.33).

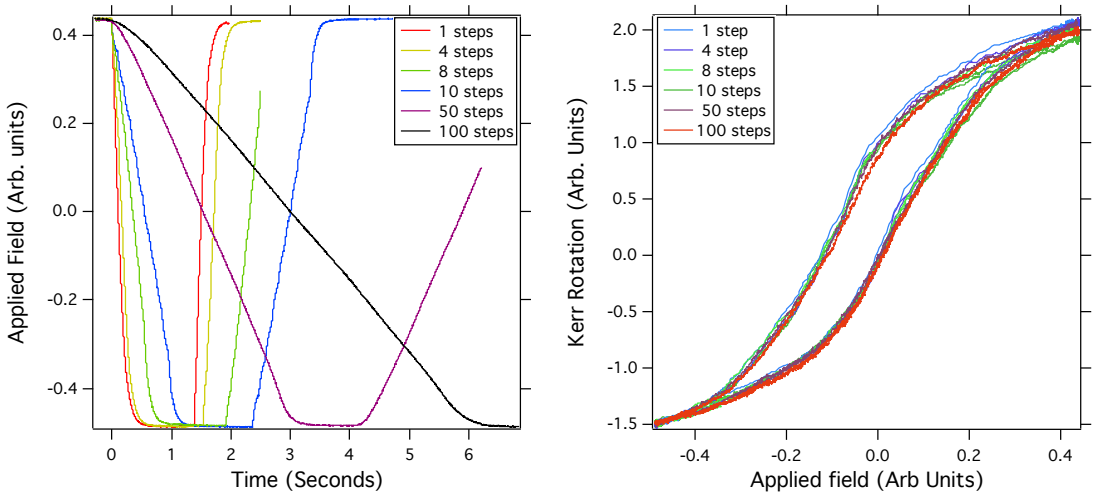

Figure 7.33 The image on the left shows the different rates of applied field for the MOKE results on the right.The right image shows the MOKE taken from the same aligned fiber agglomerate with each of the rates from the left represented.

From the right image above, we see some bulging of the MOKE data around 0 applied field in both directions. We also see, in the left image, that each of the steps taken has the predicted exponential behavior, but the visibility of this effect is minimized with 50 or 100 steps. While the bulging effect might be minor at different rates, this is the type of data that might appear in a FORC density plot as a result of our eventual FORC analysis. We employ the previously mentioned "fixed rate" approach to create a set of reversal curves which take the form in Figure 7.34

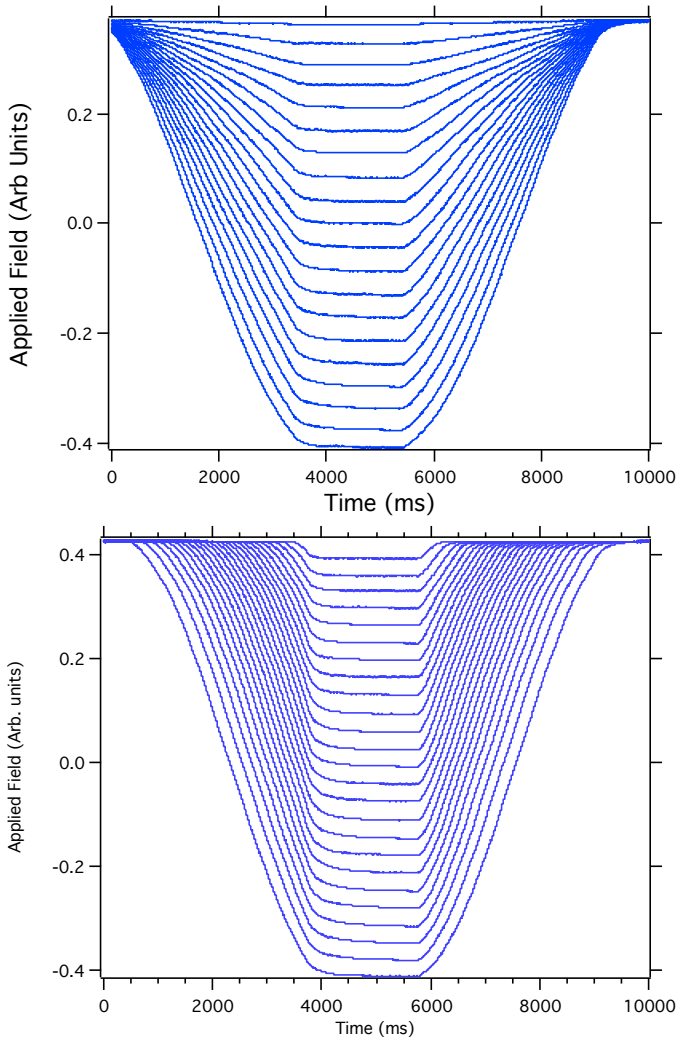

Figure 7.34 Each of the two images uses two different approaches to building field steps for FORC. The top image neglects rate-based effects, while the image on the bottom attempts to minimize such effects by ensuring that each reversal nominally has the same rate of change of field with respect to time. Looking closely, unfortunately this is not fully consistent at the smallest (least negative) reversal, when compared to the largest reversal at the bottom of the bottom image. Comparatively, though, we see notable improvement from the rates in the top image.

Implementing the methods from equation 7.16, we are able to build a set of reversal curves with constant ramp rate as shown in the right image of Figure 7.34. Similar to MOKE, now that we have a "correct" set of field values, we can continue from preparation steps to raw data acquisition. FORC density plots typically require between 75-150 rever-

sals [58], and although steps have been taken to improve the speed of data acquisition, it must still be a consideration. We have restricted a single MOKE curve to fit within a 10 sec window in the oscilloscope, but we base that on a successful trigger of the oscilloscope. If the trigger is not reset at the point the next FORC curve starts, the entire next 10 sec run will be wasted. To avoid this particular situation, an additional 2 sec wait time is added after each MOKE run, to allow the trigger to reset. While this does not seem like much of an addition, we must remember that the duration of a full FORC run can be calculated symbolically:

$$T_{FORC} = ((t_{osci} + t_{trig}) \cdot N_{runs}) \cdot N_{rev}, \tag{7.17}$$

where t_{osci} is the size of the window in the oscilloscope (typically 10 sec), t_{trig} is the additional time allowed for the trigger to reset (typically 2 sec), N_{runs} is the number of consecutive runs per reversal needed to build an average with "low enough" noise, and the N_{rev} is the number of reversals (up to 150) needed to create a high enough resolution FORC curve. If we neglect the N_{rev} term, the total time is just the time for a single MOKE run, which with typical parameters is about 40 min. The additional term, with the possibility of being as large as 150, would increase this 40 minute run to 4 days. Over this relatively long period of time, it is important to keep the laser power low enough as to not induce a slow heating and decomposition of magnetic effects, but high enough to get the optimal single-run SNR for future analysis. Initial FORC data is shown in Figure 7.35

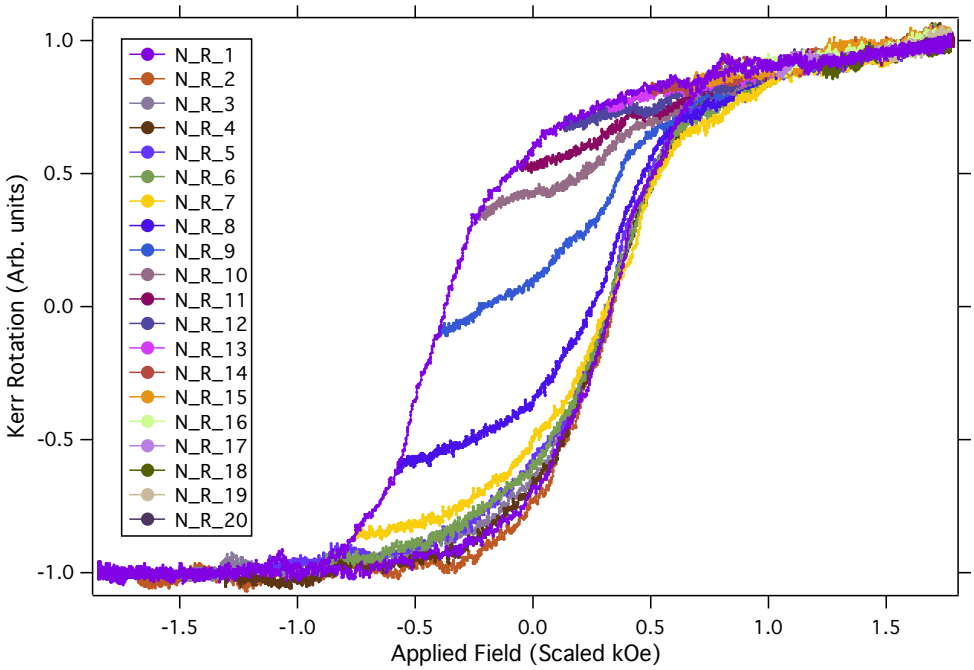

Figure 7.35 Initial FORC data with 20 reversals, which was used to test the different automation functions used in data preparation: acquisition, triggering, and normalizing. Field data is scaled by a constant factor. At the time of acquisition, the longitudinal Gauss probe was broken, and an axial probe at an arbitrary angle was implemented for the purposes of continuation of research. Only the most outer full hysteresis loop is plotted, as each of these minor reversals follow the same decreasing field path from the max field value.

The graph above shows an initial set of FORC data using the oscilloscope to monitor both the field and the Kerr rotation. Compared with a single MOKE run, the relatively large amount of data is processed with new automation programs in Igor Pro (section A.2 in Appendix). The data taken above did not implement the constant rate modifications to the field steps, as the longitudinal Gauss probe was out of commission, and accuracy measurements like that were not being considered. However, this served as a proof of concept that FORC data acquisition could be automated and subsequently processed with a series of useful functions in Igor Pro. Shortly after the data in Figure 7.35 ($\sim$1 month), was measured, a new longitudinal Gauss probe arrived that exhibited a few quirks. The first, 25 Oe minimum step size, has been discussed above in section 7.3, while the other

two were discovered while implementing the new automated FORC with this new Gauss probe. The first "new" error was a triggering delay, which happened infrequently, and without an apparent pattern. The next was seemingly an error with the probe, in which the output would discretely change the rate of field ramping, which would cause the field to not reach saturation in the given oscilloscope window. Both of these can be seen Figure 7.36.

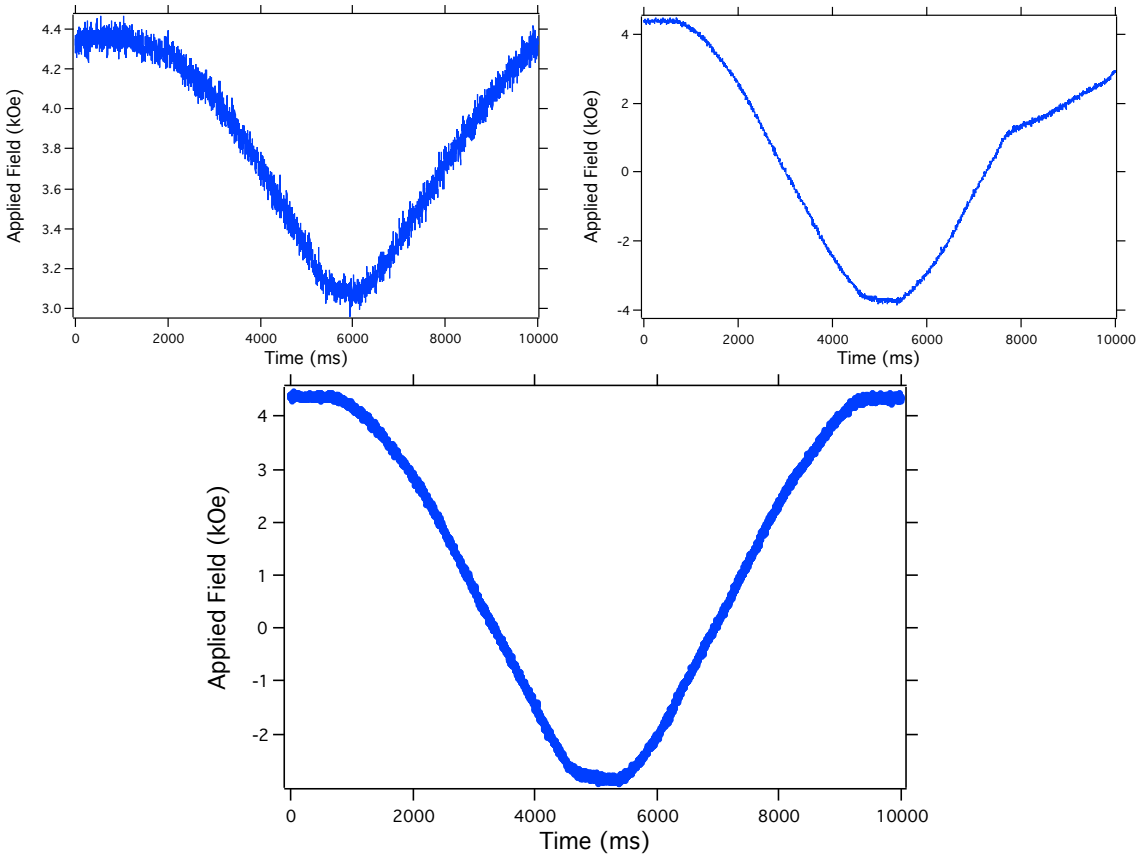

Figure 7.36 The bottom image shows a correctly triggered field sweep. The first image on the left shows a delayed/missed trigger. The Figure on the right shows erratic strange behavior with the probe. Both of the erroneous field sweeps will result in unreliable MOKE data.

Due to these unreliable field data, the corresponding MOKE data also could not be trusted, and would be removed from the total set of FORC curves. While this could be done by hand, there were often ∼10,000 files to look through, and manually detecting

small delays in triggering could leave room for error. A functional procedure was developed (section A.3.2 in Appendix), which would find the average minimum and maximum value for each field data set an remove the field and corresponding MOKE data if it appeared to be beyond a certain threshold when compared to previous consecutive runs. Additionally, identifying the minimum average value of the field would assist in separating MOKE data based on the reversal field value, H_r. The function would then proceed to remove "rotten" field and MOKE data from the larger pool of data before grouping and averaging of individual reversals was completed. Figure 7.37 shows an example of how the vetting procedure identifies erroneous field data, and Figure 7.38 shows the resulting normalized and averaged reversal curves.

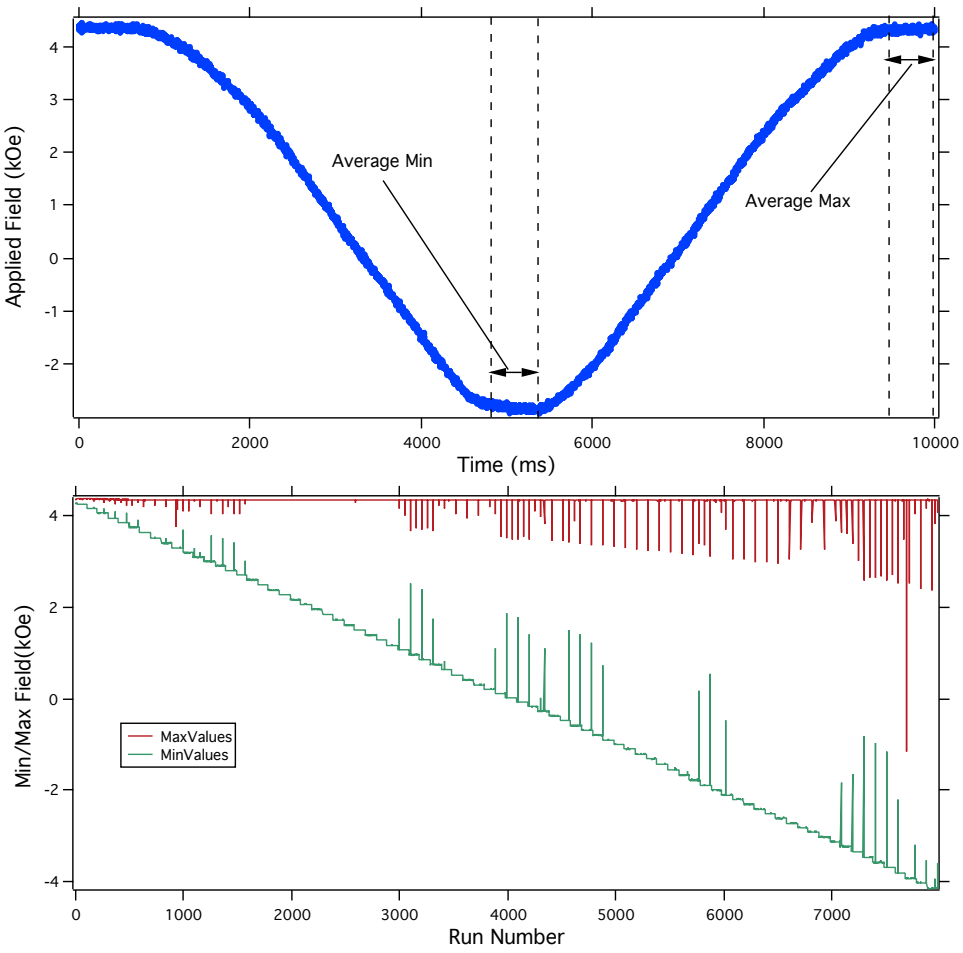

Figure 7.37 The top image shows a good field reversal and the ranges where the min and max field values are identified. For all reversals, the max field value will be the same. The min field value will decrease in a stepwise fashion after the set number of averages have been completed. The min values should correspond to the pre-determined H_r values. The sharp vertical lines in the bottom image represent the individual runs which have exhibited one of the errors from Figure 7.36. If a trigger is missed or delayed, both the min and max values will be incorrect, while if the "strange rate change" happens, we will see a spike in the max field value, as it will not reach the saturation value in the trigger duration.

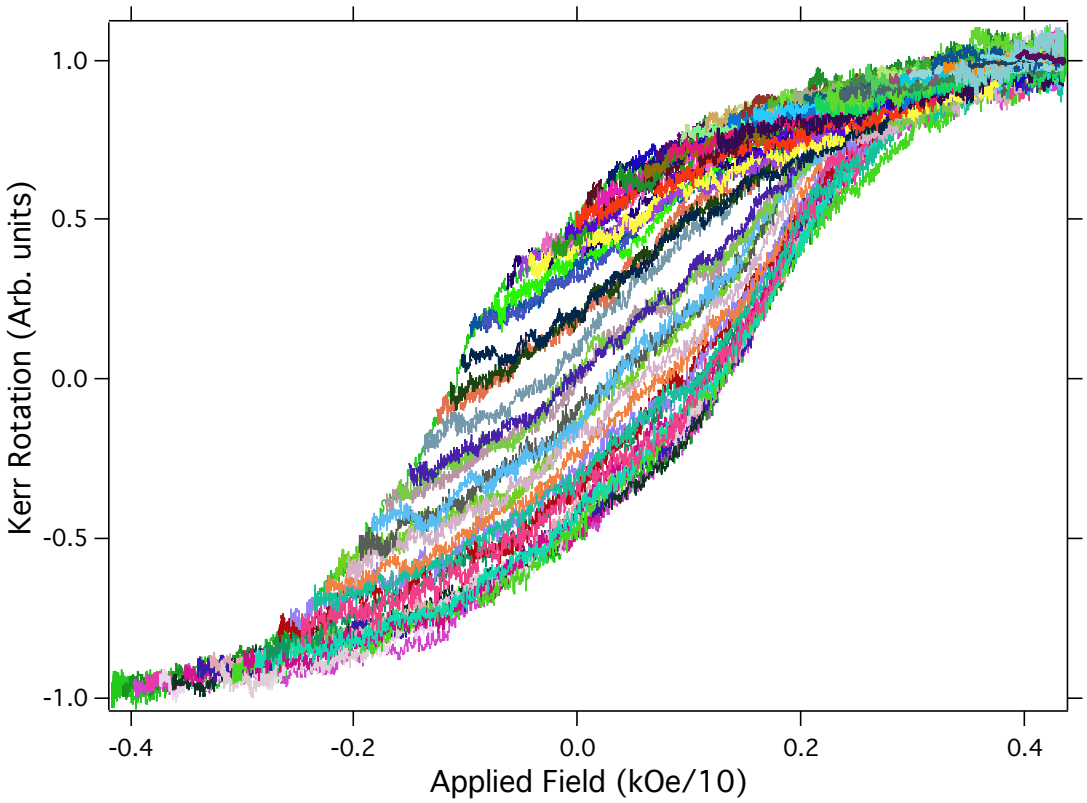

Figure 7.38 Implementing the error-finding Igor Pro procedures, we are able to pass 8000 individual curves through a series of procedures which yield a normalized set of 80 FORC curves free from erroneous data. Although this data is free of detection issues, the rates of the magnetic fields had not been equalized at the time of data acquisition.

When normalizing a set of FORC data, we must be a bit clever about how we choose to do the normalization. In a normal MOKE setting, all of the curves are normalized by taking the maximum and minimum values of the curve and scaling them to ± 1 respectively. In the case of a FORC curve, although the maximum value is consistent, the minimum value changes for each FORC reversal (Figure 7.38). Thus, rather than insisting, via mathematics, that the maximum and minimum values are rescaled to ± 1, we use the knowledge of magnetic hysteresis, and note that every reversal reaches the same max field value. Because the history, as far as the fiber is concerned, shows that the magnetization has saturated, it will follow the same magnetic path from H_{max} to H_r for all different reversal field values. This is shown in Figure 7.39 with a selection of full loops from the same data set as 7.38.

With this knowledge, the maximum value for normalization is the maximum Kerr rotation associated with H_{max} while the minimum value for normalization will be the value of the Kerr rotation of the outermost reversal curve at the specific H_r. The outermost curve is normalized to the expected ± 1, while all minor reversals are normalized to the corresponding point within the larger curve based on the value of the Kerr rotation following the magnetic field from H_{max} to H_r for each reversal.

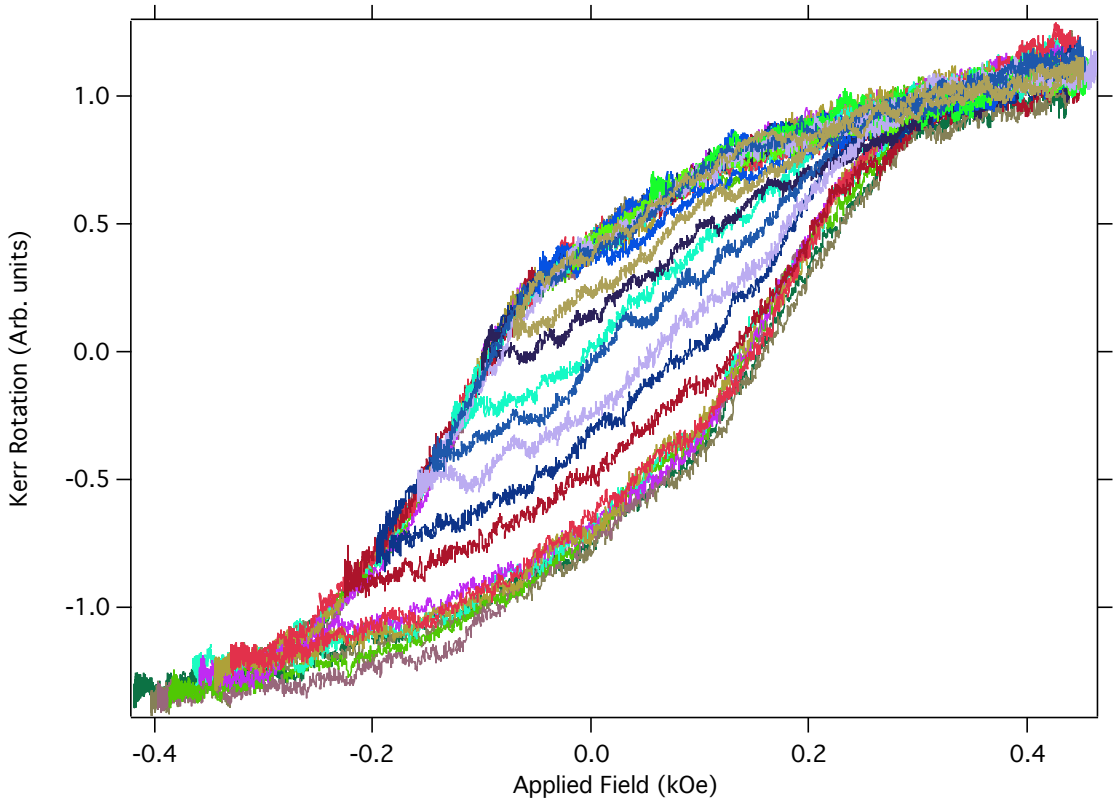

Figure 7.39 Using the same data as Figure 7.38, we see that the non-normalized FORC reversals will all follow what appears to be the same reversal path because they all reach the same saturation field value.

The data above are technically the type of data we need to pursue further FORC analysis but the random noise is far too high to perform adequate derivative analysis. At this point we employ two different noise attenuation tactics. The first is a noise reduction within the oscilloscope itself [59]. The oscilloscope has an option to improve resolution of a noisy

signal by means of bandwidth reduction. In the particular case of future data, we employ the resolution enhancement to reduce the bandwidth of our signal by a factor of 8. The second layer of smoothing is employed within our Igor Pro analysis (section A.3.5 in the Appendix). This next type of noise reduction uses the LOESS (Locally weighted smoothing) algorithm to interpolate a smoothed function at each point based on a user provided number of neighboring points which are weighted based on proximity to the center point being investigated. LOESS smoothing is especially useful when a set of data is both noisy and has an intrinsic curvature. The smoothing process will result in a set of data with lower standard deviation while preserving the interesting curvature of the data (Figure 7.41).

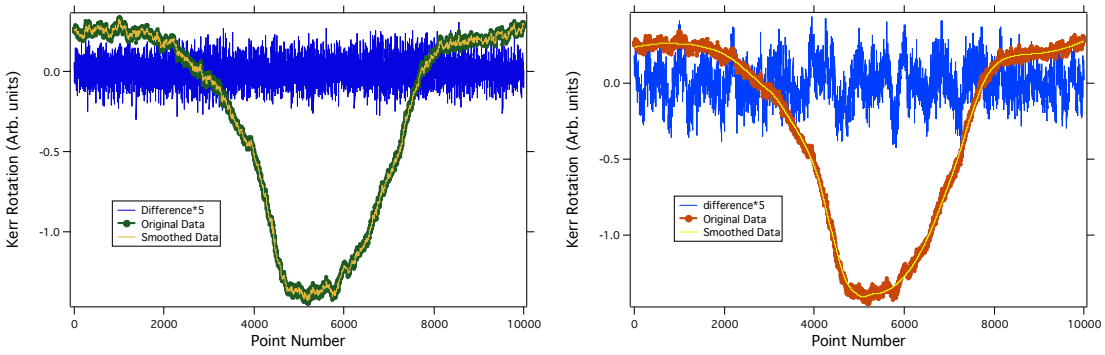

Figure 7.40 LOESS smoothing applied to the same set of data with different smoothing factors. We judge the quality of smoothing based on the difference between the original data and the smoothed data. The distribution of data is largely Gaussian in the first image, while the data on the right, which is over-smoothed, shows various peaks and valleys in the difference, suggesting that certain features have been removed via smoothing.

With the addition of these two noise reduction techniques and the modified, constant rate, field steps, we attempt a FORC data collection with 150 reversals.

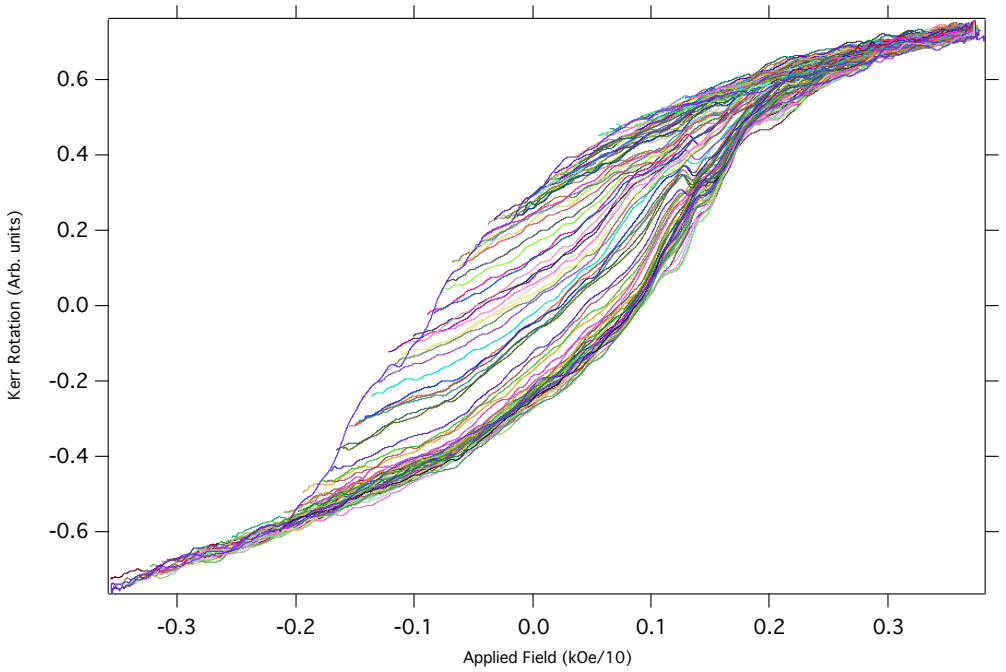

Figure 7.41 Using the bandwidth reduction from the oscilloscope, Igor Pro-implemented LOESS smoothing, and rate fixing, we acquire our first low-noise set of FORC loops with ScMOKE. Although the reversal spacing corresponds to 150 reversals, we only show the first 135 here. Beyond reversal 135, and even before that, as we will see in Figure 7.52, there is a point where the magnetization process becomes fully reversible. Once we are beyond this point, the FORC analysis will never reveal any interesting information.

A few interesting insights from this data. First, we notice that the data is not running between ± 1 on the Kerr rotation axis. When looking closely at the full loops, it appears that the largest outer loop does not fully describe the demagnetization process of all subsequent minor loops. This particular effect was undetectable in previous runs, such as Figure 7.38, where SNR was too low to detect these subtle effects. This may be due to minor loop behavior, lack of measurement precision, over smoothing, or some other effect. The truth is that although the minor loop rotations were executed as expected, the path from H_{max} to H_r differed from one reversal to the next. In Figure, 7.42, we see that as the reversals progress, the demagnetization path changes very slightly between reversals. As a result of this apparent effect, we choose to simply offset the curves so that each Kerr rotation at H_{max} is aligned with all other curves.

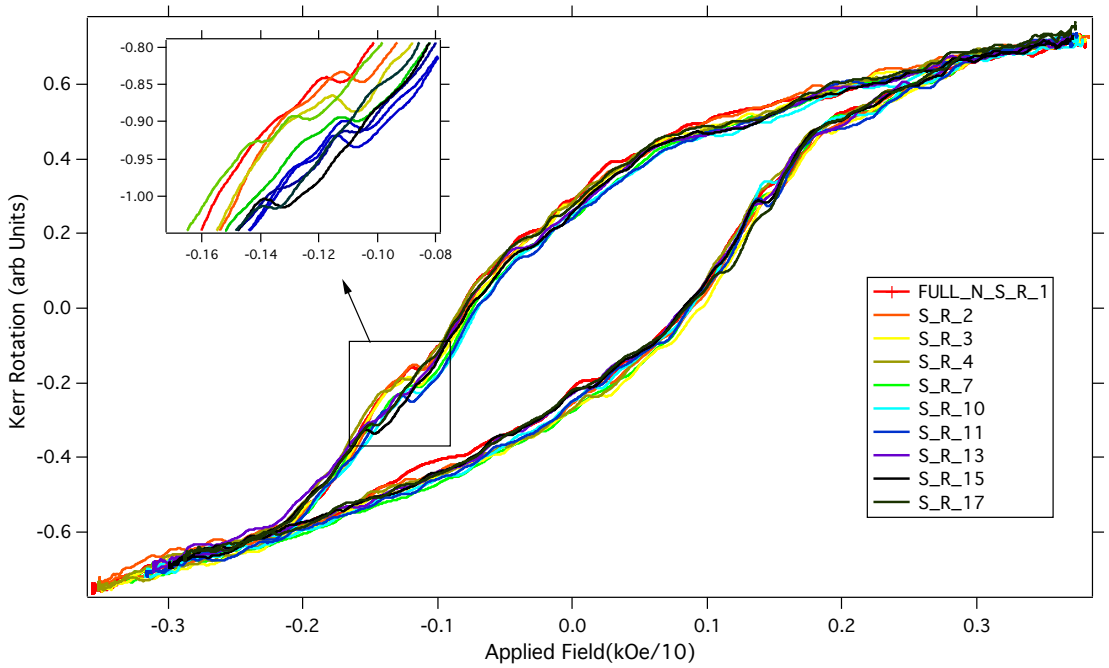

Figure 7.42 The legend indicates which reversals are plotted. The area which has been expanded shows an example of how the field path from H_{max} to H_r is not consistent between consecutive reversals.

While this may be concerning for future analysis, we are only interested in the slope of these data, rather than the raw values of magnetization. This unexpected diversion from a normal path was verified to be a real effect with a second FORC trial, taken to 53 reversals, rather than the full 150, with 200 averages per reversal. The next interesting piece of information from Figure 7.41 is the successful linear spacing of the reversal field values (Figure 7.43). Using a lookup table made with $\sim$3 sec between points, we selected only a final current value which would yield linearly spaced reversal fields. The number of steps between H_{max} and any H_r was calculated by the LabVIEW program to ensure constant rates between reversal fields, as we have already shown in Figure 7.34, by implementing the Equation 7.16 into our LabVIEW program.

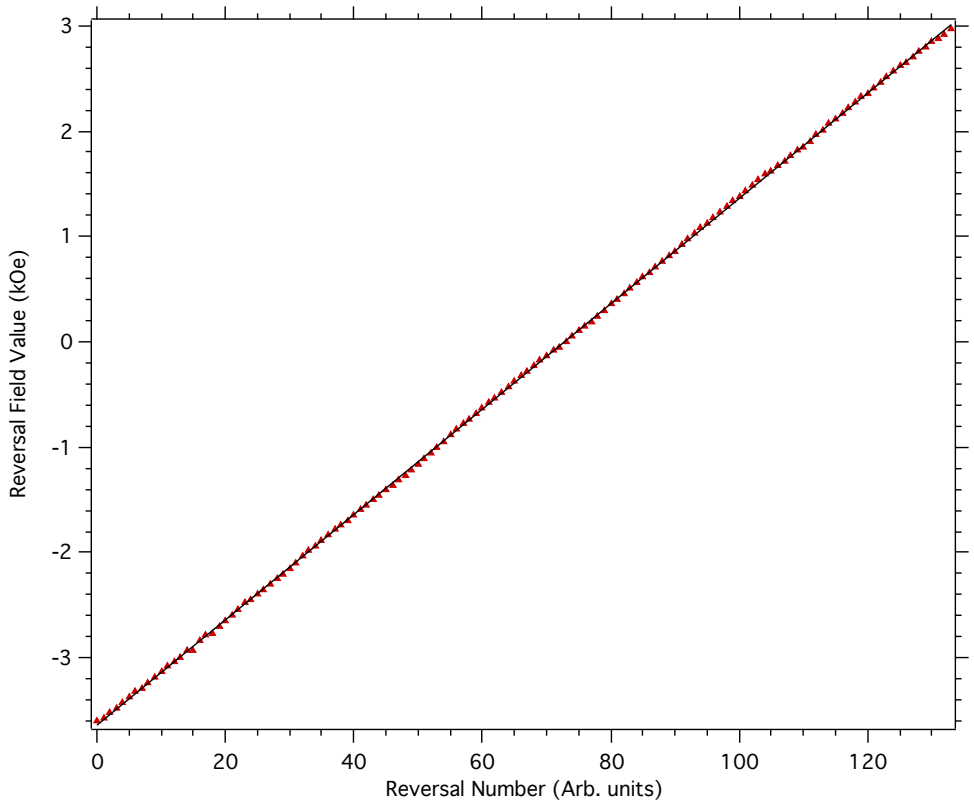

Figure 7.43 Reversal field values for each of the 135/150 reversals. There is minor deviation from the linear fit, yielding ∼50 Oe spacing per reversal.

Finally, I will save the most important and most motivated point for discussion here. The choice to pursue FORC as a measurement technique was motivated by the various persisting changes in curvature that can be seen in many of the Janus hysteresis loops in Figures 7.27 or 7.28. Originally, we suspected that these curvature changes were real magnetization effects, due to their persistence between consecutive runs and multiple averages. FORC measurements specifically investigate these types of changes in curvature as a function of reversal field, yielding information regarding interactions between a multi particle system. Because our Janus fiber agglomerates are a multi-fiber system, we expect some sort of reversal dependent effect. Rather than viewing all of the data in a hysteresis loop with minor loops within, which can be difficult to visually distinguish, we identify the individual reversals where interesting behavior occurs and stack them vertically (7.44).

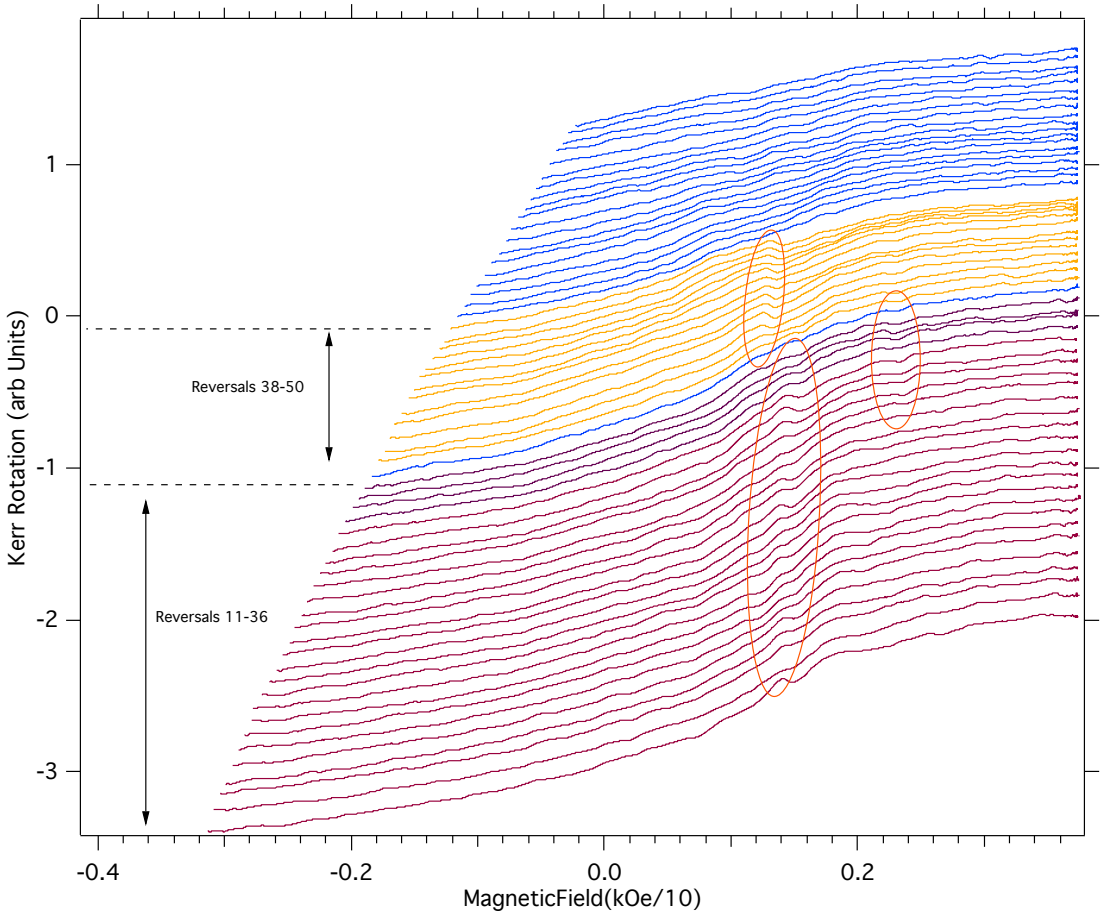

Figure 7.44 Reversals 11 through 69. The circled areas show relatively sharp deviations in an otherwise smooth magnetization process. The larger circled area (reversals 11-36) persists through many reversals and gradually becomes a smooth magnetization process. The smaller area begins gradually at reversal 38, persists, and abruptly disappears at reversal 51.

This type of feature is highly motivating for the FORC analysis. A final FORC density plot is built by executing $\frac{\partial^2 M}{\partial H_a \partial H_r}$. With data that promises an interesting set of derivatives, we now focus on taking these derivatives. With our data being nominally smooth, we look to the definition of a derivative. A derivative is nominally just the slope at the point being differentiated. Igor Pro software has a built in differentiate function that will take the slope between adjacent points in order to find a derivative, but this is very susceptible to noise and fluctuations in data amplitude. Rather than the in-built 'differentiate' function, we find the slope at the point of interest in a way that is more resistant to noise. We allow the user

to pick a symmetric range about the point of interest and apply a linear fit to that range of points. Based on the indexed proximity of contributing points, we assign a Gaussian weight to each point. A basic diagram of this can be seen in Figure 7.45, and the corresponding Igor Pro procedure can be found in the Appendix (section A.4.1).

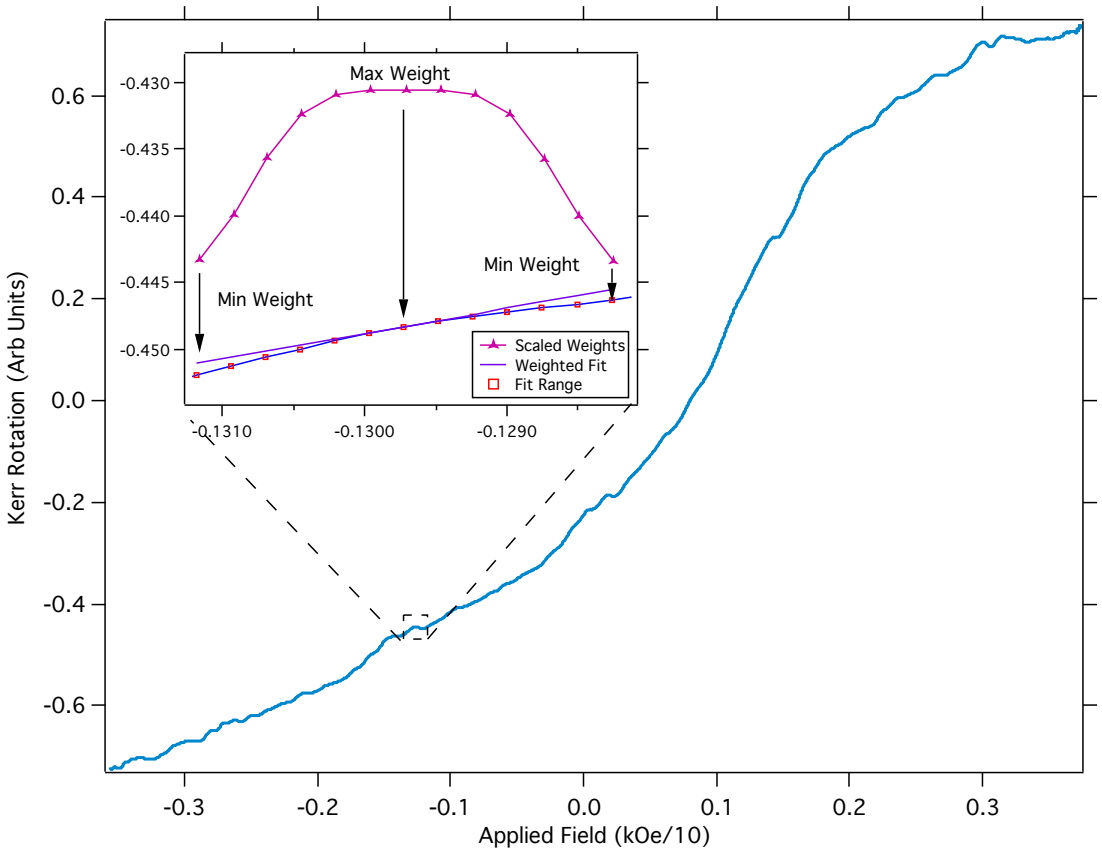

Figure 7.45 An example of a differentiation using a custom Igor Pro procedure (Appendix section A.4.1). The large graph shows the full reversal from H_r to H_{max}, while the inset shows the subset of points in which the differentiation is taking place. In the inset, we see the range of points contributing to the fit, with the middle point as the point physically being differentiated. Overlaid above is the weighting function used. The farthest point is given a 10% weight, and the other points weight are calculated based on that. If we want a sharper weighting function, we simply reduce the contribution of the furthest point. Each slope value is taken for every point for each reversal and saved as a new piece of data.

Similar to the LOESS smoothing process, each derivative is checked by integration and compared to the initial data. A "good" derivative is one with a Gaussian-like distribution

of values for the difference between integrated and original data. Like the LOESS data, a larger contribution of points will effectively over-smooth the integral, resulting in a loss of information regarding the subtly changing curvature of the data. From the process shown in Figure 7.45, we build a whole new set of data of differentiated reversal curves. At this point, before taking the next derivative, we must once again fuss over the state of the X axis. We know that the next step is a derivative of these slope data with respect to other reversal curves for each point H_a. In order to do this with the same method as the first derivative, we must have slope values at every applied field value, or, as I have done, insist on a fixed number of field step points and interpolate slope values for these points. Because of the way we have taken the first derivative, the differentiated function is smoothly changing and has no discontinuous edges. The reversal portion of the data contains up to $\sim$4400 data points for the largest reversal, and as few as $\sim$1000 points for the smallest reversal. We have started with such a large density of points so we can eventually reduce our number of points while still being confident in the accurate representation of our data. To accomplish this uniformity of field spacing, we insist on a desired number of final points, and, using H_{max} and the largest H_r, we build an array with these ideal field steps. For each differentiated value, the value of the derivative is interpolated to the desired field value such that we will have a singular shared field axis, rather than a field axis corresponding to each reversal, with minor differences in field steps. Figure 7.46 shows a mass of differentiated reversals with equivalent field steps which have been interpolated from each of the individual reversal's corresponding field data. One will notice in the Figure that there are long horizontal sections of data with unchanging slopes. This is done to equalize the total number of points between reversals. We simply extend the first value of the slope to $H_{r_{min}}$ for each reversal as suggested by previously explored FORC analysis techniques [60].

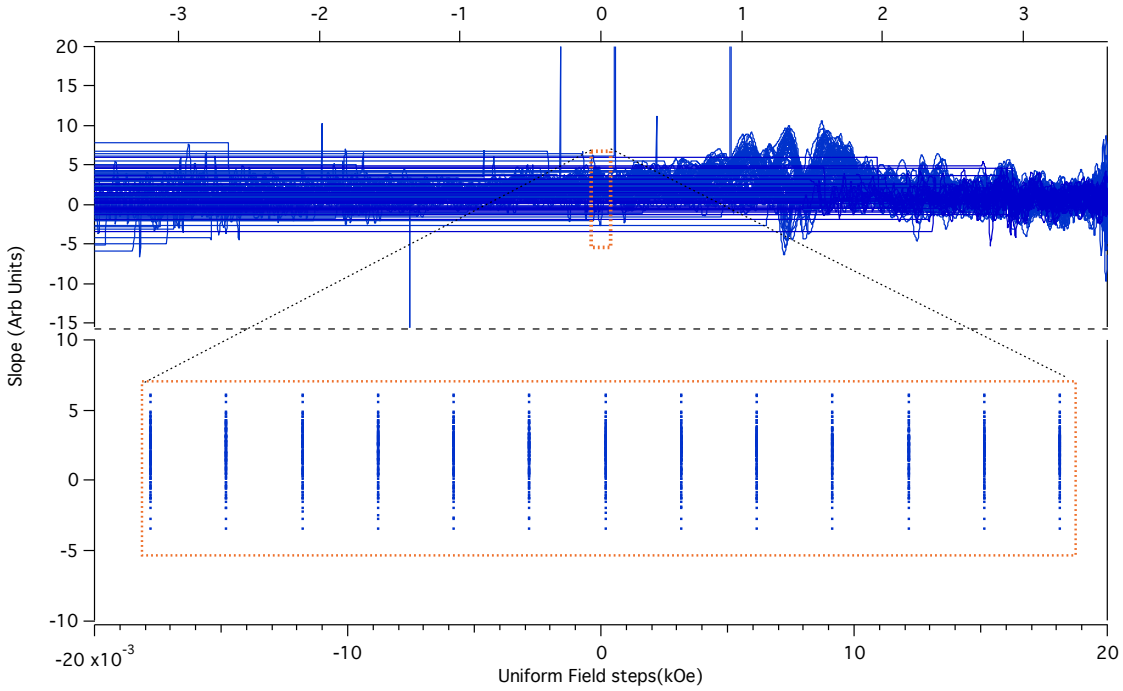

Figure 7.46 The top graph shows all 135 derivatives as a function of the new uniformly interpolated field spacing. The particular field spacing above is created by insisting that there should be 2405 points between $H_{r_{min}}$ and H_{max}. The bottom graph shows a subset of data from the highlighted area in the top graph. Using linear interpolation, all reversals now share a uniforml spaced field axis.

This step is important for two reasons. The first is that we want to facilitate the second derivative with the least amount of trouble as possible, and by having all values from the first derivative lying on the same field values, the second derivative will not require any interpolation when differentiating between reversals. Second, the presentation of our data in matrix form is highly illuminating, and will be relevant to the final form of the FORC distribution. To this point, when creating a matrix in Igor Pro, there is no option to include axes, and, instead, the both axes are simply indexed by row and column number. The change to uniformly interpolated field steps ensures that each indexed point shares the same field value. As we see from the top image in Figure 7.46, viewing all derivatives in a graph makes identifying reversal-based changes in the slope difficult. Instead, we use our newly uniformly spaced field values to build a matrix of reversals where the vertical

axis is the different reversal values, while the horizontal index is uniformly-spaced field values from $H_{r_{min}}$ to H_{max}. This matrix will satisfy the first of two necessary derivatives in $\frac{\partial^2 M}{\partial H_a \partial H_r}$, with the first being a derivative of our Kerr data (magnetization in arbitrary units) with respect to applied field (H_a), which is shown in Figure 7.47.

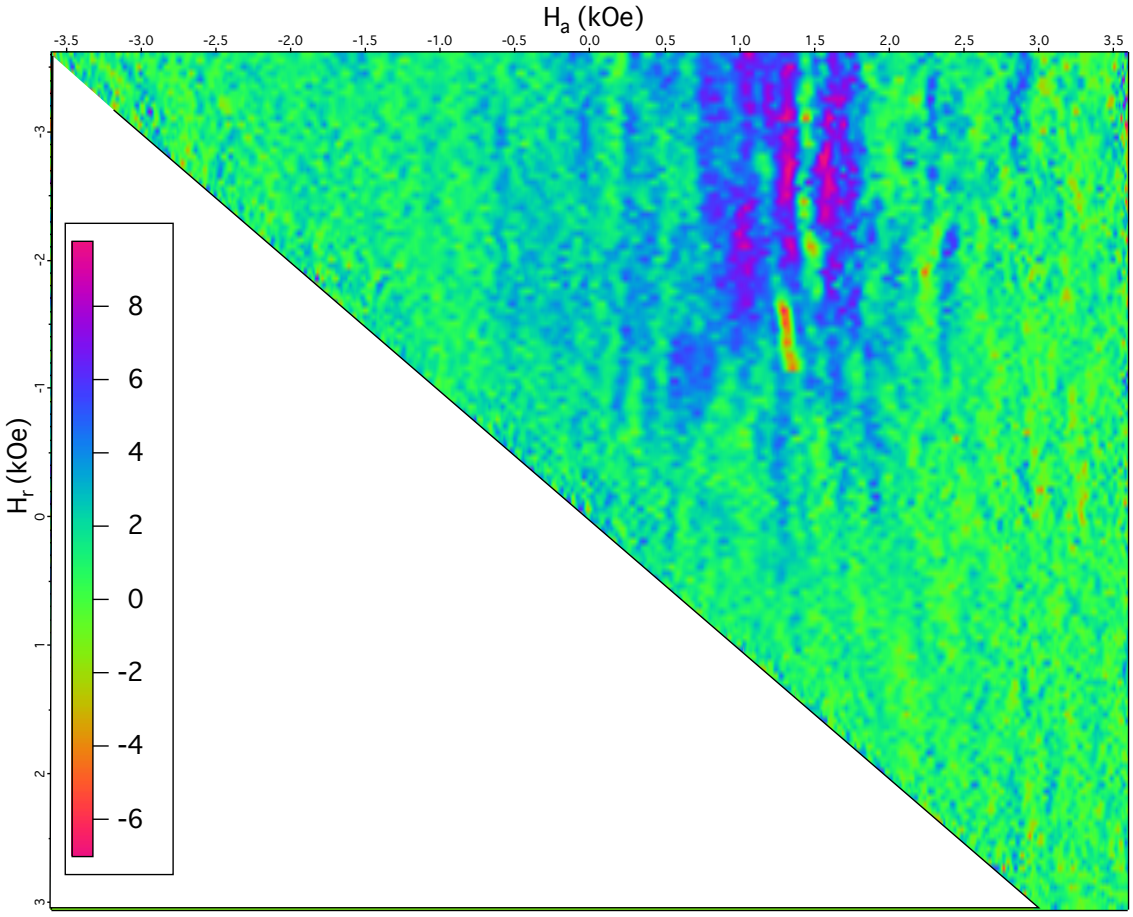

Figure 7.47 An illuminating view of our first derivative. The horizontal index shows the interpolated and equalized applied field steps and the vertical index shows the reversal field values. The contour colors represent the values of the first derivatives. Although the data is far from noise-less, we see significant instances of rapidly changing values of the first derivative in reversals 1-65, between -$H_{r_{min}}$ and $H_r = 0$ kOe and $0 < H_a < 2$ kOe. This corresponds to the previously seen features in Figure 7.44.

We see in the above image a matrix the mathematical result of the visually identified features from our 135 FORC reversals. We previously saw that there were notable swift changes in the curvature of the reversals in Figure 7.44. In addition to those which were

manually identified, we see additional changes in curvature, for example, beginning around H_a = 2.35 kOe, H_r= -2.5 kOe. This is just one of the two derivatives necessary to define a FORC density plot. As stated earlier, we seek to find $\frac{\partial^2 M}{\partial H_a \partial H_r}$, and we have now completed $\frac{\partial^M}{\partial H_a}$. The next derivative is $\frac{\partial^M}{\partial H_r}$, which we carry out by differentiating our slope data (first derivative) shown in Figures 7.47 and 7.46 with respect to the reversal field. Essentially, we are interested in how the slope curvature changes based on which reversal path is being used. We use the same method as before to find the derivative, a weighted linear fit within a subset of points surrounding the point of interest, with comparisons of the integral to the original function. The result of the second derivative with respect to H_r, in matrix form, is shown in Figure 7.48

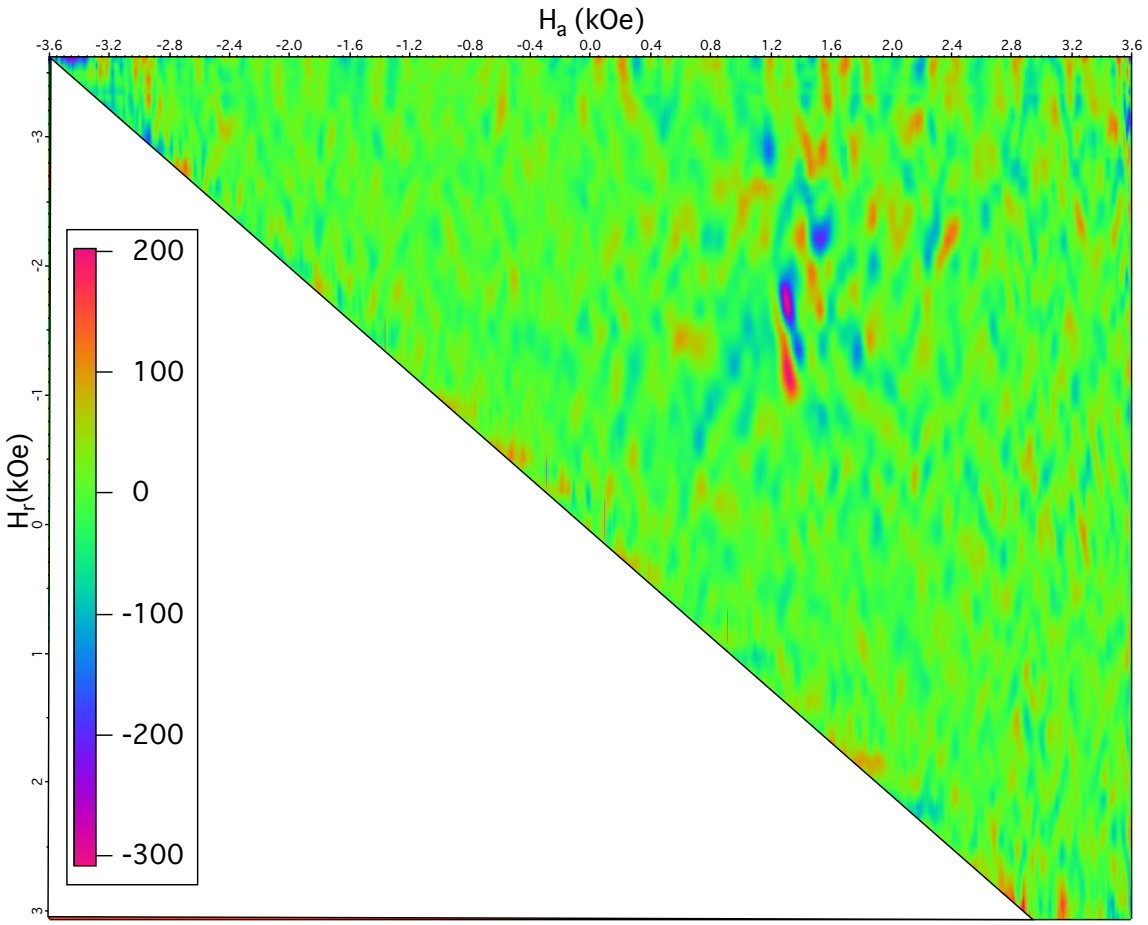

Figure 7.48 As expected from this type of differentiation, we have introduced a significant amount of noise into our resulting data. We see the result of our second derivative with respect to H_r. The sharpest values of the graph represent stark changes in the slope between consecutive reversal field values. Again, as we expected from manual identification of magnetization curve features, we see our most notable signal along $H_a = 1.2$ kOe.

Figure 7.48 shows the result of the second derivative with a large quantity of induced noise. This is to be expected, however, due to the interpolation process, and the use of numerical differentiation. Between H_a = 1.2 and 2.4 kOe we see several sharp spikes that were expected from visual identification, and several more that have come from the differentiation process. In this particular graph, we have chosen a subset of 6 contributing points on either side of the indexed differentiation point. Increasing or decreasing the

number of contributing points will result in a final graph lacking in contrast or being too sensitive to small changes in differentiation. We verify our results with the common FORC analysis method found in the literature [34]. Rather than finding the approximation of a derivative via linear fits at every point, literature suggests a polynomial surface fit following eq 6.2 using the coefficient K_4 as the term representing the second derivative. Figure 7.49 shows the resulting data from both methods side by side.

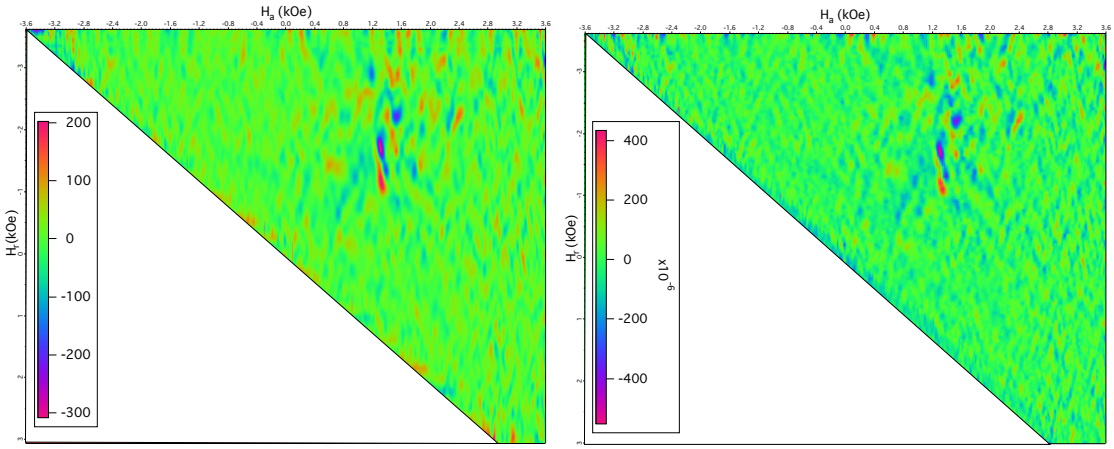

Figure 7.49 The left image is the result of our manual differentiation using Gaussian-weighted linear fits to determine the two derivatives, while the image on the right shows one of the typical approaches from the literature. Visually identifiable features are nominally identical, while minor features such as noise vary between methods.

The polynomial fit approach was carried out at each point by fitting a polynomial surface to a 3x3 subset of points centered on the point of observation. When fitting near an edge, the size of the box is reduced corresponding to the point of investigation's proximity to the nearest edge. For both methods, by changing the size of the respective "boxes" in which fitting can occur, we will either increase resolution at the cost of more noise, or decrease noise at the cost of less resolved features. With both derivatives completed and our data verified, we execute the final step, which, as stated earlier, is a change of axes:

$$y \rightarrow H_u = \frac{1}{2}(H_a + H_r) \tag{7.18}$$

$$x \rightarrow H_c = \frac{1}{2}(H_a - H_r). \tag{7.19}$$

This transformation is carried out by calculating a new matrix from each point using the above relations. The resulting symmetric graph is shown in Figure 7.50

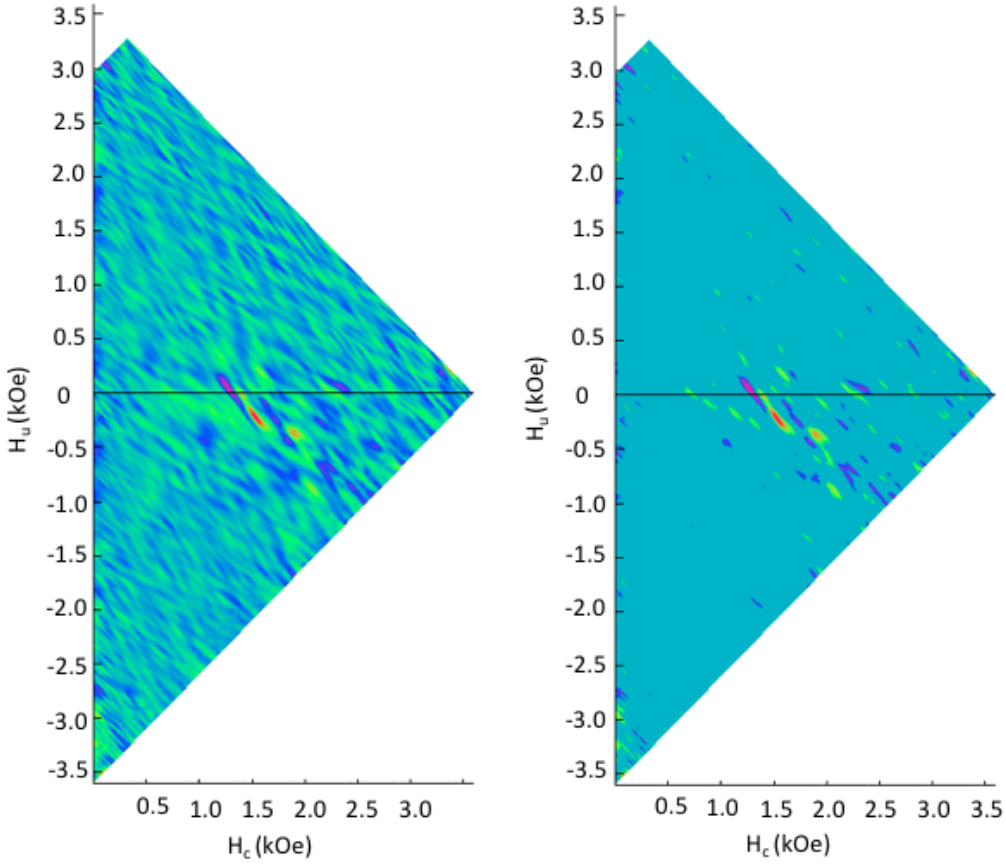

Figure 7.50 Upon rotation we see that our data is now symmetric about $H_u = 0$ axis, and the H_c axis only extends in the positive direction. The left image is the result of our second derivative, while the right images is the same data where we have applied a threshold to reduce the background noise for easier data interpretation.

7.7 Analysis of FORC

As mentioned earlier in the FORC theory section, much of the available analysis for FORC is qualitative to date, and relies heavily on visual identification of certain features along the H_u and H_c axes. Fortunately, the most common and notable effects have been mentioned in Chapter 6. One of the easiest quantitative measurements that can be extracted from a FORC diagram is the average coercivity of our array of nanofibers. This is simply done

by identifying the H_c coordinate of the largest peak on the $H_u = 0$ axis. In our case, from Figure 7.50, the peak appears at $H_c \sim 1.34$ kOe [58]. The coercivity of our major hysteresis loop (MHL) is ~ 0.89 kOe. While it might be shocking that these values are different, it is actually quite encouraging. The MOKE measurement of the MHL is a "bulk" measurement of the fiber agglomerate, which is nominally measuring a "3"μm diameter cylindrical wire. Literature tells us that the coercivity is inversely related to wire diameter with a large fiber diameter possessing a smaller coercivity [61]. We also know that the post-processed fibers created by our collaborators have a nonzero distribution of diameters and thus the individual fibers which make up an aligned fiber agglomerate will also have a distribution of diameters [62]. The larger coercivity gathered from the FORC diagram is indicative that the average diameter of a nanofiber building block is significantly smaller than the diameter of the agglomerate that it forms. Without further tests investigating the polydispersity of individual fibers, this cannot be said for certain.

Another piece of information that can be extracted from FORC data is the "reversibility" of each of the minor loops [63]. Rather than looking at the final FORC diagram, this data is gathered from individual reversals. By comparing the slope of the outer MHL to the slope of individual reversals at the point of reversal we can reveal the reversibility factor (η) of each reversal.

$$\eta = \frac{\chi_{rev}(H = H_r)}{\chi_{MHL}(H = H_r)}, \tag{7.20}$$

where χ_{rev} is the slope of the reversal and χ_{MHL} is the slope of the outer major hysteresis loop. Both of these slopes are taken at $H = H_r$, the individual reversal points. From this derivative we will hopefully find this ratio ranging between 0 and 1 with 0 indicating total orthogonality and 1 indicating a fully reversible process. Using our same software for weighted slope fitting, we calculate this value at every reversal and show the results in Figure 7.51.

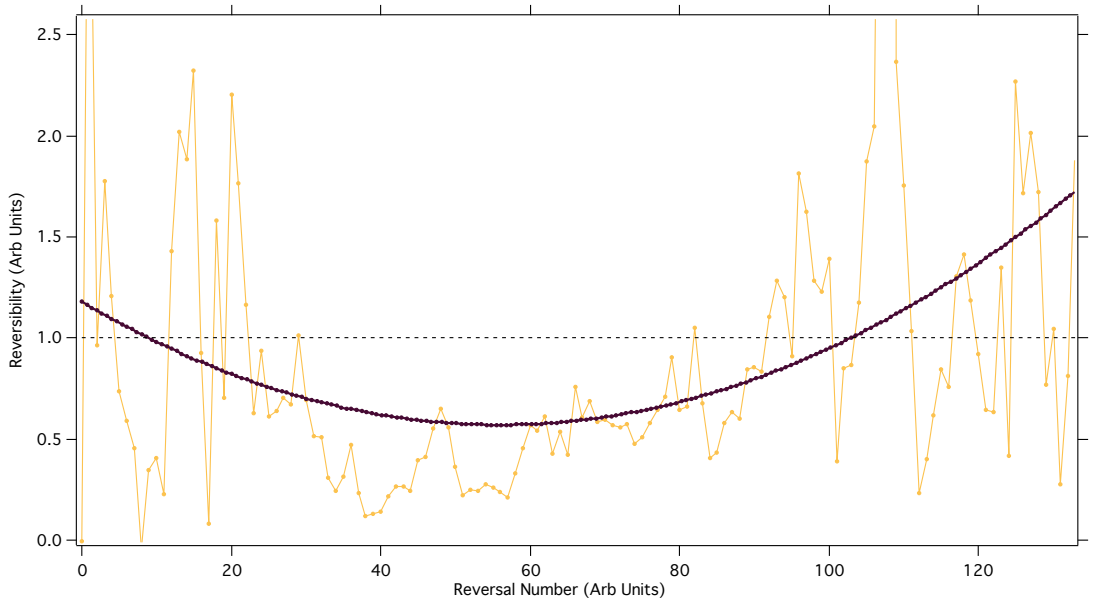

Figure 7.51 Reversibility as a function of reversal number. A polynomial fit is applied to guide the eye. The ideal value of 1, indicating total reversibility is indicated on the vertical axis.

Unfortunately, the MHL does not accurately represent the reversal process of each individual loop as we have already mentioned in Figure 7.42. Thus this data may not tell us exactly what we hope. From the graph above we do see a general trend toward irreversibility as we get closer to reversals 40-60, and toward values closer to 1 (reversibility) as we reach either end point of reversal number. Alternatively, I will define a different reversibility constant η_{rel} referring to the relative reversibility of each curve, using its own descending magnetization curve rather than that of the MHL since we have seen that the individual reversals deviate significantly from the MHL path.

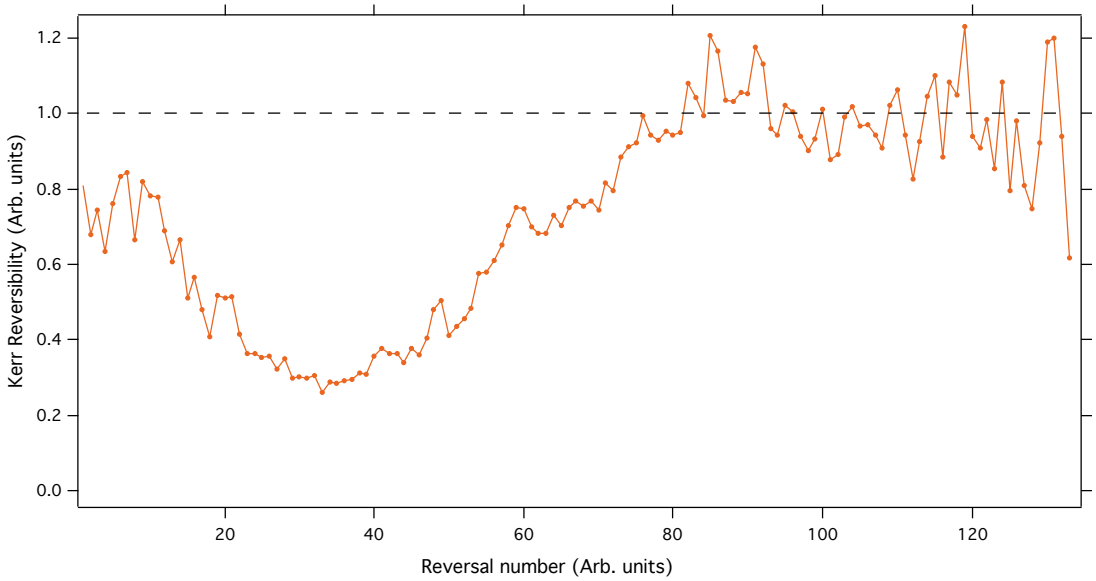

Figure 7.52 Relative reversibility found using the descending curve from each reversals.

Because the start points where reversals occur are plagued with end behavior defects as well as a reduction to fitting contribution due to boundaries, a reversibility rating is able to supplement information regarding the H_u axis when $H_c = 0$. Figure 7.52 is indeed much cleaner of a relationship, but this is partially due to the large amount of data points in each individual reversal allowing for a more robust "slope" to be taken and the fact that the reversibility for each reversal was taken from each half of a single reversal curve (descending and ascending), which allowed us to ensure that they actually connect at the H_{r_i} value. We see from our relative reversibility data that there are no truly irreversible reversals, and we see a nearly monotonic increase in reversibility until about reversal 80 where we are completing fully reversible magnetization processes.

As mentioned earlier, a feature that often arises in systems of nanorods is the wishbone shape, appearing as a result of having magnetic domains under the effect of a local interaction field which acts to demagnetize the overall magnetization. The characteristic shape is a line crossing the H_c axis while being at a nonzero angle with respect to the H_u axis. From the end of this line in the $+H_u$ direction, we will often see a second, less intense,

line returning to connect with the H_c axis at an angle. 7.53 shows an example of this type of wishbone shape, adapted from the literature [64, 37]. The angular section returning to the H_c axis typically will represent the switching back of the flipped hysterons, and the length along the H_c axis is related to the distribution of coercivities of the representative hysterons.

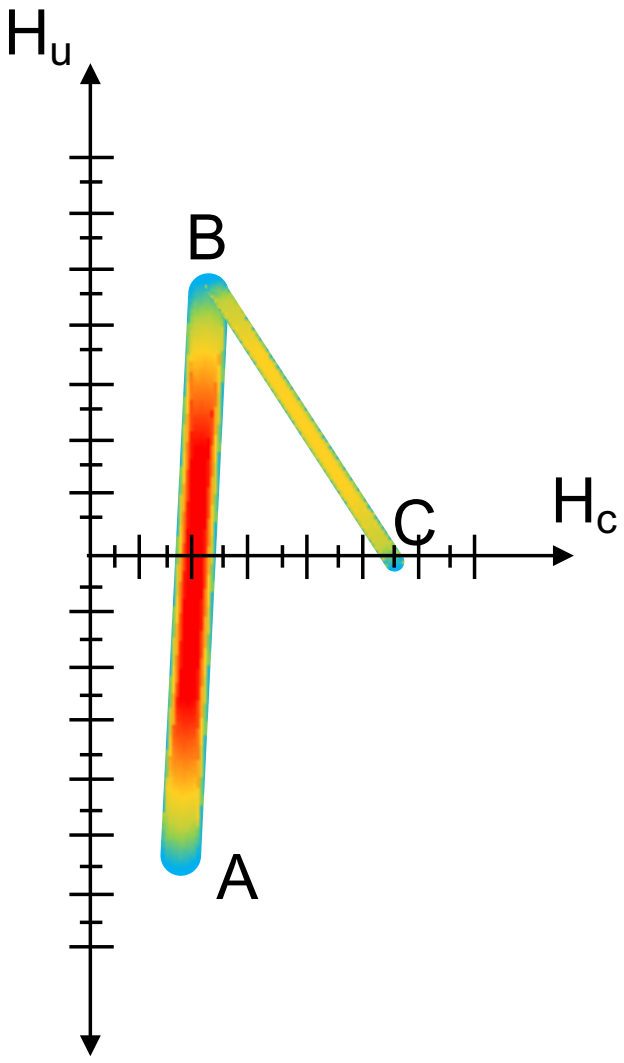

Figure 7.53 A characteristic wishbone shape originating from the distribution of hysterons with different coercive switching fields under the influence of a demagnetizing interaction field. The path A-B represents the initial switching event, while B-C represents the hysterons switching back.

Unfortunately, this type of shape can also appear flipped about the H_u axis, and although it may look similar, it represents a different magnetic process, so we will refer to it as the "V". The key difference can be seen with the appearance of large negative regions surrounding the "V" shape. Figure 7.54 shows this type of shape, where each positive region has a corresponding negative region in the FORC distribution. Additionally, the part crossing the H_c axis occurs at a larger value of H_c compared to the sloped piece which returns to the H_c axis.

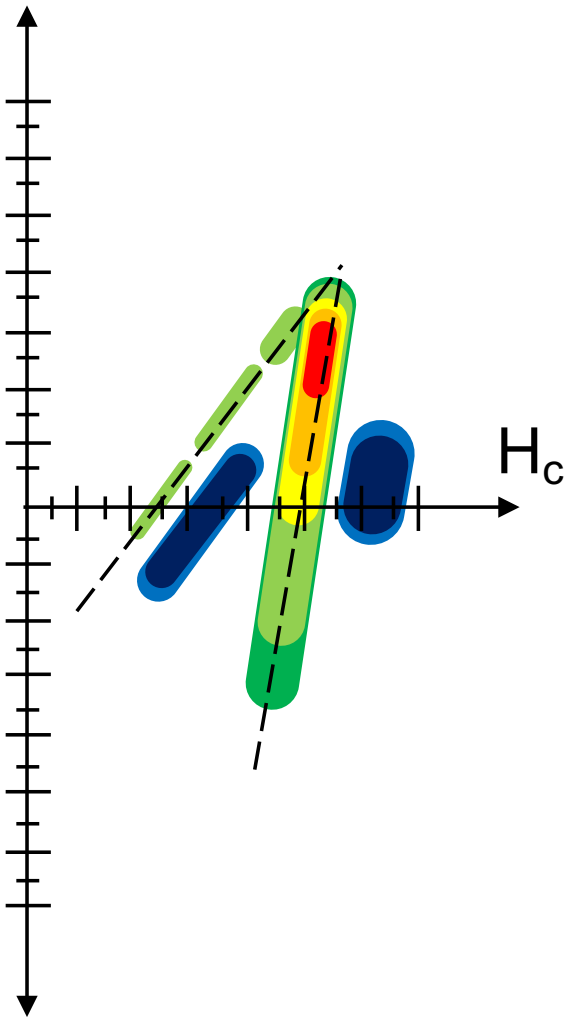

Figure 7.54 The "V" shape which is distinct from the wishbone, although they look similar. The "V" shape will show with a distinct singular maximum near the vertex of the "V" (Red spot) while there will also be accompanying negative regions (shown in blue).

While the wishbone and "V" look the same, the position of key features along the H_c axis serve to differentiate the two. Comparing previous data from literature to the appearance of this "V" shape, it seems that this appears when minor loop reversals cross the MHL upon reversal. This effect is indicative of the appearance and subsequent disappearance of an out-of-plane component of magnetization [40, 41].

Looking at our thresholded FORC diagram, it seems like we have a situation in which we observe the "V" shape, rather than the wishbone. Figure 7.55 shows an expanded view of the "V" region.

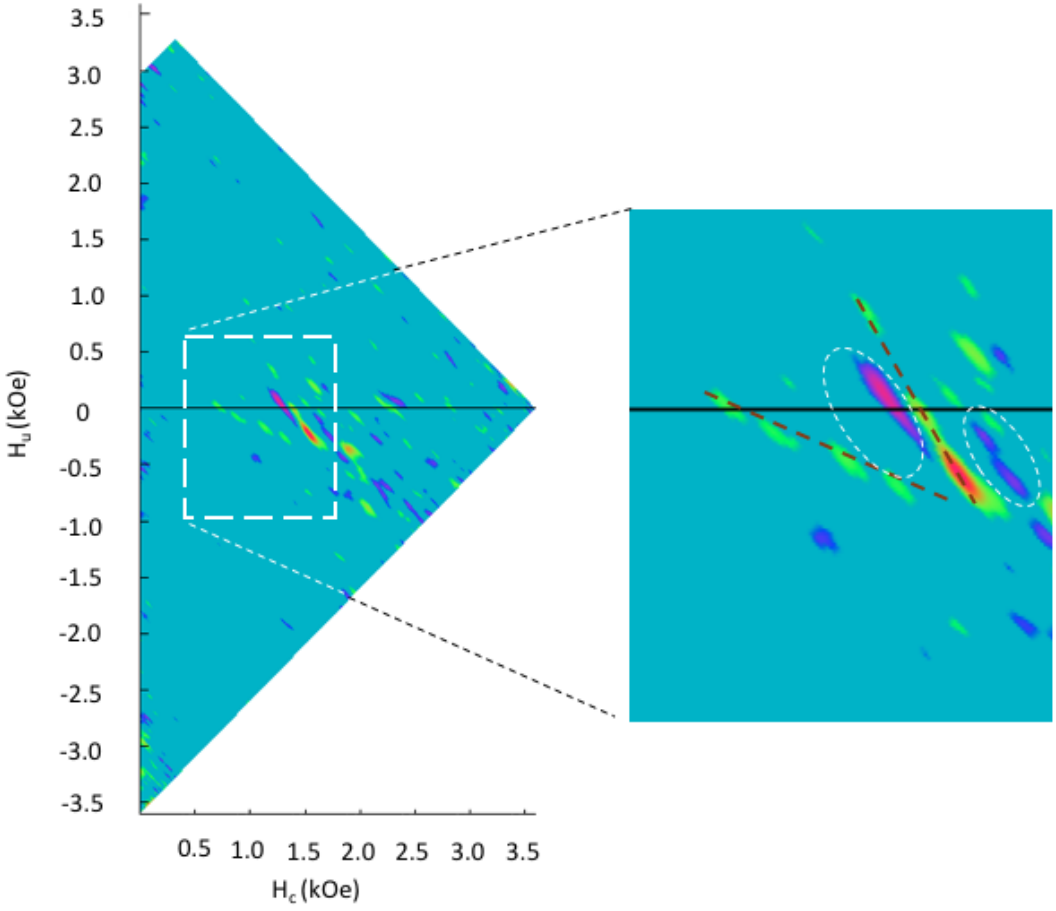

Figure 7.55 From our FORC diagram, we see the "V" shape, which identifies the crossing of the MHL by the reversals, and indicates the existence of an OOP magnetization. The dark dotted lines trace the shape of the "V" while the circled areas show the negative regions.

Looking at the full FORC diagram in the left image of Figure 7.55, we see multiple pairs of positive and negative contour pairs. This has been seen to occur in systems where discrete magnetic orientations occur as a result of a noncontinuous distribution of coercivities in hysterons [41]. This particular effect is more easily identified in systems where the distribution of magnetic materials is known, such as a series of patterned wires.

While processing data, care was taken to minimize the loss of information as much as possible given the instruments which were available. However, between smoothing, bandwidth reduction, fitting, the methods of differentiation, it is likely that information has been obscured, smoothed over, or replaced. When viewing our data using the H_r and H_a axes in Figure 7.47, we took care to align the H_a axis by reducing the number of equivalently spaced points from a larger set of points by means of linear interpolation. This was not done, however for the reversal fields. Figure 7.43 shows that the reversals follow a linear fit closely, though there are clearly points that do not lie directly on the line. With our total field range of $\sim$7.3 kOe, and the reversal spacing designed for 150 reversals, that means that the ideal spacing is roughly 50 Oe. Looking back to Section 7.3, we noted that the minimum resolution available to our magnetic probe was 25 Oe. It follows that our H_r values have error of 50%. We attempted to correct the "slowly changing field" stepping behavior by replacing saturation parts of the field graph with exponentials that appeared to match seamlessly to the rest of the data, but without verification from a fully resolved magnetic probe, we cannot be sure.

Additionally, when acquiring FORC data, each reversal was acquired separately from each other reversal data. In other words, for each reversal 200 consecutive MOKE runs were completed to build a sufficiently low noise data set before moving to a more positive reversal value, which then was subsequently ramped to H_{max} before returning to the new reversal value for the next 200 averages. It is unclear how other experimenters execute data acquisition, but we may see a different magnetic response, which may match more closely to the MHL, if we were to complete every reversal in order from most negative to

least negative as a single run, and then repeat that pattern 200 times. Figure 7.12 shows an example of this type of field stepping but only for 3 reversals. The former was chosen as the processing and separation of data would be slightly less cumbersome, but a test of the latter would be instructive on the "proper" way to handle this variable type of magnetic history with the least amount of artifacts.

Lastly, this type of data acquisition takes a very long time ($\sim$80 hrs). Due to how processing is done, and the amount of interpolation therein, we may have the opportunity to reduce the oscilloscope window from 10 to 5 seconds, which would change time/sweep from $\sim$12 to $\sim$7 seconds per run, drastically reducing the total time for data acquisition or for some sort of long term heating related magnetic change to take place. At the end of this research, while we have many many magnetization curves of Janus fiber agglomerates, we have only a singular FORC data set from which to draw conclusions, and, as we know, a trend line cannot be made with a single data point. The measurement seems promising for future work, but with so little data, and many possible errors, the FORC specific conclusions are currently limited.

CHAPTER 8

CONCLUSION

Novel materials seem to dictate new and necessary measurement techniques needed for successful characterization. Born from a collaboration between Dr. Andrew's research group and our own, we were given a novel multiferroic Janus fiber material, which in turn was aligned into discrete agglomerates via extensive efforts within our research group. With few techniques existing to magnetically characterize a small subset of these fibers, we set out to develop an adaptation of MOKE which allows for magnetization measurements on topographically diverse surfaces while maintaining a SNR quality high enough to discriminate between noise and subtle magnetization effects. This was accomplished by taking advantage of the nominally cylindrical macroscopic shape of the aligned microscopic fibers. Rather than using light specularly reflected in a direction following the substrate reflections, we find an ample Kerr signal in light scattered in the 2π angular directions of the scattering plane. By leveraging noise reduction techniques in conjunction with continuous averaging, we are able to use these robust, repeatable scattered MOKE measurements to extend our magneto-optical investigations to FORC measurements. Ideally a successful FORC measurement will yield information regarding the distribution of magnetic coercivities and interactions within a composite magnetic material. Using MOKE in the scattered geometry, in conjunction with careful application of smoothing and noise reduction, we are able to construct a family of FORC curves which lead to a successful production of a FORC diagram. While we are not able to make any absolute claims about the exact magnetic properties of the fibers contained within the larger agglomerate, we have seen clear indications that there are complex magnetic interactions, and possible appearances of out

of plane magnetization components when executing reversal curves. Future work using this characterization technique seems promising. By repeating this measurement for different fiber agglomerates we are likely to see fiber-specific features in all of our FORC diagrams, which will likely lead to greater insight on the nature of how these fibers aggregate and behave magnetically.

BIBLIOGRAPHY

[1] Manfred Fiebig. Revival of the magnetoelectric effect. *Journal of Physics D: Applied Physics*, 38(8):R123–R152, 2005.

[2] R.E Newnham and Et.al. Connectivity and piezoelectric and pyroelectric composites, 1978.

[3] Zheng-ming Huang, Y Zhang, M Kotaki, and S Ramakrishna. A review on polymer nanofibers by electrospinning and their applications in nanocomposites. *Composites Science and Technology*, 63:2223–2253, 2003.

[4] Chuan-ling Zhang and Shu-Hong Yu. Nanoparticles meet electrospinning : recent advances and future prospects. *Royal Society of Chemistry*, 43:4423–4448, 2014.

[5] Alexander V Bazilevsky, Alexander L Yarin, and Constantine M Megaridis. Co-electrospinning of Core - Shell Fibers Using a Single-Nozzle Technique. *Langmuir*, 23:2311–2314, 2007.

[6] Andreas Walther and Axel H E Muller. Janus Particles : Synthesis , Self-Assembly , Physical Properties , and Applications. *Journal of Materials: Chemsitry*, 17:3509–3514, 2007.

[7] S H Xie, J Y Li, Y Y Liu, L N Lan, G J, and C Zhou, Y. Electrospinning and multiferroic properties of NiFe2O4–Pb(Zr0.52Ti0.48)O3 composite nanofibers. *Journal of Applied Physics*, 104(024115):024115–1 –024115–7, 2008.

[8] C L Zhang, W Q Chen, S H Xie, J S Yang, J Y Li, C L Zhang, W Q Chen, S H Xie, J S Yang, and J Y Li. The magnetoelectric effects in multiferroic composite nanofibers. *Applied Phyics Letter*, 94, 2009.

[9] Justin D Starr and Jennifer S Andrew. Janus-type bi-phasic functional nanofibers. *Chem Comm*, 49:4151–4153, 2013.

[10] Bi Fu, Ruie Lu, Kun Gao, Yaodong Yang, and Yaping Wang. Substrate clamping effect onto magnetoelectric coupling in multiferroic BaTiO3-CoFe2O4 core-shell nanofibers via coaxial electrospinning. *Epl*, 112:27002–p1 –27002–p5, 2015.

[11] Bryan L Chavez, Kevin C Sosnowski, Matthew J Bauer, Maeve A K Budi, Jennifer S Andrew, and Thomas M Crawford. Toward nanoscale multiferroic devices : Magnetic field-directed self-assembly and chaining in Janus nanofibers. *AIP Advances*, 8, 2018.

[12] E. Hecht. *Optics*. Pearson education. Addison-Wesley, 2002.

[13] B.D. Guenther. *Modern Optics*. Oxford University Press, 2015.

[14] Mario González-cardel, Pedro Arguijo, and Rufino Díaz-uribe. Gaussian beam radius measurement with a knife-edge : a polynomial approximation to the inverse error function. *Applied Optics*, 52(16):3849–3855, 2013.

[15] David J Griffiths. *Introduction to electrodynamics; 4th ed.* Pearson, Boston, MA, 2013. Re-published by Cambridge University Press in 2017.

[16] J.D. Jackson. *Classical Electrodynamics*. Wiley, 2007.

[17] Van de Hulst. H. C. *Light Scattering by Small Particles*. Dover, 1981.

[18] C. Bohren and D. R. Huffman. *Absorption and Scattering of Light by Small Particles*. Wiley Science Paperback Series, 1998.

[19] Kuo-nan Liou. Electromagnetic Scattering by Arbitrarily Oriented Ice Cylinders. *Applied Optics*, 11(3):667–674, 1972.

[20] R.M. Bozorth. *Ferromagnetism*. The Bell Telephone Laboratories series. Van Nostrand, 1951.

[21] B. D. Cullity and C. D. Graham. *Introduction to Magnetic Materials*. Wiley-IEEE Press, 2 edition, 2008.

[22] John Kerr LL.D. Xliii. on rotation of the plane of polarization by reflection from the pole of a magnet. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, 3(19):321–343, 1877.

[23] Chun-yeol You and Sung-chul Shin. Derivation of simplified analytic formulae for magneto-optical Kerr effects. *Applied Phyics Letters*, 69(26):1315–1317, 1996.

[24] J Zak, E R Moog, C Liu, and S D Bader. Universal approach to magneto-optics. *Journal of Magnetism and Magnetic Materials*, 89:107–123, 1990.

[25] J Zak, E R Moog, C Liu, and S D Bader. Fundamental magneto-optics. *Journal of Applied Physics*, 68(4203):4203–4207, 1990.

[26] Petros N Argyres. Theory of the Faraday and Kerr effects in Ferromagnetics. *Physical Review*, 97(2):334–345, 1955.

[27] Merritt N Deeter and Dror Sarid. Magnetooptical characterization of multilayers by inceident angle analysis. *IEEE*, 24(6):7–9, 1988.

[28] Ferenc Preisach. On the Magnetic Aftereffect. *IEEE Transactions on Magnetics*, 53(2):1–11, 1935.

[29] I D Mayergoyz. Hysteresis models from the mathematical and control theory points of view. *Journal of Applied Physics*, 57:3803–3805, 1985.

[30] Edward Della Torre, John Oti, and Gyorgy Kadar. Preisach Modeling and Reversible Magnetization. *IEEE Transactions on Magnetics*, 26(6):3052–3058, 1990.

[31] C Pike and A. Fernandez. An investigation of magnetic reversal in submicron-scale Co dots using first order reversal curve diagrams. *Journal of Applied Physics*, 85(9):6668–6676, 1999.

[32] Costin-Ionut Dobrota and Alexandru Stancu. What does a first-order reversal curve diagram really mean ? A study case : Array of ferromagnetic nanowires. *Journal of Applied Physics*, 113(043928-1 -043928-12), 2013.

[33] Fanny Béron, David Ménard, and Arthur Yelon. First-order reversal curve diagrams of magnetic entities with mean interaction field: A physical analysis perspective. *Journal of Applied Physics*, 103(7):1–4, 2008.

[34] Richard J Harrison and Joshua M Feinberg. FORCinel: An improved algorithm for calculating first-order reversal curve distributions using locally weighted regression smoothing. *Geochemistry, Geophysics, Geosystems*, 9(5):1–11, 2008.

[35] Philip Sergelius, Javier Garcia Fernandez, Stefan Martens, Michael Zocher, Tim Böhnert, Victor Vega Martinez, Victor Manuel De La Prida, Detlef Görlitz, and Kornelius Nielsch. Statistical magnetometry on isolated NiCo nanowires and nanowire arrays: A comparative study. *Journal of Physics D: Applied Physics*, 49:1–7, 2016.

[36] Joachim Gräfe, Mathias Schmidt, Patrick Audehm, Gisela Schütz, Eberhard Goering, Joachim Gräfe, Mathias Schmidt, Patrick Audehm, and Gisela Schütz. Application

of magneto-optical Kerr effect to first-order reversal curve measurements. *Review of Scientific Instruments*, 85(023901):1–7, 2014.

[37] C R Pike, C A Ross, R T Scalettar, and G Zimanyi. First-order reversal curve diagram analysis of a perpendicular nickel nanopillar array. *Physical Review B*, 71(134407):1–12, 2005.

[38] Christopher R. Pike, Andrew P. Roberts, and Kenneth L. Verosub. Characterizing interactions in fine magnetic particle systems using first order reversal curves. *Journal of Applied Physics*, 85(9):6660–6667, 1999.

[39] Andrew Roberts, Christopher R. Pike, and Kenneth L. Verosub. First-order reversal curve diagrams : A new tool for characterizing the magnetic properties of natural samples. *Journal of geophysical Reseach Atmospheres*, 105(B12):28461–28475, 2000.

[40] Fanny Béron, Andreas Kaidatzis, Murilo F. Velo, Luis C.C. Arzuza, Ester M. Palmero, Rafael P. del Real, Dimitrios Niarchos, Kleber R. Pirota, and José Miguel García-Martín. Nanometer Scale Hard/Soft Bilayer Magnetic Antidots. *Nanoscale Research Letters*, 11(86):1–11, 2016.

[41] Felix Groß, Sven Erik Ilse, Gisela Schütz, Joachim Gräfe, and Eberhard Goering. nterpreting first-order reversal curves beyond the Preisach model : An experimental permalloy microarray investigation. *Physical Review B*, 99(064401):1–8, 2019.

[42] D A Allwood, X Gang, M D Cooke, and R P Cowburn. Magneto-optical Kerr effect analysis of magnetic nanostructures. *Journal of Physics D: Applied Physics*, 36(18):2175–2182, 2003.

[43] M. A. K. Budi, E .B. Glass, N.G. Rudawski, and J.S. Andrew. Exchange bias in bismuth ferrite / cobalt ferrite Janus nanofibers. *Journal of Materials Chemistry C*, 5:8586–8592, 2017.

[44] D. H. Kim and Chun Yeol You. Enhancement of magneto-optical Kerr effect signal from the nanostructure by employing anti-reflection coated substrate. *Journal of Magnetics*, 13(2):70–75, 2008.

[45] E. Nikulina, O. Idigoras, P. Vavassori, A. Chuvilin, and A. Berger. Magneto-optical magnetometry of individual 30 nm cobalt nanowires grown by electron beam induced deposition. *Applied Physics Letters*, 100(14), 2012.

[46] Wijn, van der Heide, and Fast. 'ORDERING IN COBALT-FERROUS FERRITES. *Convention on Ferrites*, pages 412–417, 1957.

[47] Hong-guo Zhang, Yu-jie Zhang, Weng-hong Wang, and Guang-heng Wu. Origin of the constricted hysteresis loop in cobalt ferrites revisited. *Journal of Magnetism and Magnetic Materials*, 323(15):1980–1984, 2011.

[48] T. George, A. T. Sunny, and T. Varghese. Magnetic properties of cobalt ferrite nanoparticles synthesized by sol-gel method. *IOP Conference Series: Materials Science and Engineering*, 73(1):1–5, 2015.

[49] K. Maaz, Arif Mumtaz, S. K. Hasanain, and Abdullah Ceylan. Synthesis and magnetic properties of cobalt ferrite (CoFe2O4) nanoparticles prepared by wet chemical route. *Journal of Magnetism and Magnetic Materials*, 308(2):289–295, 2007.

[50] Y. Suzuki, G. Hu, R. B. Van Dover, and R. J. Cava. Magnetic anisotropy of epitaxial cobalt ferrite thin films. *Journal of Magnetism and Magnetic Materials*, 191(1-2):1–8, 1999.

[51] Shi-shen Yan, R Schreiber, P Gru, and R Scha. Magnetization reversal in (0 0 1)Fe thin films studied by combining domain images and MOKE hysteresis loops. *Journal f Magnetism ans Magnetic Materials*, 210:309–315, 2000.

[52] R Varga, K Richter, A Zhukov, and V Larin. Domain Wall Propagation in Thin Magnetic Wires. 44(11):3925–3930, 2008.

[53] S Da Col, S Jamet, N Rougemaille, A Locatelli, T O Mentes, B Santos Burgos, R Afid, M Darques, L Cagnon, J C Toussaint, and O Fruchart. Observation of Bloch-point domain walls in cylindrical magnetic nanowires. *Physical Review B*, 180405:1–5, 2014.

[54] Z Q Qiu and S D Bader. Surface magneto-optic Kerr effect (SMOKE). *Journal of Magnetism and Magnetic Materials*, 200:664–678, 1999.

[55] M Grimsditch and P Vavassori. The diffracted magneto-optic Kerr effect : what does it tell you ? *Journal of Physics: Condensed Matter*, 16:R275–R294, 2004.

[56] Xiaobin Zhu, Zhigang Liu, Mark R Freeman, Vitali Metlushko, and I Introduction. Diffracted magneto-optical Kerr effects of permalloy ring arrays. *Journal of Applied Physics*, 97:1–2, 2005.

[57] T Schmitte, O Schw, S Goek, K Westerholt, and H Zabel. Magnetooptical Kerr effect of Fe-gratings. *Journal of Magnetism and Magnetic Materials*, 240:24–26, 2002.

[58] Fanny Béron, Louis-Philippe Carignan, David Ménard, and Arthur Yelon. Extracting Individual Properties from Global Behaviour: First-order Reversal Curve Method Applied to Magnetic Nanowire Arrays. In *Electrodeposited Nanowires and their Applications*, pages 168–188. 2010.

[59] Teladyne. *Lecroy waverunner Manual*. 2008.

[60] Andrew P. Roberts, David Heslop, Xiang Zhao, and Christopher R. Pike. Understanding fine magnetic particle systems through use of first-order reversal curve diagrams. *Reviews of Geophysics*, 52:557–602, 2014.

[61] Sirshendu Gayen, Milan K Sanyal, Biswarup Satpati, and Atikur Rahman. Diameter-dependent coercivity of cobalt nanowires. *Applied Physics A: Materials Science and Processing*, 112:775–781, 2013.

[62] Justin D. Starr, Maeve A K Budi, and Jennifer S. Andrew. Processing-property relationships in electrospun Janus-type biphasic ceramic nanofibers. *Journal of the American Ceramic Society*, 98(1):12–19, 2015.

[63] Fanny Béron, Liviu Clime, Mariana Ciureanu, David Ménard, Robert W Cochrane, and Arthur Yelon. Reversible and quasireversible information in first-order reversal curve diagrams. *Journal of Applied Physics*, 101(107):1–4, 2007.

[64] Fanny Béron, Luiz A S de Oliveira, Knobel Marcelo, and Kleber R Pirota. A novel method for identifying the local magnetic viscosity process of heterogeneous magnetic nanostructures A novel method for identifying the local magnetic viscosity process of heterogeneous. *Journal of Physics D: Applied Physics*, 46:1–10, 2013.

APPENDIX A

APPENDIX

A.1 GENERAL PROGRAMS

All functions which appear in the appendix have been referenced in the main body of the text and are written within the Igorpro v6.38B01 program. The programs are grouped in subsections based on how they were used, and with which other programs they were executed in conjunction.

A.1.1 KNIFE EDGE PROCESSING PROGRAM

This program is used in conjunction with a Labview process in order to extract waist and FWHM parameters of a focussed gaussian beam. Each file of raw data is fitted with an error function, and plotted. The fit function is shown below.

$$P(x) = A + P_{max} * (erf(\frac{\sqrt{2}(x - X_o)}{w})) \tag{A.1}$$

From the fit function, we extract the w parameter, which is the $\frac{1}{e^2}$ parameter, where the minimum is referred to as the beam waist. This is done for each different focal distance in order to build a beam profile to find the optimal focal length and minimum beam waist. From that we can extract the FWHM by multiplying our $\frac{1}{e^2}$ value by$\sim$1.1774.

```
#pragma rtGlobals=3          // Use modern global access method and
   strict wave access.
function KE_real()
variable i=0
setdatafolder root:Keraw
string KeRawAll=WaveList("!*step*",";","")
```

```
variable runs= Countobjectsdfr(root:KERAW:,1)
setdatafolder root:
 string Stepname,nameoferf
 make/n=(runs) WaistParameter
  make/n=(runs) FWHM
 make/n=(4)/o Coeffsb
 make/n=(runs)/t WidthName
 Wave coeffsb
 variable w
 Variable Rng1
Variable Rng2
Variable Rng3
do
        NameofErf = stringfromlist(i,KeRawAll)
        setdatafolder root:keraw
        wave TobeFit = \$Nameoferf
        setdatafolder root:
         Stepname = "steps_"+stringfromlist(i,KeRawAll)
         Wave stepwave = root:Stepwaves:\$(stepname)
         variable stepincrement = stepwave[2]-stepwave[1]
         wavestats/q tobefit

        CoeffsB[0]=v_max/100
        CoeffsB[1]=-v_max                                          //fit
            erf to loaded wave
        CoeffsB[2]=stepincrement*numpnts(tobefit)/2
        CoeffsB[3]=stepincrement*numpnts(tobefit)/4


        Funcfit/q Erf2,Coeffsb,Root:Keraw:\$(nameoferf)/x= Root:
           Stepwaves:\$(stepname) /D
        string fiterf = "fit_"+nameoferf
        w=CoeffsB[3]
        widthname[i]=nameoferf
        waistparameter[i]=Abs(w)
        FWHM[i] = 1.1774*Abs(w)
        rng1 = Floor(abs(enoise(65535)))
        rng2 = Floor(abs(enoise(65535)))
        rng3 = Floor(abs(enoise(65535)))
        If(i==0)
               Display :keraw:\$(nameoferf) vs :stepwaves:\$(stepname)
               appendtograph \$fiterf
               ModifyGraph rgb(\$fiterf)=(rng1,rng2,rng3)
               ModifyGraph rgb(\$nameoferf)=(rng1,rng2,rng3)
        else
               appendtograph :keraw:\$(nameoferf) vs :stepwaves:\$(
                  stepname)
               appendtograph \$fiterf
               ModifyGraph rgb(\$fiterf)=(rng1,rng2,rng3)
               ModifyGraph rgb(\$nameoferf)=(rng1,rng2,rng3)
        endif
        //Print w
```

```
        i+=1
while (i<(runs))
wave w_sigma
killwaves coeffsb,w_sigma
end
%
```

A.1.2 BALLISTIC SCATTERING FROM CYLINDER

As the section title indicates, this program simulates light scattering from a circular surface. In an attempt to do initial "back of the napkin" tests for the feasibility of MOKE in a scattering geometry, this program was created. Using the equation of a sphere and the law of reflection, this program will graph the reflected rays from an incoming "focused" plane wave. Additionally, there is an option to place a hypothetical lens to gather scattered rays which will cause reflected rays in the graph to appear a different color than rays missing the hypothetical lens. The diameter, focal length, and position of all hypothetical distances can be adjusted via prompts.

```
#pragma rtGlobals=3          // Use modern global access method and
   strict wave access.
Function BBeam_m()
variable Rmin,Rmax,Diam, FocalL,ThetaR,ThetaEye,ThetaEyedeg,x,y,r
variable numberofslices,FocalLMM,DiamMM
Variable Osize,Fsize,Distance,Distancea,thetaslices,DivChange
variable Dprime,Scalerbprime,currentangle,m,n, mmax,radius,LensAngle,
   variable lensangledeg,INSTANCE
Osize = 3000000.0 //nm
Fsize=23000.000 //nm
Distance = 75 //mm
R=500
FOCALLMM=75
DIAMMM=50
NUMBEROFSLICES=400
THETAEYEDEG=45
variable i=0
variable ycirc=0
variable p=0
string foldername

                    Prompt Osize, "pre-Focussed Beam size in nm"
                    prompt Fsize, "final size of beam at surface in
                       nm"
                    Prompt Distancea, "Focussing lens focal length(
                       mm)"
                    Prompt FolderName, "name of folder to move
                       things post simulation"
```

```
                    Prompt ThetaEyeDeg, "Incident angle in Degrees"
                    Prompt R, "Radius of cylinder in nm"
                    prompt FocalLMM, "distance to gathering lens in
                       mm"
                prompt lensangledeg, "Angular position of gathering
                   lens\r in degs" //@@@
                    Prompt DiamMM, "diameter of lens in mm"
                    Prompt Numberofslices,"How many divisions?"
                    Doprompt "Parameters of simulation, ~2.3 degree
                       beam diverence is built in.",osize,fsize,
                       distancea, Foldername,thetaeyedeg,R,focalLMM,
                       lensangledeg,DiamMM,Numberofslices
Make/o/t/n=(7) Settings


Settings[0]=num2str(Osize)+"nm"
settings[1]=num2str(fsize)+"nm"
settings[2]=num2str(distance)+"mm"
settings[3]=num2str(Thetaeyedeg)+" degrees"
settings[4]=num2str(R)+"nm"
settings[5]=num2str(focalLmm)+"mm"
settings[6]=num2str(diammm)+"mm"

lensangle = lensangledeg*Pi/180
print lensangle
thetaeye=thetaeyedeg*Pi/180
distance =Distance*1000000
diam = diamMM*1000000 //converts to nm from mm
focalL=FocalLMM*1000000
variable MaxDiv= atan((Osize-Fsize)/(2*distance))*180/pi
make/d/o/N=(numberofslices) RelaDiv
wave RelaDiv
make/d/o/N=(numberofslices) RelaDivdeg
wave RelaDivDeg
newdatafolder root:\$(foldername)

ThetaSlices= (Pi)/numberofslices
make/o/N=(numberofslices) Yvals
wave Yvals
make/o/N=(numberofslices) Yvalsdegs
wave Yvalsdegs
make/d/o/N=(numberofslices) Yvalsrads
wave Yvalsrads
make/d/o/N=(numberofslices) RelaDiv
wave RelaDiv
make/d/o/N=(numberofslices) RelaDivdeg
wave RelaDivDeg

Make/o ImagedArea
wave imagedarea

//Calculate a wave with all the Y points of the circular arc and the
   divergences as a function of angle around the circle
```

```
do
Currentangle= thetaslices*i
Ycirc = r*Sin(Currentangle)
        If(Currentangle<Thetaeye)
                M=r*sin(-(Currentangle)+ThetaEye)
                N=r*(cos(ThetaEye-Currentangle)+1)
        elseif (Currentangle > thetaeye)
                M=r*sin(Currentangle-ThetaEye)
                N=r*(cos(-ThetaEye+Currentangle)+1)
        elseif(Currentangle==thetaeye)
                M=0
                N=2*r
        endif
Mmax=Fsize+N*tan(Maxdiv)
Dprime=distance-N
ScalerBprime=M/(Fsize+N*tan(maxdiv))

        if(currentangle > thetaeye)
                Divchange=Atan(scalerbprime*(Osize-MMax)/(2*Dprime))
        else
                Divchange=-Atan(scalerbprime*(Osize-MMax)/(2*Dprime))
        endif
Reladiv[i] = Divchange
Reladivdeg[i] = Divchange*180/pi
Yvals[i]=ycirc
YvalsRads[i]=thetaslices*i
Yvalsdegs[i]=thetaslices*i*180/pi
i+=1
while(i<numberofslices)

Make/o/d/N=(numberofslices) ThetaEyePrime
wave ThetaEyePrime
ThetaEyePrime = thetaeye+reladiv //new ThetaEye that accounts for
   divergences


i=0
//Make new wave that is the angle that the beam would reflect from
make/o/N=(numberofslices) ThetaAR
wave ThetaAR

do
        Currentangle= thetaslices*i
        ThetaAR[i] = 2*currentangle-thetaeyeprime[i]
        i+=1
while (i < numberofslices)
i=0
//Finding the Slope/Y intercept of this junk
Make/o/N=(numberofslices) Bee
wave Bee

do
if(yvals[i]==R)
```

```
yvals[i]=yvals[i]-.0001*yvals[i] //makes sure r^2-y^2 isn't 0,
   yielding y intercept = inf
endif
i+=1
while(i<numberofslices)

bee=(yvals-tan(thetaAR))/Sqrt(r^2-Yvals^2)
i=0
variable w
variable h
If (lensangledeg >90)
        W = abs(FocalL*COS(lensangle))+abs((diam/2)*Sin(Lensangle))
       h = +abs(FocalL*sin(lensangle))+abs((diam/2)*cos(Lensangle))
else
        W = FocalL*COS(lensangle)+(diam/2)*Sin(Lensangle)
        h= FocalL*SIN(lensangle)+(diam/2)*Cos(Lensangle)
endif

VARIABLE w1=w
w=w*1.000001
variable h1= h
h=h*1.000001

do
       string dwink="Yline_"+num2str(i)
       Make/o/n=2 \$dwink
       wave YlineLine=\$dwink
       Ylineline[0]=R*sin(i*thetaslices)
       string Xtring="Xline_"+num2str(i)
       Make/o/n=2 \$Xtring
       wave XlineLine=\$Xtring
       Xlineline[0]=R*Cos(i*thetaslices)

If (thetaAR[i]>(Pi+ThetaEyePrime[i]))
 Ylineline[1]=ylineline[0]
 xlineline[1]=xlineline[0]

ELSE

If(thetaAR[i]> -Pi/2 && thetaAR[i]<=0) //the if statements include
   desired x/y components, so generalizeing them may be the solution
   to "moving" the gathering lens
       //print thetaAR[i]*(W1)+bee[i]
       if(tan(thetaAR[i])*(W1)+bee[i]>H1 )

                Ylineline[1]=H1
              Xlineline[1]= (h1-Bee[i])/tan(thetaAR[i])
       elseif (tan(thetaAR[i])*(W1)+bee[i]<=H1)
             Ylineline[1]=tan(thetaAR[i])*(W1)+bee[i]
             Xlineline[1]=(w1)
       endif
elseIf(thetaAR[i]> 0 && thetaAR[i]<=Pi/2)
//print thetaAR[i]*(W1)+bee[i]
```

```
        if(tan(thetaAR[i])*(W1)+bee[i]> H1)
                Ylineline[1]=H1
               Xlineline[1]= (H1-Bee[i])/tan(thetaAR[i])        //
                  FInish double check then check W, H assignments for
                  different quadrents
               INSTANCE= 1
        elseif(tan(thetaAR[i])*(W1)+bee[i]<=H1)
              Ylineline[1]=tan(thetaAR[i])*(W1)+bee[i]
              Xlineline[1]=(W1)
              instance=2
        endif
elseIf(thetaAR[i]> Pi/2 && thetaAR[i]<=Pi)
        if(tan(thetaAR[i])*(-W1)+bee[i]> H1)
                Ylineline[1]=H1
               Xlineline[1]= (h1-Bee[i])/tan(thetaAR[i])
               Instance=1
        elseif(tan(thetaAR[i])*(-w1)+bee[i]<=h1)
              Ylineline[1]=tan(thetaAR[i])*(-w1)+bee[i]
              Xlineline[1]=(-w1)
              Instance=3
        endif
elseIf(thetaAR[i]> Pi && thetaAR[i]<=3*Pi/2)

        if(tan(thetaAR[i])*(-w1)+bee[i]> h1)
                Ylineline[1]=h1
               Xlineline[1]= (h1-Bee[i])/tan(thetaAR[i])
               Instance=1
        elseif(tan(thetaAR[i])*(-w1)+bee[i]<=h1)
              Ylineline[1]=tan(thetaAR[i])*(-w1)+bee[i]
              Xlineline[1]=(-w1)
              Instance=3
        endif
ENDIF
endif
If (i==0)
               Display/W=(0,0,1200,700)/n=\$(FOLDERNAME) Ylineline vs
                  Xlineline
               Doupdate

else



               variable Xwidth
               xwidth= w1-Diam*sin(lensangle)

        //THese if statements are wacky and need work....
               //Print i
               //variable eWhigh= -w
               // variable ewlow=(-w+Diam*sin(lensangle)
               //variable HHigh= H
               //variable hlow = (H-Diam*Abs(cos(lensangle))
               //variable Ecks = xlineline[1]
```

```
                //variable ewhy = ylineline[1]
                //Print ewhigh, ewlow, hhigh, hlow,ecks, ewhy

        if(lensangledeg>90)
                if(xlineline[1]>=(-w) && Xlineline[1]<=-w1+Diam*sin(
                   lensangle) && Ylineline[1]>=H1-Diam*abs(cos(
                   lensangle)) &&abs(Ylineline[1])<=abs(H))
                    if (p>=127)
                            insertpoints p+1, 1, imagedarea
                    endif
                            appendtoGraph/c=(0,0,65000)/w=\$(
                               FOLDERNAME) Ylineline vs Xlineline
                            imagedarea[p] =thetaslices*i
                            p+=1
                elseif (Ylineline[1]!=0)
                            appendtoGraph/w=\$(FOLDERNAME) Ylineline
                               vs Xlineline
                endif
        else
                if(abs(xlineline[1])<=abs(w)&& Xlineline[1]>=w1-Diam*
                   sin(lensangle) && Ylineline[1]>=H1-Diam*abs(cos(
                   lensangle)) &&abs(Ylineline[1])<=abs(H))
                    if (p>=127)
                            insertpoints p+1, 1, imagedarea
                    endif

                    appendtoGraph/c=(0,0,65000)/w=\$(FOLDERNAME)
                       Ylineline vs Xlineline
                    imagedarea[p] =thetaslices*i
                    p+=1
                elseif (Ylineline[1]!=0)
                    appendtoGraph/w=\$(FOLDERNAME) Ylineline vs
                       Xlineline
                endif
        endif
        Doupdate
endif
i+=1
while(i<numberofslices)

imagedarea =imagedarea==0?NaN : imagedarea
wavetransform zapnans imagedarea
variable length = numpnts(imagedarea)


variable arclength = imagedarea[length-1]*R-(imagedarea[0]*R)
string arclengthWord=num2str(arclength) //Blue arclength

variable deltatheta = (180/pi)*(imagedarea[length-1]-imagedarea[0])
string DELTATHETAWORD = num2str(deltatheta) //Blue angles
variable theta1=imagedarea[0]*180/Pi
variable theta2=imagedarea[length-1]*180/pi
string theta1word = num2str(theta1)
```

```
string theta2word=num2str(theta2)

TextBox/C/N=text0/S=3/A=MC "Arclength that was imaged is "+
   arclengthword+"nm\rAngle starts at "+theta1word+" deg and ends at
   "+theta2word+" degs\rTotal angle angle imaged is "+deltathetaword
   +"deg"+"\rDiameter, distance, and angular position of lens is \r"+
    num2str(diammm)+"mm, "+num2str(focallmm)+"mm, and "+ num2str(
   lensangledeg)+ "degrees respectively."

String ListofEverything = wavelist("*",";","")
variable L=0
variable Items= itemsinlist(listofeverything)
do
string mover =stringfromlist(l,listofeverything)
Movewave \$mover, root:\$(foldername):
L+=1
while(L<items)
SetAxis left -R-((.25*R)),R+(.25*R)
SetAxis bottom -R-((.25*R)),R+(.25*R)

Print w1,w,h1,h
print (w1-Diam*sin(lensangle))
print (h1-Diam*cos(lensangle))

end
%
```

A.1.3 Magnetic Rise Time Fitting

When investigating the response from the GMW 3470 electromagnet, I found that the response was relatively slow and would result in a slowly deviating field value when compared to a calculated value from a lookup table. To parametrize the functional response of the electromagnet, I created this program which allows a user to select a beginning and endpoint of an exponential-like magnetic field response by placing cursors on a graph, and then choose one of three functions to attempt to fit the subset of data. Based on trial and error, the initial guesses for the fit functions are drawn from the data itself. The unique aspect of this program allows the user to store the location of the cursor placements in waves which is useful if you are fitting many successive magnetic responses and you need to exit the program. fit data is saved as literal fir graphs for exact reproduction of the fit functions.

```
#pragma rtGlobals=3          // Use modern global access method and
   strict wave access.
```

```
Function MagneticFit()
string thesteplocation, Thetimelocation, fittype

//(X)Locate the large wave of magnetic steps and locate the x axis (
   time) also
//(X)Prompt user with a textbox that asks for the user to select a
   start and end location with cursors. (Use Pauseforuser
//(X)ask user to pick a fit function to use.
//(X)Ask for initial guesses (this is probably better done outside of
   the program first since I am using iuser defined fit functions.
//(X)some of the parameters can be acquired from the cursors' location
//(X)Fit the function and save the fit parameters of note. Not sure
   which are important yet though :\ (use pcsr() which saves location
    of cursors)
//(X) Repeat in loop, propmting user if they want to do it again at
   the end of each loop.
//(X)Use fit parameters from previous run to initialize guess for the
   next run.
//(X)use showinfo/hideinfo to make the cufrsors appear.


       string cdfBefore = GetDataFolder(1) // Save step data folder
          before.
Execute "CreateBrowser prompt=\"Locate The Steps In Current/Magnetic
   Field\", showWaves=1, showVars=0, showStrs=0" //asks you to find a
    wave, who'se path is stored as S_browserlist
       string cdfAfter = GetDataFolder(1)  // Save current data
          folder after, though seeems like this is the same as
          CDFbefore since execute doesn't set the folder unless you
          do that, just saves path
       SetDataFolder cdfBefore                                  // Restore
          you location to original location (presumably root).
       SVAR S_BrowserList=S_BrowserList //no clue why this is here
       variable pathlength =strlen(s_browserlist) //length, in
          numbers, of the path name.
       TheStepLocation= s_browserlist[0,pathlength-2] //cuts the
          semicolon from the path name


       cdfBefore = GetDataFolder(1) // Save time data folder before.
       Execute "CreateBrowser prompt=\"Locate The X-axis for the
          steps. Usually this is Time\", showWaves=1, showVars=0,
          showStrs=0" //asks you to find a wave, who'se path is
          stored as S_browserlist
       cdfAfter = GetDataFolder(1)  // Save current data folder after,
           though seeems like this is the same as CDFbefore since
          execute doesn't set the folder unless you do that, just
          saves path
       SetDataFolder cdfBefore                                  // Restore
          you location to original location (presumably root).
       SVAR S_BrowserList=S_BrowserList //no clue why this is here
```

```
        variable pathlength2 =strlen(s_browserlist) //length, in
           numbers, of the path name.
        TheTimeLocation= s_browserlist[0,pathlength2-2] //cuts the
           semicolon from the path name

        wave Timewave =$thetimelocation
        wave Stepswave = $thesteplocation
variable v_chisq
Variable p=1
Variable i=0
variable/G ender=1
Variable/g Oldies =0 //We assume that we are starting from scratch,
   previous cursors used
Variable Oldrows
Make/N=1/o TauA
Make/N=1/o TauB
Make/o/N=1 StepSize
Make/o/N=1 AplusB
Make/o/N=1 Aye
Make/o/N=1 Bee


string listoffittypes="ExpAndExp;Exp;ExpAndLog;SingleExpFinder"
Prompt Fittype, " which type of Fit Function do you want?",popup,
   listoffittypes
Doprompt "Select the fit that you want to use",fittype

NewPanel/Flt /K=1 /W=(500,368,800,531) as "Have you Done this before?"
DoWindow/C dejavoo
DrawText 21,20,"If you have, the cursors locations are saved"
DrawText 21,40,"Select Yes to used old cursors"          //All of
   this is to see if you have old cursors stored that we can use!
DrawText 21,60,"Select No to start from scratch"
Button button2,pos={80,75},size={92,20},title="Yes"
Button button2,proc=OldCursors
Button button3,pos={80,105},size={92,20}
Button button3,proc=NoOldies,title="No"
pauseforuser dejavoo

if(oldies==1) //if we have an old cursor file This is variable from
   the buttonpress

         cdfBefore = GetDataFolder(1) // Save step data folder before.
        Execute "CreateBrowser prompt=\"Locate Cursor A's point wave
           \", showWaves=1, showVars=0, showStrs=0" //asks you to find
            a wave, who'se path is stored as S_browserlist
        cdfAfter = GetDataFolder(1)  // Save current data folder after,
            though seeems like this is the same as CDFbefore since
           execute doesn't set the folder unless you do that, just
           saves path
        SetDataFolder cdfBefore                          // Restore
           you location to original location (presumably root).
        SVAR S_BrowserList=S_BrowserList //no clue why this is here
```

```
        variable pathlength3 =strlen(s_browserlist) //length, in
           numbers, of the path name.
        string CursorALocation= s_browserlist[0,pathlength3-2] //cuts
           the semicolon from the path name


        cdfBefore = GetDataFolder(1) // Save time data folder before.
        Execute "CreateBrowser prompt=\"Locate B's Cursor wave\",
           showWaves=1, showVars=0, showStrs=0" //asks you to find a
           wave, who'se path is stored as S_browserlist
        cdfAfter = GetDataFolder(1)  // Save current data folder after,
            though seeems like this is the same as CDFbefore since
           execute doesn't set the folder unless you do that, just
           saves path
        SetDataFolder cdfBefore                                  // Restore
           you location to original location (presumably root).
        SVAR S_BrowserList=S_BrowserList //no clue why this is here
        variable pathlength4 =strlen(s_browserlist) //length, in
           numbers, of the path name.
        string CursorBLocation= s_browserlist[0,pathlength4-2] //cuts
           the semicolon from the path name

        wave MemoryCursorA= $CursorAlocation
        wave MemoryCursorB= $CursorBlocation

         OldRows =numpnts(MemorycursorA)

 else //make a new set of meory files with a "unique" name by
    appending _0,1,2,3 by seeing if one exists with the same name
        string MemCursrA_="StoredCursorA_"
        string MemCursrB_="StoredCursorB_"
        string uniquenameforcursorA=uniquename( MemCursrA_,1,0)
        string uniquenameforcursorB=uniquename( MemCursrB_,1,0)
        Make/o/n=1 $uniquenameforcursorA
        Make/o/n=1 $uniquenameforcursorB
        wave MemoryCursorB=$uniquenameforcursorB
        wave memoryCursorA=$uniquenameforcursorA
        Oldrows=0
endif


string ListofGraphNames=Winlist("Thegraph",";","")

        If(strlen(stringfromlist(0,ListofGraphnames))!=0) //check if
           the graph window is already there so we don't end up with
           100 of them wen troubleshooting
               killwindow Thegraph
        endif

Display/w=(300,100,1600,1000)/N=Thegraph StepsWave vs Timewave

Do
```

```
Dowindow/f Thegraph
showinfo/cp=0/w=thegraph


 if(i >= Oldrows||Oldrows==0) //if we are beyond the cursors that have
     been stored previously, start prompting again

       if (oldies==1&& i==oldrows)
              print "you did your old cursors if you had any, you are
                  on step " + num2str(i)
              Print "the last cursor set down (B) was at " + num2str(
                 MemorycursorB[oldrows-1])
       endif

       NewPanel /K=1 /W=(187,368,437,531) as "Pause for Cursor"
       DoWindow/C tmp_PauseforCursor                                  //
          Set to an unlikely name
       AutoPositionWindow/E/M=1/R=thegraph                // Put panel
           near the graph
       DrawText 21,20,"Adjust the cursors and then"
       DrawText 21,40,"Click Continue."
       Button button0,pos={80,58},size={92,20},title="Continue"
       Button button0,proc=CursorControl
       Button button1,pos={80,90},size={92,20}
       Button button1,proc=Endthething,title="End"
       dowindow/f thegraph
       PauseForuser tmp_PauseforCursor,Thegraph

endif

If(ender==1)//if ender is 1, continue, ender is 0, END THE PROGRAM

       If(stringmatch(fittype,"ExpAndLog")==1)//!!!!DOES NOT CONTAIN
          FULL PROGRAMMING AS EXP PLUS EXP DOES!!!!!
              Make/d/o/n=6 w_coef
              W_coef[0]=vcsr(b,"thegraph") //shift
              W_coef[1]=vcsr(b,"thegraph")-vcsr(a,"thegraph") //A
              W_coef[2]=abs(hcsr(a,"thegraph")-hcsr(a,"thegraph")/10)
                 //toff must not be able to add to t to =0, that
                 gives an infinity
              W_coef[3]=-(vcsr(b,"thegraph")-vcsr(a,"thegraph"))/10//
                 B
              W_coef[4]=(hcsr(b,"thegraph")-hcsr(a,"thegraph"))/5//
                 logtau
              W_coef[5]=(hcsr(b,"thegraph")-hcsr(a,"thegraph"))/100//
                 expTau

              FuncFit/n/NTHR=0 LogplusExp W_coef stepswave[pcsr(a,"
                 thegraph"),pcsr(b,"thegraph")] /X=timewave[pcsr(a,"
                 thegraph"),pcsr(b,"thegraph")] /D
              ModifyGraph rgb(wave1)=(3,52428,1)
              TauA[i]= {W_coef[4]}
              TauB[i]= {W_coef[5]}
```

```
        //!!!!DOES NOT CONTAIN FULL PROGRAMMING AS EXP PLUS EXP
            DOES!!!!!
        i+=1

elseif(stringmatch(fittype,"ExpAndExp")==1)
        Make/d/o/n=6 w_coef

        if(i==0&&oldrows==0)// AND oldies=0//do the thing here,
            otherewise, need to look at the old cursors

                W_coef[0]=vcsr(b,"thegraph") //y0
                W_coef[1]=vcsr(a,"thegraph")-vcsr(b,"thegraph")
                    //A
                W_coef[2]=hcsr(a,"thegraph")//toff must not be
                    able to add to t to =0, that gives an
                    infinity
                W_coef[3]=(hcsr(b,"thegraph")-hcsr(a,"thegraph")
                    )/40///taua
                W_coef[4]=(hcsr(b,"thegraph")-hcsr(a,"thegraph")
                    )/5 //TauB
                W_coef[5]=(vcsr(a,"thegraph")-vcsr(b,"thegraph")
                    )/10//B
                MemoryCursorA[i]={Pcsr(A,"Thegraph")}
                MemoryCursorB[i]={Pcsr(B,"Thegraph")}

                FuncFit/L=800/n/NTHR=0 ExpPlusExp W_coef
                    stepswave[pcsr(a,"thegraph"),pcsr(b,"thegraph
                    ")] /X=timewave[pcsr(a,"thegraph"),pcsr(b,"
                    thegraph")] /D

        elseif(i>0 && i>=oldrows) //and if i>
            numpntsmemorycursorAVariable (this CANNOT be a
            moving value, must be counted when we find it)//do
            the thing here, otherewise, need to look at the old
            cursors
                W_coef[0]=vcsr(b,"thegraph") // only need to
                    adjust these two fit parameters, the rest
                    come from the
                W_coef[2]=hcsr(a,"thegraph")
                W_coef[1]=vcsr(a,"thegraph")-vcsr(b,"thegraph")
                    //A
                W_coef[3]=(hcsr(b,"thegraph")-hcsr(a,"thegraph")
                    )/40///taua
                W_coef[4]=(hcsr(b,"thegraph")-hcsr(a,"thegraph")
                    )/5 //TauB
                MemoryCursorA[i]={Pcsr(A,"Thegraph")}
                MemoryCursorB[i]={Pcsr(B,"Thegraph")}

                FuncFit/L=800/n/NTHR=0 ExpPlusExp W_coef
                    stepswave[pcsr(a,"thegraph"),pcsr(b,"thegraph
                    ")] /X=timewave[pcsr(a,"thegraph"),pcsr(b,"
                    thegraph")] /D
```

```
elseif(i<oldrows && oldies==1&&i==0) //if it is an old
   cursor list, AND we are within the list AND if it is
    the first point, give guesses for all W_coef,
   Otherwise we don't need to give all guesses.
       W_coef[0]=stepswave[memorycursorb[i]] //y0
       W_coef[1]=stepswave[memorycursora[i]]-stepswave[
          memorycursorb[i]] //A
       W_coef[2]=TimeWave[memorycursora[i]]//toff must
          not be able to add to t to =0, that gives an
          infinity
       W_coef[3]=(TimeWave[memorycursorB[i]]-TimeWave[
          memorycursora[i]])/40///taua
       W_coef[4]=(TimeWave[memorycursorB[i]]-TimeWave[
          memorycursora[i]])/5 //TauB
       W_coef[5]=(stepswave[memorycursora[i]] -
          stepswave[memorycursorb[i]] )/10//B
       STEPSIZE[i] = Stepswave[MemorycursorB[i]]-
          Stepswave[MemorycursorA[i]]
       Setaxis left stepswave[memorycursora[i]]-.01,
          stepswave[memorycursorb[i]]+.01
       SetAxis bottom TimeWave[memorycursorA[i]]-.35,
          TimeWave[memorycursorB[i]]+.35
       Textbox/c/N=Text0/s=3/A=MC "i= "+num2str(i)
       FuncFit/Q/L=800/n/NTHR=0 ExpPlusExp W_coef
          stepswave[memorycursora[i],Memorycursorb[i]]
          /X=timewave[memorycursora[i],Memorycursorb[i
          ]] /D

elseif(i<oldrows && oldies==1&&i!=0)
       W_coef[2]=TimeWave[memorycursora[i]]//toff must
          not be able to add to t to =0, that gives an
          infinity
       W_coef[0]=stepswave[memorycursorb[i]]
       W_coef[1]=stepswave[memorycursora[i]]-stepswave[
          memorycursorb[i]] //A
       W_coef[3]=(TimeWave[memorycursorB[i]]-TimeWave[
          memorycursora[i]])/40///taua
       W_coef[4]=(TimeWave[memorycursorB[i]]-TimeWave[
          memorycursora[i]])/5 //TauB
       W_coef[5]=(stepswave[memorycursora[i]] -
          stepswave[memorycursorb[i]] )/10//B
       Setaxis left stepswave[memorycursora[i]]-.01,
          stepswave[memorycursorb[i]]+.01
       SetAxis bottom TimeWave[memorycursorA[i]]-.35,
          TimeWave[memorycursorB[i]]+.35
       Textbox/c/N=Text0 "i= "+num2str(i)
       FuncFit/Q/L=800/n/NTHR=0 ExpPlusExp W_coef
          stepswave[memorycursora[i],Memorycursorb[i]]
          /X=timewave[memorycursora[i],Memorycursorb[i
          ]] /D
```

```
            STEPSIZE[i] = Stepswave[MemorycursorB[i]]-
               Stepswave[MemorycursorA[i]]
        endif

        //FuncFit/L=400/n/NTHR=0 ExpPlusExp W_coef stepswave[
           pcsr(a,"thegraph"),pcsr(b,"thegraph")] /X=timewave[
           pcsr(a,"thegraph"),pcsr(b,"thegraph")] /D
        doupdate/w=thegraph
        string nameofFit="Fit_"+(nameofwave(stepswave))
        wave fitwave =$nameoffit
        ModifyGraph rgb($nameoffit)=(3,5000,5300)
        doupdate/w=thegraph
        TauA[i]= {W_coef[3]}
        TauB[i]= {W_coef[4]}
         Aye[i]= {W_coef[1]}
        Bee[i]= {W_coef[5]}

        string newfitname= NameofFit+"_"+num2str(i)

        Duplicate/o fitwave, $newfitname
        //StepSize[i]={vcsr(b,"thegraph")-vcsr(a,"thegraph")}
        AplusB[i]={W_coef[1]+W_Coef[5]}//gonna try A+B on this
           one, cause they look like they sum to be the exact
           step size
        i+=1


elseif(stringmatch(fittype,"Exp")==1)//!!!!DOES NOT CONTAIN
   FULL PROGRAMMING AS EXP PLUS EXP DOES!!!!!

        if (oldies==1&& i<oldrows)
               curveFit/q/L=800/n/NTHR=0 Exp_Xoffset,kwcWave=
                  w_coef, stepswave[memorycursora[i],
                  Memorycursorb[i]] /X=timewave[memorycursora[i
                  ],Memorycursorb[i]]/D

               Setaxis left stepswave[memorycursora[i]]-.01,
                  stepswave[memorycursorb[i]]+.01
               SetAxis bottom TimeWave[memorycursorA[i]]-.35,
                  TimeWave[memorycursorB[i]]+.35
               Textbox/c/N=Text0 "i= "+num2str(i)
        elseif(oldies==1 && i>=oldrows)
               curveFit/Q/L=800/n/NTHR=0 Exp_Xoffset,kwcWave=
                  w_coef, stepswave[pcsr(a,"thegraph"),pcsr(a,"
                  thegraph")+150] /X=timewave[pcsr(a,"thegraph
                  "),pcsr(a,"thegraph")+150]/D
               MemoryCursorA[i]={Pcsr(A,"Thegraph")}
               MemoryCursorB[i]={Pcsr(A,"Thegraph")+150}
        elseif(oldies!=1)
               curveFit/Q/L=800/n/NTHR=0 Exp_Xoffset,kwcWave=
                  w_coef, stepswave[pcsr(a,"thegraph"),pcsr(a,"
```

```
                    thegraph")+150] /X=timewave[pcsr(a,"thegraph
                    "),pcsr(a,"thegraph")+150]/D
                MemoryCursorA[i]={Pcsr(A,"Thegraph")}
                MemoryCursorB[i]={Pcsr(A,"Thegraph")+150}
        endif

        doupdate/w=thegraph
        nameofFit="Fit_"+(nameofwave(stepswave))
        wave fitwave =$nameoffit
        ModifyGraph rgb($nameoffit)=(3,5000,5300)
        doupdate/w=thegraph
        Aye[i]={w_coef[1]}
        TauA[i]= {W_coef[2]}
                i+=1
Elseif(stringmatch(fittype,"SingleExpFinder")==1)
        String chisname="AllmyChi_"+num2str(i)
        Make/o/n=1 $chisname
        wave Chiwave=$chisname

        String chisNameTime="ChiTimes_"+num2str(i)
        Make/o/n=1 $chisNameTime
        wave ChiTimeWave=$chisNameTime

        variable R=0
        p=2

        if(oldies==1)

                do
                        curveFit/Q/L=800/n/NTHR=0 Exp_Xoffset,
                           kwcWave=w_coef, stepswave[
                           memorycursora[i],memorycursora[i]+200+
                           p] /X=timewave[memorycursora[i],
                           memorycursora[i]+200+p]/D
                        chiwave[R]={v_chisq/(p+200)}
                        //Print v_chisq
                        //print chiwave[r]
                        Chitimewave[R]={Timewave[memorycursora[i
                           ]+200+p]-Timewave[memorycursora[i]]}

                        p+=10
                        R+=1

                while((200+p)<=((memorycursorb[i])-memorycursora
                   [i]))

        elseif(oldies==0)

                do
                        curveFit/Q/L=800/n/NTHR=0 Exp_Xoffset,
                           kwcWave=w_coef, stepswave[pcsr(a,"
                           thegraph"),pcsr(a,"thegraph")+200+p] /
```

```
                                    X=timewave[pcsr(a,"thegraph"),pcsr(a,"
                                    thegraph")+200+p]/D
                                Chiwave[R]={v_chisq/(p+201)}
                                Chitimewave[R]={Timewave[pcsr(a,"thegraph
                                    ")+200+p]-Timewave[pcsr(a,"thegraph")
                                    ]}
                                p+=10
                                R+=1

                            while(p<=(pcsr(b,"thegraph")-pcsr(a,"thegraph"))
                               )


                    endif
            Print nameofwave(chitimewave)
            i+=1
            print i
            endif

endif //endif for ender, otherwise it skips the whole fitting part and
    goes straight to end

While(Ender!=0)

//killwindow thegraph
end


Function CursorControl(CtrlName) : ButtonControl
string ctrlName
variable/G Ender=1
killwindow tmp_pauseforcursor


End

Function Endthething(CtrlName) : ButtonControl
string CtrlName

variable/G Ender=0  //End the program, presumably if you have no more
   steps to take. Variable is glabal so it can be shared between
   programs
killwindow tmp_pauseforcursor
End

Function NoOldies(CtrlName) : ButtonControl
string CtrlName

variable/G oldies=0
killwindow Dejavoo
End

Function OldCursors(CtrlName) : ButtonControl
string CtrlName
```

```
variable/G oldies=1
killwindow Dejavoo
End
//using wave[i]={variabe} will addit to the end even if there are not
   enough indicies.
//Uniquename is a function that avoids naming issues! :D
//Log all of the cursor positions so that multiple fits can be run off
    of the same graph once it has been used once?
//On the note above, I can store the Cursor positions , or just the
   Whole W_COef as a function of i, so we get Wcoef1,wcoef2,wcoef3..
   etf.
```

A.2 MOKE Specific Programs

A.2.1 Bulk Normalizing

This program is used to normalize large batches of MOKE data. By looking at the average min and average max of each MOKE run, the data is normalized and centered about y=0 and the min and max values are scaled to ± 1. Unfortunately this program requires all files to be in certain folders to work properly. It is best used in conjunction with the program in

```
Function MassSaS()
        string source,folderofinterest,suffix
        variable i=0
        variable c=0
        string yes

        Prompt FolderofInterest,"root:subfolder1:subfolder2: ... :
           subfolderN:"
        doPrompt "full path to waves",Folderofinterest
        Variable Counter
        print folderofinterest


        //if(stringmatch (folderofinterest, "root"))
        //      Counter = CountobjectsDFR(root:,1)
        //else
                Counter = CountobjectsDFR(\$(folderofinterest),1)
      // endif
       print folderofinterest
       Prompt Suffix, "what will be the suffix of rotation wave. eg.
          Rotation_'...'"
       doprompt "naming", suffix

       string fullRotationname= "Rotation_"+suffix
        make/o/n=(counter) \$fullrotationname
        wave rotation =\$fullrotationname
```

```
        string nameofrotators = "Rotators_"+suffix
        make/o/n=(counter)/t \$nameofrotators
        wave/t rotators=\$nameofrotators

        NewDataFolder/O root:NormalizedWaves
        Newdatafolder/O root:OriginalWaves
        setdatafolder root:Normalizedwaves
        killwaves/a
        setdatafolder root:

        Do
               String Namedatwave

               namedatwave=Getindexedobjnamedfr(\$(folderofinterest),1,
                  i)
               setdatafolder \$(folderofinterest)

                wave uggwave =\$namedatwave
                //print namedatwave
                Variable n
                Variable Totalpoints =numpnts(uggwave)

                n =floor(numpnts(uggwave)/4)
                String destination // name of destination wave
                Variable segment, numSegments
                Variable startX, endX, lastX
                destination= NameOfWave(uggwave)+"_m" // derive name of
                    dest from source
                numSegments = trunc(numpnts(uggwave) / n)
                if (numSegments < 1)
                       DoAlert 0, "Destination must have at least one
                          point"

                endif

if(numsegments>0)

                       Make/O/N=(numSegments) \$destination
                       WAVE destinationw = \$destination
                       lastX = pnt2x(uggwave, numpnts(uggwave)-1)

                for (segment = 0; segment < numSegments; segment += 1)

                       startX = pnt2x(uggwave, segment*n) // start X
                          for segment
                       endX = pnt2x(uggwave, (segment+1)*n - 1)// end X
                           for segment
                       // this handles case where numpnts(source)/n is
                          not an integer
                        endX = min(endX, lastX)
                       destinationw[segment] = mean(uggwave, startX,
                          endX)
```

```
        endfor


        String dest
        variable size
        variable Offset
        Variable scale
        Variable MinAvg
        Variable MaxAvg //Change the min / max finder to query
           the average values instead of the min/max
        make/o M_wavestats
        wave M_wavestats
        make/o/n=(floor(totalpoints/25)) first1
        make/o/n=(floor(totalpoints/25)) last1
        wave Firstwave = first1
        wave lastwave = last1

        duplicate/o/R=[0,FLOOR(totalpoints*.04)-1] uggwave,
           firstwave
        duplicate/o/R=[totalpoints-1-FLOOR(totalpoints*.04),
           totalpoints-1] uggwave, lastwave

        concatenate/o {firstwave,lastwave}, FLwave
        wavestats/w/Q FLwave
        MinAvg = M_wavestats(3)

        wavestats/w/q/R=(floor((totalpoints/2)-0.04*totalpoints
           ),floor((totalpoints/2)+0.04*totalpoints)) uggwave
        MaxAvg = M_wavestats(3)
         KILLWAVES FIRST1, LAST1, FLWAVE, M_wavestats

        rotators[i] = nameofwave(uggwave)
        Rotation[i]= Maxavg-minavg
        Offset=(MinAvg+MaxAvg)/2
        Scale=(MaxAvg-MinAvg)/2
        dest ="N_"+ NameOfWave(Uggwave) // derive name of dest
           from source
size=numpnts(uggwave)//Match numer of points in new wave = to
   the number of points in the source wave

        Make/O/N =(size) \$dest //make a new wave to fill the
           space reserved for dest
        wave destw= \$dest //assign a name destw to the wave
           derived from dest
        destw=Uggwave-Offset//fill slots of destw with this
           operation
        destw/=scale

        Killwaves destinationw
        Print Nameofwave(uggwave)+" is now normalized!"
        Movewave destw,root:normalizedwaves: //puts the
           normalized wave in a seperate folder
```

```
endif
                i+=1
                //print i

     while(i<=(counter-1))
     killwaves m_wavestats
//   do
  // string movingwave =getindexedobjname("",1,0)
 //  wave movingwave=\$movingstring
 //  movewave movingwave:
 //  while



 setdatafolder \$(folderofinterest)//root:


 String ListofEverything = wavelist("*",";","") //moves all the
    original waves but leaves the rotatio nwave
variable L=0
variable Items= itemsinlist(listofeverything)
do
string mover =stringfromlist(l,listofeverything)
if( stringmatch(mover,"!rotat*")==1)
Movewave \$mover, root:originalwaves:
endif
L+=1
while(L<(items))
setdatafolder root:

End
```

A.2.2 Wave Averages

This program uses the filename prefixes to match like-runs and then averages them and sorts them into separate folders. Useful to parse large numbers of individual runs to singular low-noise average runs. The beginning of the program includes some notes specific to usage.

```
#pragma rtGlobals=3          // Use modern global access method and
   strict wave access.
function WaveAverages()

//This program will hopefully avearge waves with the same prefix name,
    aka if you have run_0001, run_0002, run_0003 ,runb_0001,runb_0002
   , test
```

```
//the program will average the Run prefix, the runb prefix and leave
   the test alone since it doesn't have a suffix.
//this will help, ideally, speed up averaging if i have multiple few-
   run trials, aka 50, 20-run averages that would take a while to use
    the waves average panel to do averaging for.
//what do i need then? Since my files always end up in 2 folders ,
   Normalized waves and Original waves from the MassSas() program (
   run that first), I should be able to just ask
// for normalized or original waves....or just do that by default All
   i need is the path to get to the folder because maybe i had pre
   set them ahead of time.

Variable i=0
Variable p=0
String nameoffolder
string thefolder
//just like the coercivity() program, i will ask you to find the
   folder that you want to pull waves from.
String cdfBefore = GetDataFolder(1) // Save current data folder before
   .
Execute "CreateBrowser prompt=\"Find the folder with MOKE data\",
   showWaves=0, showVars=0, showStrs=0" //asks you to find a data
   folder, who'se path is stored as S_browserlist
String cdfAfter = GetDataFolder(1)  // Save current data folder afte,
   though seeems like this is the same as CDFbefore since execute
   doesn't set the folder unless you do that, just saves path
SetDataFolder cdfBefore                              // Restore current
   data folder.
SVAR S_BrowserList=S_BrowserList //no clue why this is here
                                                        All of this just
    locates the right folder full of waves.
variable pathlength1 =strlen(s_browserlist) //length, in numbers, of
   the path name..
Thefolder=s_browserlist[0,pathlength1-2] //cuts the semicolon from the
    path name


setdatafolder \$(thefolder)
variable numberofFolders =countobjectsdfr(\$(thefolder),4)

       do
              string AreyouSingle =getindexedobjnamedfr(\$(thefolder)
                 ,4,p)

              if(stringmatch(areyousingle,"singles")==0&&(p==
                 numberoffolders-1)||numberoffolders==0)
                     newdatafolder singles
                     p=numberoffolders
                                                        //Looks to see
                        if there is already a folder named "singles"
                        in which to put single runs, if so, it puts
                        them in there, if not, it makes one.
              elseif (stringmatch(areyousingle,"singles")==1)
```

```
                p+=numberoffolders
        else
                p+=1
        endif

    while (p<numberoffolders)

Variable NumberOfWaves =countobjectsDFR(\$(thefolder),1)
String AllTheWaves=wavelist("*",";","")

    Do
        String JustoneWave=StringFromlist(i,Allthewaves)

        If(stringmatch(Justonewave, "*_0*")==0) //this attempts
            to find all waves taht were part of multiruns,
           which wil have _XXXX where XXXX is
           0000,0001,0002...9999.
            Movewave \$Justonewave, :singles:       //will
               move, non-multirun waves to their own folder
        Endif

                i+=1
    While (i<numberofwaves)
//previous to this point is just identifying the folder of interest
   and making sure taht the only waves within it are ones that are
   part of a multifun

//Now begins the averaging. First thing to check is if there is a
   destination for each average. Use same checking method as earlier
   for "singles" folder.
i=0
p=0
 numberofFolders =countobjectsdfr(Root:,4)

    do
        string AreyouAverage =getindexedobjnamedfr(root:,4,p)

        if(stringmatch(areyouAverage,"AverageWaves")==0&&(p==
           numberoffolders-1)||numberoffolders==0)
            newdatafolder Root:AverageWaves
            p=numberoffolders
                                        //Looks to see
               if there is already a folder named "singles"
               in which to put single runs, if so, it puts
               them in there, if not, it makes one.
        elseif (stringmatch(areyouaverage,"AverageWaves")==1)
                p+=numberoffolders
        else
                p+=1
        endif
    while (p<numberoffolders)

p=0
i=0
```

```
NumberOfWaves =countobjectsDFR(\$(thefolder),1)
AllTheWaves=wavelist("*",";","") //at this point we want to start
   creating average waves by looking through lists and dentifying
   waves with similar prefixes (Name_XXXX, remove the XXXX and use
   that to match)
       Do
              String Lookingatyou=Stringfromlist(P,Allthewaves)
              String LookingatyouNext=Stringfromlist(P+1,Allthewaves)
              Variable Length1=strlen(lookingatyou)
              Variable Length2=strlen(lookingatyounext) //need to
                 make a wave that they will all add into!
              Variable SameRun=cmpstr(Lookingatyou[0,length1-6],
                 Lookingatyounext[0,length2-6]) //eliminates the
                 _xxxx from the wave so we are just comparing the
                 prefixes Wave_0000-->Wave

                  If(i==0)
                         string LookingatyourAverage =
                            lookingatyou[0,LENGTH1-6]+"_Avg"
                         Make/o/n=(numpnts(\$Lookingatyou)) \
                            $lookingatyouraverage
                         wave W_lookingatyouraverage=\
                            $lookingatyouraverage
                         wave W_lookingatyou=\$lookingatyou
                         W_lookingatyouraverage=W_lookingatyou

                         i+=1
                  elseif(sameRun ==0)
                         WAVE w_lookingatyou = \$lookingatyou
                         W_lookingatyouraverage=
                            W_lookingatyouraverage+W_lookingatyou
                            //Sorts and averages waves based on
                            prefixes
                         i+=1
                  Elseif(Samerun!=0)
                         WAVE w_lookingatyou = \$lookingatyou
                         W_lookingatyouraverage=
                            W_lookingatyouraverage+W_lookingatyou
                         W_Lookingatyouraverage=(
                            W_lookingatyouraverage)/(i+1)
                         MOVEWAVE w_lookingatyouraverage, Root:
                            averagewaves:
                         i=0
                  endif
              p+=1
       While(p<Numberofwaves)


       setdatafolder root:
       end
```

A.3 FORC Preparation Programs

The FORC Programs are generally part of a flow and should be used in the order presented here. The first program (LoaderOsci()) takes raw data from the oscilloscope and loads it into Igorpro. Care must be taken that the maximum amount of data is not exceeded within Igorpro. It is often wise to Split the raw data into folders with no more than 4 GB of data in each. The next program, BadEgg(), will look for errors as outlined in figures **??** and **??**.When gathering FORC data, the gauss probe will infrequently fail to record the field sweeps properly. This program looks a the min and max values of the field sweeps and compares them, within a threshold, to neighboring field sweeps. The program marks a value as rotten when the values fall outside of the threshold range. The index of the bad run is marked and the name of the field file, and corresponding MOKE is recorded. Next, we use TimeEqualizer() to take the MOKE and Field data which are both formatted in 2 column format: Signal,Time, and match up the indices by time so that we can remove time as an axis, knowing that we will now have MOKE and Field corresponding at each index for each point. Next we use our data, which has been cleaned of bad runs and is in the Kerr-Field pairs as desired, and we sort each wave by it's reversal number which was determined from the BaggEqq() prrogram, when looking at min field values (which determines at which field value a reversal occurs). Here, also, the bad eggs are deleted and the program will produce one averaged set of data for each field/Kerr pair for each reversal. The next step is optional but recommended. Using the Smoother() program, the user is able to look at each piece of data, wether it be the Kerr or Field data, and apply a LOESS or Box smoothing algorithm to reduce gaussian-type noise. This is encouraged but not mandatory. Additionally, the program allows for exponential fitting between a subset of data points determined by user placed cursors.The panels for user interface are shown here.

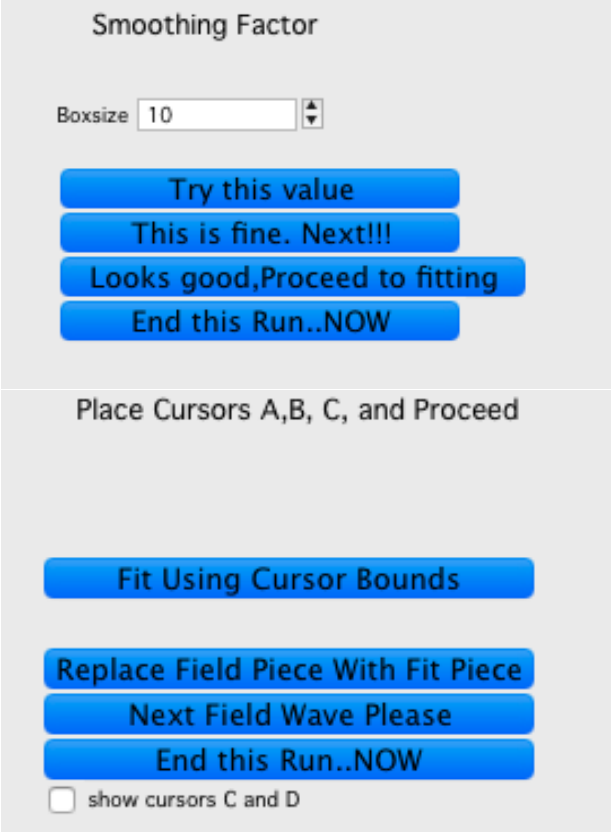

Figure A.1 The top panel allows the user to try various box sizes for numerical smoothing and will display the resulting graph as a result of the value chosen. This top box will continue to appear until the user selects any of the bottom 3 options. The lower panel appears if we select the "Looks good, Proceed to fitting" option. This is used to fit the saturation limits of the magnetic field with a single exponential to overcome the poor resolution of the gauss probe.

Following a satusfactory fitting, the user has the option to replace the subset of points with the fit. This is especially useful for fixing the 25G minimum step issue that is displayed with certain gaussprobes near saturation values. Finally, we use OsciForcNorm() to normalize and split each reversal Kerr data so that it is only the increasing half. The user has the option to simply offset the minor reversals to line up with the sautration value of the largest outer reversal or offset every curve and normalized them so that the outer curve runs between ± 1 and the minor loops are scaled appropriately to the major loop. This particular workflow prepares the data for the actual FORC analysis.

A.3.1 Loading FORC Data from the Oscilloscope

```
#pragma rtGlobals=3          // Use modern global access method and
   strict wave access.
//This program specifically loads from the oscilloscope. The reason
   being is that the namin convention into igor assumes
//a naming convention coming from the osci. We assume that the name
   attached to the file is followed by 00000.txt which is
//important for naming AND sorting files with this program.
// Make sure there are no lingering files in the root before you start
    this. It will have trouble exiting
//possibly could add a quick loop to put all files in root into a
   temporary folder, then move them back after the program runs???

function LoaderOsci()
       //initialize loop variable
       Variable L=0
       variable FolderWatch=200
       variable Sorter = 0
       variable foldersorter=0
       variable i=0
       variable g=0
       string wname,fname         //wave names and file name,
          respectively
       String XorY
       getfilefolderinfo/D
       if (datafolderexists("A_waves")==0)
              newdatafolder/O root:A_waves
       endif
       if (datafolderexists("B_waves")==0)
              newdatafolder/O root:B_waves
       endif
       if (datafolderexists("Timewaves_A")==0)
              newdatafolder/O root:Timewaves_A
       endif
              if (datafolderexists("Timewaves_B")==0)
              newdatafolder/O root:Timewaves_B
       endif


       newpath/O DataAnalysis S_path
       //Create a list of all files that are .txt files in the folder
          . -1 parameter addresses all files.
       string filelist= indexedfile(DataAnalysis,-1,"????")
       filelist = SortList(filelist, ";", 16)

       //Pull the prefix of the first wave, compare each prefix to
          that of the first, if it matches, name it one way, if not
          name the other.
       //Begin processing the list
       do
              //store the ith name in the list into wname.
```

```
    fname = stringfromlist(i,filelist)

If  (stringmatch(fname ,".DS_Store")==1)
  i=i+1
     Fname = stringfromlist(i,filelist)

else

          string Timename
       string name
       variable totallength=strlen(fname)
       variable namelengthvariable=0
       L=0

       Do                  //pull the suffix number from the
          first non-zero digit following the name
                  string readspot= Fname[totallength-9+L]

                         if (str2num(readspot)!=0)
                                string Suffix= Fname[
                                   totallength-9+L,
                                   totallength-5]
                         endif
                  L+=1
            while(str2num(Readspot)==0)


            if(sorter==0)
                         string matchMeprefix = "C2" //
                            Changing this from fname[0,1]
                         string Actualprefix = fname[0,1]
                         sorter+=1
                  elseif(sorter!=0)
                         Actualprefix = fname[0,1]
                  endif


                  if (stringmatch(matchMeprefix,
                     Actualprefix)==1)
                  name = fname[2,totallength-10]+"_"+Suffix
                     +"a"
                       Timename="T_"+name

                  else
                         name = fname[2,totallength-10]+"_
                            "+Suffix+"b"
                                        //gootta add 2
                            more indexes here for when i
                       Timename="T_"+name

                                              //am moving
                          things to folders mid run.
```

```
            endif
               //used to be linear Time as X axis,
                  now linear magnetic field

              string info=""

              info+="N='"+Timename+"';"
              info += "N='"+name+"';"
              info+= "N='_skip_';N='_skip_';"

                     Loadwave/a/b=info/d/J/Q/P=
                        DataAnalysis stringfromlist(i
                        ,filelist)

                     // if(i==(i+50))
                           print  "Loaded "+fname
                    //  endif

if(i == folderwatch)

            do
                    string overflowlist=WaveList
                       ("*",";","")
                    string overflowname =
                       stringfromlist(0,overflowlist)
                    wave overflowwave= $overflowname
                    if(stringmatch(overflowname[strlen
                       (overflowname)-1],"A")==1&&(
                       stringmatch(overflowname[0],"T
                       ")==0))
                           deletepoints/M=0 0, 5,
                              overflowwave
                           Movewave overflowwave,
                              root:A_waves:
                    elseif(stringmatch(overflowname[
                       strlen(overflowname)-1],"B")
                       ==1&&(stringmatch(overflowname
                       [0],"T")==0))
                           deletepoints/M=0 0, 5,
                              overflowwave
                           Movewave overflowwave,
                              root:B_waves:
                    elseif(stringmatch(overflowname
                       [0],"T")==1&&stringmatch(
                       overflowname[strlen(
                       overflowname)-1],"B")==1)
                           deletepoints/M=0 0, 5,
                              overflowwave
                           Movewave overflowwave,
                              root:Timewaves_B:
                    elseif(stringmatch(overflowname
                       [0],"T")==1&&stringmatch(
                       overflowname[strlen(
```

```
                                        overflowname)-1],"A")==1)
                                              deletepoints/M=0 0, 5,
                                                 overflowwave
                                              Movewave overflowwave,
                                                 root:Timewaves_A:

                                     endif

                                 while(Countobjectsdfr(root:,1)!=0)

                           folderwatch+=200
                           endif

                     i =i+1                //move to next file
                     endif

               while(i<itemsinlist(filelist))              //end when all
                  files are processed.

do
                                 overflowlist=WaveList("*",";","")
                                 overflowname =stringfromlist(0,
                                    overflowlist)
                                 wave overflowwave= $overflowname
                                 if(stringmatch(overflowname[strlen(
                                    overflowname)-1],"A")==1&&(stringmatch
                                    (overflowname[0],"T")==0))
                                              deletepoints/M=0 0, 5,
                                                 overflowwave
                                              Movewave overflowwave,
                                                 root:A_waves:
                                     elseif(stringmatch(overflowname[
                                        strlen(overflowname)-1],"B")
                                        ==1&&(stringmatch(overflowname
                                        [0],"T")==0))
                                              deletepoints/M=0 0, 5,
                                                 overflowwave
                                              Movewave overflowwave,
                                                 root:B_waves:
                                     elseif(stringmatch(overflowname
                                        [0],"T")==1&&stringmatch(
                                        overflowname[strlen(
                                        overflowname)-1],"B")==1)
                                              deletepoints/M=0 0, 5,
                                                 overflowwave
                                              Movewave overflowwave,
                                                 root:Timewaves_B:
                                     elseif(stringmatch(overflowname
                                        [0],"T")==1&&stringmatch(
                                        overflowname[strlen(
                                        overflowname)-1],"A")==1)
                                              deletepoints/M=0 0, 5,
                                                 overflowwave
```

```
                                                Movewave overflowwave,
                                                   root:Timewaves_A:
                                        endif
                                while(Countobjectsdfr(root:,1)!=0)

                Print "Presto Load-o!!"
                print "The next thing you want to run is Badegg()"
end
%
```

A.3.2 Bad Run Finder

```
#pragma rtGlobals=3          // Use modern global access method and
   strict wave access.

Function Badegg()
//Look at the min field value where it reverses. Find the avg value
   there at the bottom.
//make an indexed list that has in it the value of the min field.
//we can give it an approx point value to start at AND do the slope
   thing
variable stinker=0
variable maxstinker= 0
variable i=0
variable s=0
variable r=1
variable wereversed=0
variable deltaH =.8371/100 //this should be the total H range / number
    of reversals, here it was approx .85/80 i think
variable MinThreshFraction=.15 //these are picked arbitraily
variable MaxThreshFraction=.006
Variable Minpoint1=4800
variable minpoint2=5200
variable pointsfromEnd=300
variable thisminvalue, thismaxvalue, lastminvalue,lastmaxvalue,
   nextminvalue,nextmaxvalue,mindeltalast,maxdeltalast,mindeltanext,
   maxdeltanext
//Execute "CreateBrowser prompt=\"Find the folder with Field Waves\",
   showWaves=1, showVars=0, showStrs=0" //asks you to find a wave,
   who'se path is stored as S_browserlist
//SVAR S_BrowserList=S_BrowserList //no clue why this is here
//variable pathlength2 =strlen(s_browserlist) //length, in numbers, of
    the path name.
//string waveLocationB= s_browserlist[0,pathlength2-2] //cuts
   thesemicolon from the path name
string wavelocationb="Root:B_waves"

make/o/n=1 $(wavelocationb+":Trash")=1 //this fixes our endpoint
   problems, but adds an extra wave to the list which is delteed at
   the end
wave trash=    $(wavelocationb+":Trash")

       if(stringmatch(wavelocationB,"*B_waves*")==1)
```

```
                setdatafolder Root:B_waves
                                        //we look at the field waves
                   because they are the X axis, if they are bad, then
                   there is no reason to look at the corresponding MOKE
                string list1=WaveList("*",";","")
                setdatafolder Root:
        else
                setdatafolder root:A_waves
                list1=WaveList("*",";","")
                Setdatafolder root:
        endif


Make/n=1/o MinValues
Make/n=1/o MaxValues
Make/n=1/o/t Indexednames
Make/n=1/o StinkerList



variable Numfiles= countobjectsDFR($(wavelocationB),1)//have this to
   try to avoid lastpoint errors

Do

        if(strlen(stringfromlist(i+1,list1))==0)
                break
        endif

        if(i!=0)
                string lastfieldname = wavelocationB+":'"+
                   stringfromlist(i-1,list1)+"'"        //construct the
                    path to the previousfieldFWave
                wave lastfieldwave= $lastfieldname
        endif
                string thisfieldname = wavelocationB+":'"+
                   stringfromlist(i,list1)+"'"          //construct the
                    path to the fieldFWave
                wave thisfieldwave= $thisfieldname
                string nextfieldname = wavelocationB+":'"+
                   stringfromlist(i+1,list1)+"'"        //construct the
                    path to the next fieldFWave
                wave nextfieldwave= $nextfieldname


//it also makes our exit condition look at the lenght of the NEXT wave
    since this is the last wave and I ultimately intend to skip it!
       if(i==0)
                wavestats/q/r=[minpoint1,minpoint2]/w thisfieldwave

                      wave m_wavestats                    //the 4800
                   is picked based on this particular data set,
                   one should look at the data you are using to
```

```
                        find where the field becomesflat at the
                        minimum. The time should be the same
                    minvalues[i]={M_wavestats[3]}
                wavestats/q/r=[numpnts(thisfieldwave)-pointsfromend,
                   numpnts(thisfieldwave)-1]/w thisfieldwave        //
                   just iek the min, the max index is chosen
                   arbitralily
                       maxvalues[i]={m_wavestats[3]}
                wavestats/q/r=[minpoint1,minpoint2]/w nextfieldwave
                       minvalues[i+1]={M_wavestats[3]}
                wavestats/q/r=[numpnts(nextfieldwave)-pointsfromend,
                   numpnts(nextfieldwave)-1]/w nextfieldwave        //
                   just iek the min, the max index is chosen
                   arbitralily
                       maxvalues[i+1]={m_wavestats[3]}
        elseif(i>0&&i!=numfiles-2)
                wavestats/q/r=[Minpoint1,minpoint2]/w nextfieldwave
                       minvalues[i+1]={M_wavestats[3]}
                wavestats/q/r=[numpnts(nextfieldwave)-pointsfromend,
                   numpnts(nextfieldwave)-1]/w nextfieldwave        //
                   just iek the min, the max index is chosen
                   arbitralily
                       maxvalues[i+1]={m_wavestats[3]}
        endif
                                        /// for all reversal field
           values because of how the labview is written to hit the min
            at 5 sec or whatever, it will hit min at 1/2 totat time

        IndexedNames[i]={nameofwave(thisfieldwave)}
         ThisMinValue = minvalues[i]

            // since we are threshholding, we are comparing points
            with eachother, looking for consistancy.
         ThisMaxValue=Maxvalues[i]

if(i!=numfiles-2)
         nextminvalue=minvalues[i+1]
         nextmaxvalue=maxvalues[i+1]
endif

         if(i!=0)
                lastminvalue=minvalues[i-1]
                lastmaxvalue=maxvalues[i-1]
         endif

        if (stinker==1||Maxstinker==1)
                lastminvalue=minvalues[i-2]
                lastmaxvalue=maxvalues[i-2]
        endif
        MindeltaLast=abs(thisminvalue-lastMinValue)
        MaxdeltaLast=abs(thisMaxvalue-lastMaxValue)
        MindeltaNext=abs(thisminvalue-NextMinValue)
```

```
        MaxdeltaNext=abs(thisMaxvalue-NextMaxValue)
if(i==7979)
        print "oi"
endif

        if(i>0&&Maxdeltalast<=Maxthreshfraction*Lastmaxvalue||
           MaxdeltaNext<=Maxthreshfraction*Nextmaxvalue)  //checks if
           the previous point minus the current point fall within the
           threshold for the MAX value. this catches large errors
               maxstinker=0
                                                              //if it
               falls within, we say that there are no such errros
               like that

               if(i>0&&MindeltaNext<=(minthreshFraction*deltaH)||
                  MindeltaLast<=(MinThreshFraction*deltaH))
                  //if there are no MAX errors, check the min. Does it
                   fall in the threshold?
                      stinker=0

               elseif(i>0&&thisminvalue>=lastMinValue-(1+1.2*
                  MinThreshFraction)*DeltaH&&thisminvalue<=
                  lastMinValue-(1-1.2*MinThreshFraction)*DeltaH) //
                  this looks for a threshhold that is displaced by
                  DelttaH ( the reversal step sizes)
                      stinker=0
               elseif(i>0&&thisminvalue>=nextMinValue-(1+1.2*
                  MinThreshFraction)*DeltaH&&thisminvalue<=
                  nextMinValue-(1-1.2*MinThreshFraction)*DeltaH)
                      stinker=0
               elseif(i>0)
                      stinkerlist[s]={i}                  // if it
                         falls outside of the threshold, mark as
                         stinker, then check the point after in the
                         later (earlier, up the page) part of the code
                      s+=1
                      stinker=1
               endif

        elseif(i>0&&Maxdeltalast>=Maxthreshfraction*Lastmaxvalue||
           MaxdeltaNext>=Maxthreshfraction*nextmaxvalue)  //if there
           is a max stinker, mark it as such
               stinkerlist[s]={i}
               s+=1
               MaxStinker=1
        elseif(i==0)

        endif

        i+=1
while(1)
killwaves trash
```

```
//indexednames[i]={nextfieldname}

duplicate/o minvalues, Cut_minvalues              //after the points
   have all been hit , make duplicates of the lists we have been
   making and remove the stinnker indicies from those lists
duplicate/o maxvalues,Cut_maxValues
duplicate/o/t IndexedNames,Cut_indexednames
variable p=1
do

        variable badIndex=stinkerlist[numpnts(stinkerlist)-p]
        deletepoints badindex,1,cut_maxvalues
        deletepoints badindex,1,cut_minvalues
        deletepoints badindex, 1, cut_IndexedNames
        p+=1
while(p<=numpnts(stinkerlist))
//Now I have to find reversals. Should be easier to do sepearetley
   since there are no more errors to dodge around.


i=0
if(abs(cut_maxvalues[0]-cut_minvalues[0])<abs(cut_maxvalues[numpnts(
   Cut_minvalues)-1]-cut_minvalues[numpnts(cut_minvalues)-1]))
        Make/o/n=0 Reversing_minvalues
        Make/o/n=0 Reversing_maxvalues
        Make/o/t/N=0 Reversing_Names
        //Make/o/N=0 ReversingReversal_index

        Do
                Reversing_minvalues[i]={Cut_minvalues[numpnts(
                   cut_minvalues)-1-i]}
                Reversing_maxvalues[i]={Cut_Maxvalues[numpnts(
                   cut_maxvalues)-1-i]}
                string newname=cut_indexedNames[numpnts(
                   cut_indexednames)-1-i]
                Reversing_Names[i]={newname}
                i+=1
        while(i<numpnts(cut_minvalues))
 WeReversed=1
 duplicate/o Reversing_minvalues, Cut_minvalues
 duplicate/o Reversing_Maxvalues,Cut_maxvalues
 duplicate/o/t Reversing_names,Cut_Indexednames
 Killwaves Reversing_minvalues,reversing_maxvalues,Reversing_names
endif

Make/o/n=1 Reversal_Index


variable C=0
Do

if(c==numpnts(cut_minvalues)-1)
        Reversal_Index[c]={r}
```

```
        break
endif


        thisminvalue = Cut_minvalues[c]
        nextminvalue= Cut_minvalues[c+1]

If(abs(thisminvalue-nextminvalue)<=(7/16)*deltaH)
        reversal_index[c]={r}
else

        reversal_index[c]={r}
        r+=1
endif
c+=1
while(1)


print "Does it look good? If so, Move on to run TimeEqualizer()!"
Print "If not, just change some threshold parameters and try again!"
end
%
```

A.3.3 Matching Time Axes

```
#pragma rtGlobals=3          // Use modern global access method and
   strict wave access.
Function timeEqualizer()

        variable v_value
        variable i=0


        string timelocationa="Root:Timewaves_A"
        string timelocationb="Root:Timewaves_B"
        string wavelocationa="Root:A_waves"
        string wavelocationb="Root:B_waves"
        setdatafolder $(TimelocationA)
        String TimelistA=wavelist("*",";","")
        setdatafolder $(TimelocationB)
        String TimelistB=wavelist("*",";","")
        setdatafolder $(WavelocationA)
        String ListA=wavelist("*",";","")
        setdatafolder $(WavelocationB)
        String ListB=wavelist("*",";","")
        setdatafolder root:

        Do

        string TimeNameA = Stringfromlist(i,TimelistA)

        if(strlen(timenameA)==0)
                        Break
```

```
        endif

    Wave TimeWaveA= $(TimeLocationA+":"+"'"+TimenameA+"'")


    string NameA = Stringfromlist(i,listA)
    Wave waveA= $(waveLocationA+":"+"'"+nameA+"'")

    string TimeNameB= Stringfromlist(i,TimelistB)
    Wave timewaveB= $(TimeLocationB+":"+"'"+TimenameB+"'")

    string NameB = Stringfromlist(i,listB)
    Wave waveB= $(waveLocationB+":"+"'"+nameB+"'")

    Variable MinA= Wavemin(timewaveA)
    variable MinB=wavemin(timewaveB)
    variable LargerMin = max(mina, minb)

        if(Mina/Largermin==1)
            Findvalue/T=.0005/v=(largermin) TimewaveB
            deletepoints 0, v_value, TimewaveB
            deletepoints 0,v_value, waveB
        else
            Findvalue/T=.0005/v=(largermin) timewaveA
            deletepoints 0, v_value, timewaveA
            deletepoints 0,v_value, waveA
        endif

    variable pointsA=numpnts(TimewaveA)
    variable pointsB=numpnts(TimewaveB)

        If(pointsA!=pointsB)
            if(pointsb>PointsA)
                deletepoints pointsA, (pointsB-pointsA),
                   timewaveB
                deletepoints pointsA, (pointsB-pointsA),
                   waveB
            Else
                deletepoints pointsB, (pointsA-pointsB),
                   timewaveA
                deletepoints pointsB, (pointsA-pointsB),
                   waveA
            Endif
        endif

    i+=1

    while(1)
    Print "YouR TIMES HAVE BEEN EQUALIZED!"
    Print "Run cutterAndSorter() next '_'"
end
%
```

A.3.4 FORC Averaging and Grouping

```
#pragma rtGlobals=3          // Use modern global access method and
   strict wave access.

Function CutterAndSorter()
variable Mover=0
variable i=0
variable R=1
wave reversal_index
string Foldera= "A_waves"
String FolderB= "B_waves"                              //This procedure
   uses the Reversal_index wave from the previous prcedure to group
String TimeA="Timewaves_A"                  //All waves by which
   reversal field they are associated with and take an average. A bad
    Reversal index wave
String TimeB="Timewaves_B"                  // can make this bad. If
   we are using the multi-batch Loading from Forc, You need to change
string folderpathstringa                                    //the
   reversal indicies somehow
string folderpathstringb,oldfolder
string FolderPathStringTimeA
String FolderPathStringTimeB
variable v_value,R_offset
wave/t cut_indexednames
//Introduce an R offset? so that when naming, they end up written as
   the correct reversal? right here!?
Prompt R_offset, "Is the first reversal 1 or is it something else?"
Doprompt "Input the starting Reversal (even if it's 1)", R_offset
R_offset -=1 //R_offset, is actually going to be 1 less than the input
    value?
//eg. if the last reversal was 34, then the offset would want me to
   start naming at 35, instead of 1, so i would say the first
   reversal is 35
// and it would take 35-1=34 and add 34 to the index, so instead of 1,
    i would get 34+1=35
Do

        FolderPathStringB= "root:B_waves:R_"+num2str(reversal_index[i
           ]+R_offset)

        if (datafolderexists(folderpathstringB)!=1)
              Newdatafolder $(folderpathstringB)
              FolderPathStringA= "root:A_waves:R_"+num2str(
                 reversal_index[i]+R_offset)
              Newdatafolder $(folderpathstringA)
        //      FolderPathStringTimeA= "root:Timewaves_a:R_"+num2str(
           reversal_index[i]+R_offset)
        //      Newdatafolder $(folderpathstringTimeA)
        //      FolderPathStringTimeB= "root:Timewaves_B:R_"+num2str(
           reversal_index[i]+R_offset)
        //      Newdatafolder $(folderpathstringTimeB)
        endif
```

```
        i+=1
while(i<numpnts(reversal_index))

i=0
//setdatafolder root:timewaves_A
//String TimelistA=wavelist("*",";","") //I think it's easier to move
   all the waves then do killwaves/a to all the stragglers!
//setdatafolder root:Timewaves_B
//string timelistB=wavelist("*",";","")
setdatafolder root:A_waves
string ListA=wavelist("*",";","")
setdatafolder root:B_waves
string listB=wavelist("*",";","")
setdatafolder root:


prompt OldFolder, "yes or no"
doprompt "Have you already put waves in reversal Folders?", oldfolder

if(stringmatch(oldfolder,"no")==1)
Do
//the index of the reversal and the index of the name are shared, so i
    can build a matchstring to put the file in the right folder
//=reversal_index[i]
//which wavename are we trying to move


string NameOfMover=cut_indexednames[i]
Variable MoverIndex=Whichlistitem(NameofMover,ListB)
FolderPathStringA= "root:A_waves:R_"+num2str(reversal_index[i]+
   R_offset)
//FolderPathStringTimeA= "root:Timewaves_a:R_"+num2str(reversal_index[
   i]+R_offset)
//FolderPathStringTimeB= "root:Timewaves_B:R_"+num2str(reversal_index[
   i]+R_offset)
FolderPathStringB= "root:B_waves:R_"+num2str(reversal_index[i]+
   R_offset)

if(moverindex!=-1)
        string WavenameA=stringfromlist(Moverindex,ListA)
        string WavenameB=stringfromlist(MoverIndex,ListB)
//      string TimeNameA=stringfromlist(Moverindex,TimeListA)
//      string TimeNameB=stringfromlist(MoverIndex,TimeListB)

        Movewave Root:A_waves:$(wavenameA), $(FolderpathstringA)+":"
        Movewave Root:B_waves:$(wavenameB), $(FolderpathstringB)+":"
//      Movewave Root:Timewaves_A:$(timenameA), $(
   FolderpathstringTimeA)+":"
//      Movewave Root:Timewaves_B:$(timenameB), $(
   FolderpathstringTimeB)+":"
endif
```

```
i+=1
while(i<numpnts(cut_indexednames))

endif


//setdatafolder root:timewaves_A
//killwaves/a
//setdatafolder root:timewaves_B // we clear out the folders since now
    each wave is in it's own named folder
//killwaves/a
setdatafolder root:B_waves
Killwaves/a
Setdatafolder root:A_waves
Killwaves/a
setdatafolder Root:

Variable ReversalFolders=countobjectsdfr(Root:B_waves,4)
i=0
Do

        FolderPathStringA= "root:A_waves:R_"+num2str(r+R_offset)
        setdatafolder FolderpathstringA
        listA=wavelist("*",";","")

//      FolderPathStringTimeA= "root:Timewaves_a:R_"+num2str(r+
   R_offset)
//      setdatafolder FolderpathstringTIMEA
//      TimelistA=wavelist("*",";","")

//      FolderPathStringTimeB= "root:Timewaves_B:R_"+num2str(r+
   R_offset)
//      setdatafolder FolderpathstringTIMEB
//      TimelistB=wavelist("*",";","")

        FolderPathStringB= "root:B_waves:R_"+num2str(r+R_offset)
        setdatafolder FolderpathstringB
        listB=wavelist("*",";","")

        string pointsname=getindexedobjnamedfr($(folderpathstringa)
            ,1,0)
        wave Pointswave=$(folderpathstringa+":"+"'"+pointsname+"'") //
            since we are doing averaging, this pretty much sets the
            denominator in which we
//divide each sum by. It seems liek it wouldn't be too hard to just do
    this for each wave in case they have different numbers of points.
        variable Points=numpnts(pointswave)
        variable denominator= countobjectsdfr($(folderpathstringA),1)
        setdatafolder root:
        i=0

        Do
```

```
                string ReversalName="R_"+num2str(r+R_offset)

                if(i==0)

                        string AvgNameA="root:A_waves:"+reversalname
                        string AvgNameB="root:B_waves:"+reversalname
//                      string AvgNameT_B="root:Timewaves_B:"+
   reversalname
//                      string AvgNameT_A="root:Timewaves_A:"+
   reversalname
                        Make/n=(points)/o $(AvgNamea)=0
                        Make/n=(points)/o $(AvgNameB)=0
//                      Make/n=(points)/o $(AvgnameT_B)=0
//                      Make/n=(points)/o $(AvgnameT_A)=0
                        wave avgwavea= $(AvgNamea)
                        wave avgwaveB=$(AvgNameB)
//                      wave avgTimeA=$(AvgNameT_A)
//                      wave avgTimeB=$(AvgNameT_B)

                endif

                if(strlen(stringfromlist(i,lista))==0)
                        avgwavea=avgwavea/denominator
                        avgwaveb=avgwaveb/denominator
//                      avgTimea=avgtimea/denominator
//                      avgtimeb=avgtimeb/denominator

                        break
                endif

                string activename=folderpathstringA+":"+"'"+
                   stringfromlist(i,lista)+"'"
                wave activewave=$(activename)

                avgwavea=Avgwavea+activewave

                activename=folderpathstringB+":"+"'"+stringfromlist(i,
                   listB)+"'"
                wave activewave=$(activename)
                avgwaveB=AvgwaveB+activewave

//              activename=folderpathstringTIMEA+":"+"'"+stringfromlist
   (i,Timelista)+"'"
//              wave activewave=$activename
//              avgTimea=AvgTimea+activewave

//              activeName=folderpathstringTimeB+":"+"'"+stringfromlist
   (i,TimelistB)+"'"
//              wave activewave=$activename
//              avgTimeB=AvgTimeB+activewave


                i+=1
```

```
        while(1)
        r+=1
while(R<=reversalfolders)
Print "now That all the files are averaged appropriately, you might
   want to look into smoothing those X -Axis field waves"
print "With Smoother() and then move onto osciForcNorm()"
end

//Look at the name of the file, then move it to the folder of
   folderpathstring
%
```

A.3.5 SMOOTHER AND REPLACER

```
#pragma rtGlobals=3          // Use modern global access method and
   strict wave access.

function Smoother()
string LoessOrBox

Execute "CreateBrowser prompt=\"Find the folder with the waves to be
   smoothed\", showWaves=0, showVars=0, showStrs=0" //asks you to
   find a wave, who'se path is stored as S_browserlist
        SVAR S_BrowserList=S_BrowserList //no clue why this is here
        variable pathlength =strlen(s_browserlist) //length, in
           numbers, of the path name.
        string FieldLocation= s_browserlist[0,pathlength-2] //cuts
           thesemicolon from the path name
        string fieldname
        variable i=0
        variable dontstartatzero=1
        variable/g boxsize=10
        variable/g ender=0
        string/g process ="Smooth"
        Setdatafolder \$(fieldlocation)
        string Fieldlist=wavelist("*",";","")
        setdatafolder root:
        string SmoothedFolder=FieldLocation+":SmoothedFields"
        dfref SmoothedFolderDfref =\$(smoothedfolder)
if(datafolderexists(smoothedFolder)==0)
        Newdatafolder \$(smoothedFolder)
endif
if(whichlistitem("SmoothOperator",winlist("*",";","Win:64"))!=-1)
        killwindow SmoothOperator
elseif(whichlistitem("FitManager",winlist("*",";","Win:64"))!=-1)
        killwindow FitManager
endif

Prompt LoessorBox, "Loess Or Box Smoothing",popup "Loess;Box"
DoPrompt "Loess or Box smoothing?" , LoessOrBox
prompt Dontstartatzero, "Which wave is up next?"
doprompt "Did you stop in the middle of a run, which wave should we
   start on", Dontstartatzero
```

```
i=dontstartatzero-1

do
                        fieldname= fieldlocation+":'"+stringfromlist(i,
                           fieldlist)+"'"

                                if(strlen(stringfromlist(i,fieldlist))
                                   ==0)
                                    break
                                endif

                        wave fieldwave=\$fieldname
                if(stringmatch(Process,"Smooth") ==1)
                        String SmoothedName=SmoothedFOlder+":"+"'S_"+
                           nameofwave(fieldwave)+"'"
                        duplicate/o \$fieldname, \$SmoothedName
                        wave smoothedwave=\$smoothedName
                endif
                        if(whichlistitem("SmoothingGraph",winlist
                           ("*",";",""))==-1) //if the window isn't
                           there, make one
                                Display/N=SmoothingGraph/w
                                   =(300,300,1500,1000) fieldwave
                        endif

        //gotta do "if trace exists" kinda thing here. Only puts the
           trace on the graph if it isn't there already. Had issues of
            this making 100000 copies of the same trace.

                String NameOfGraphTraces = TraceNamelist("
                   SmoothingGraph",";",1)
                variable TraceonGraph=strsearch(NameofGraphTraces,
                   nameofwave(Smoothedwave),0)
                //print traceongraph
                        if(traceongraph==-1)
                                appendtograph/c=(1,1,50000)/w=
                                   SmoothingGraph Smoothedwave
                        endif

        if(stringmatch(Process,"Smooth") ==1)

                if(Stringmatch(LoessorBox,"Loess"))
                        Loess/v=0/N=(Boxsize) srcwave=Smoothedwave
                else
                        Smooth/B boxsize, Smoothedwave
                endif
        elseif (stringmatch(Process,"Fit") ==1)
                if(ender==4)
                        variable j=0
                        Curvefit/q/L=(pcsr(B)-Pcsr(A)+1)/Nthr=0
                           Exp_XOffset smoothedwave[PCSR(A),PCSR(B)] /D
                        Make/o/N=(numpnts(smoothedwave)) ExtendedFit =
                           nan
```

```
                wave w_coef
                ExtendedFit[0,pcsr(c)-pcsr(a)+1] = W_coef(0)+
                   W_coef(1)*exp(-(x/w_coef(2)))
                make/n=(numpnts(smoothedwave))/o
                   ExtendedFit_offset=nan
                       Do
                              extendedfit_offset[j+pcsr(a)]=
                                 extendedfit[j]
                              j+=1
                       while(j+pcsr(a)<=pcsr(c))

                Appendtograph/c=(0,0,4000) extendedfit_offset
        elseif(ender==3) //replaceing wave with fit
                variable p=0
                //String NameofFit = "fit_"+Nameofwave(
                   smoothedwave)
                //wave fitwave= \$nameoffit
                Do
                       smoothedwave[Pcsr(A)+p]=
                          extendedfit_offset[p+pcsr(a)]
                       p+=1
                while(Pcsr(A)+P<=Pcsr(C))
                removefromgraph/w=smoothingGraph
                   extendedfit_offset
        //      killwaves extendedfit,extendedfit_offset
        endif
endif

if(stringmatch(Process,"Smooth") ==1)//Topdown programminglul.
    So you can do multiple fits in a row
        doupdate /w=smoothingGraph
        NewPanel /W=(150,50,450,239)
        DoWindow/C SmoothOperator// set to an unlikely name
        DrawText 43,23,"Smoothing Factor"
        SetVariable setvar0,pos={27,49},size={126,17},limits={-
           Inf,Inf,1}
        SetVariable setvar0,value= root:Boxsize
        Button button0,pos={30,80},size={180,20}
        Button button0,proc=ReplaceValue ,title="Try this value
           "
        Button button1,pos={30,100},size={180,20} ///These
           Functions are in the ButtonDemo() procedure (proceed
            replacevalue and leaving)
        Button button1,proc=Proceed,title="This is fine. Next
           !!!"
        Button button3,pos={30,120},size={210,20}
        Button button3,proc=Proceed2Fits,title="Looks good,
           Proceed to fitting"
        Button button2,pos={30,140},size={180,20}
        Button button2,proc=Leaving,title="End this Run..NOW"
        pauseforuser smoothoperator, smoothinggraph
elseif(stringmatch(Process,"Smooth")==0)
        //begin a new panel for doing the fitting stuff and
```

```
                have it all here
            doupdate /w=smoothingGraph
            NewPanel /W=(150,50,450,260)
            DoWindow/C FitManager// set to an unlikely name
            showinfo/cp=0/w=SmoothingGraph // Shows the cursor
                selection so that I can pick a range to fit within.
            DrawText 43,23,"Place Cursors A,B, C, and Proceed"
            Button button0,pos={30,80},size={215,20}
            Button button0,proc=FitWithCursors ,title="Fit Using
                Cursor Bounds"
            //Button button4,pos={30,100},size={215,20}
            //Button button4,proc=theextracursor,title="Cursors Are
                 placed for extended fit"
            Button button1,pos={30,120},size={215,20} ///These
                Functions are in the ButtonDemo() procedure (proceed
                 replacevalue and leaving)
            Button button1,proc=ReplaceWithFit,title="Replace Field
                 Piece With Fit Piece"
            Button button3,pos={30,140},size={215,20}
            Button button3,proc=Proceed,title="Next Field Wave
                Please"
            Button button2,pos={30,160},size={215,20}
            Button button2,proc=Leaving,title="End this Run..NOW"
            Checkbox Check1,pos={30,180}
            checkbox check1, Proc=ShowCD,title="show cursors C and
                D"
            pauseforuser Fitmanager, smoothinggraph
        endif
            if(ender==1) //to end program
                    //killwindow SmoothingGraph
                    abort

            elseif(ender==2) //to move to next graph to fit

                    killwindow SmoothingGraph
                    i+=1
            //elseif(ender==3)//if we are keeping the same graph
                but now looking at the Fit Stuff

            endif


while(1)
        Print "Now run the OsciForcNorm() with your new smoothed x-
            Axes"
end
```

These are functions associated with the smoother button panel.

```
        Function Proceed2Fits(CtrlName) : ButtonControl
string ctrlName

variable/g ender= 0 //to just return to operating on the same wave,
    but changing process to smooth switches button panel
```

```
string/g process= "Fit"
if(whichlistitem("SmoothOperator",winlist("*",";","Win:64"))!=-1)
	killwindow SmoothOperator
elseif(whichlistitem("FitManager",winlist("*",";","Win:64"))!=-1)
	killwindow Fitmanager
endif

end


Function FitWithCursors(CtrlName) : ButtonControl
string ctrlName

variable/g ender= 4 //to just return to the same loop after fitting
string/g process= "Fit"
if(whichlistitem("SmoothOperator",winlist("*",";","Win:64"))!=-1)
	killwindow SmoothOperator
elseif(whichlistitem("FitManager",winlist("*",";","Win:64"))!=-1)
	killwindow Fitmanager
endif

end


Function ReplaceWithFit(CtrlName) : ButtonControl
string ctrlName

variable/g ender= 3 //if you just did a fit, liked it, then clicked to
    have it replace a piece of wave. In which this will re-direct
//you to a process where we make a duplicate wave, and replace the
   points with tthe fit, then look at that wave instead
string/g process= "Fit"
if(whichlistitem("SmoothOperator",winlist("*",";","Win:64"))!=-1)
	killwindow SmoothOperator
elseif(whichlistitem("FitManager",winlist("*",";","Win:64"))!=-1)
	killwindow FitManager
endif

end


Function ThatExtraCursor(CtrlName) : ButtonControl //this is going to
   have to do with the 3rd cursor thingie for better fits *Beanface*
string ctrlName

variable/g ender= 4 //to just return to the same loop after fitting
string/g process= "Fit"
if(whichlistitem("SmoothOperator",winlist("*",";","Win:64"))!=-1)
	killwindow SmoothOperator
elseif(whichlistitem("FitManager",winlist("*",";","Win:64"))!=-1)
	killwindow Fitmanager
endif

end
```

```
function showcd(ctrlname,checked) : checkboxcontrol
        string ctrlname
        variable checked
        variable/g ender=0
        if (checked==1)
                showinfo/cp=1/w=SmoothingGraph
        else
                showinfo/cp=0/w=SmoothingGraph


        endif
end
```

A.3.6 Normalizing FORC Data

```
#pragma rtGlobals=3          // Use modern global access method and
   strict wave access.


function OsciForcNorm()
// It seems like the waves are pretty well "normalized" withought
   having todivide them by different factors. I think a simple
// offset will be good enough to make an accurate first graph. Ideally
    I want to just have the option of doing one or the other
//Need a selector early, then just an if here and there to include
   certain calculations.
//select a wave
//for each point, check to see if each of the 500 points following (
   with 10 possible booboos?)
//are monatonically increasing
variable v_value
make/N=1/o ReversalFieldValues
make/n=1/o ReversalIndexedLocation
variable g=0
variable/g OldIndex = 0
variable/g root:lazy
nVar lazy = root:lazy
//We can run this with different ranges of reversals, but we intend to
    always have them normalized to the most outer one. I think we can
    just have
//cutter and sorter que up all the averages to use. Might not have to
   change this one.

//I would like to preserve the Full Waves, not just reversals, so i
   can see the decreasing field behaviour


Execute "CreateBrowser prompt=\"Find the folder with Magnetic FIeld
   Waves\", showWaves=1, showVars=0, showStrs=0" //asks you to find a
    wave, who'se path is stored as S_browserlist
```

```
SVAR S_BrowserList=S_BrowserList //no clue why this is here
variable pathlength =strlen(s_browserlist) //length, in numbers, of
   the path name.
string FieldLocationA= s_browserlist[0,pathlength-2] //cuts
   thesemicolon from the path name

Execute "CreateBrowser prompt=\"Find the folder with Kerr Rotations\",
    showWaves=1, showVars=0, showStrs=0" //asks you to find a wave,
   who'se path is stored as S_browserlist
SVAR S_BrowserList=S_BrowserList //no clue why this is here
variable pathlength2 =strlen(s_browserlist) //length, in numbers, of
   the path name.
string waveLocationB= s_browserlist[0,pathlength2-2] //cuts
   thesemicolon from the path name

if(datafolderexists("root:NormalizedReversals")==0)
       newdatafolder Root:NormalizedReversals
endif

setdatafolder \$(FieldLocationA)
string FieldListA=wavelist("*",";","")
setdatafolder Root:
setdatafolder \$(wavelocationB)
string waveListB=wavelist("*",";","")
setdatafolder Root:

Do
       //Wave testwave = root:A_waves:R_2
       string FieldName=FieldlocationA+":"+stringfromlist(g,
          FieldListA)
       string Kerrname=wavelocationB+":"+stringfromlist(g,waveListB)

       if(strlen(stringfromlist(g,FieldlistA))==0)
              break
       endif
       Wave KerrWave= \$kerrname
       Wave Fieldwave = \$fieldname
       variable i=0
       variable p=0
       variable booboos=0
       variable success=0
       variable stepper=0
       variable maxbooboos=3

       DO

              Do
                     variable value1=fieldwave[stepper+i]

                     If(stepper== (numpnts(fieldwave)-40))

                            Display/w=(300,100,1600,1000)/N=Thegraph
                               fieldwave
```

```
                dowindow/f thegraph
                showinfo/cp=0/w=thegraph
                NewPanel /K=1 /W=(187,368,437,531) as "
                   Pause for Cursor"
                DoWindow/C NoMinHere
                          // Set to an unlikely name
                AutoPositionWindow/E/M=1/R=thegraph
                                // Put panel near the
                   graph
                DrawText 21,20,"Adjust the cursors and
                   then"
                DrawText 21,40,"Click Continue."
                Button button0,pos={80,58},size={160,20},
                   title="Use Cursor Index as min"
                Button button0,proc=CursorControlA
                Button button1,pos={80,90},size={160,20}
                Button button1,proc=Endthething2,title="
                   End"
                Button button2,pos={80,110},size={160,20}
                Button button2,proc=LazyMin,title="Use
                   Previous Min Index"
                dowindow/f thegraph
                PauseForuser NoMinHere,Thegraph

                if(lazy==0)
                stepper=pcsr(A,"Thegraph")
                elseif(lazy==1)
                stepper = oldIndex
                endif
                booboos=0
                killwindow thegraph
                Break

        endif

        variable value2=fieldwave[stepper+1+i]
        if(value2>=value1)
                success+=1
        else
                //print i+stepper
                booboos+=1
        endif

        if(booboos==maxbooboos)
                break
        endif
        i+=1
while(i<=1500 )

if(booboos<maxbooboos)
        break
endif

stepper+=1
```

```
                booboos=0
                success=0
                i=0
        while(1)


        print "the start of monoatonic increase for wave "+nameofwave(
           fieldwave)+" was " + num2str(stepper)
        variable minfield = fieldwave[stepper]
        ReversalFieldValues[g] = {minfield}
        ReversalIndexedLocation[g]={stepper}
        print "the field value at the min point is " + num2str(
           minfield)

        g+=1

while(1)
string ScaleOrShift
Prompt Scaleorshift, "offset and norm?",Popup "NormAndOffset;Offset;"
Doprompt "Will you offset this and Norm or just offset?", scaleorshift
        g=0
        i=0
        setdatafolder root:
        //What do i need?
        //Need the kerr at Hr, and the Kerr at Hmax for each reversal
           and the outer one.
        //easy enough, lets find the outer one.
Do
        If(i==0)
                FieldName=FieldlocationA+":"+stringfromlist(0,
                   FieldListA)
                wave outerfieldwave=\$(fieldname)
                string outerfieldreversalName = "FieldR_1"
                string OuterKerrname=wavelocationB+":"+stringfromlist
                   (0,waveListB) //sums the 10 points on either side of
                    the "min value" and averages them
                wave outerkerrwave=\$outerkerrname
                variable SmallestKerr=sum(outerkerrwave,
                   reversalindexedlocation[i]-12,
                   reversalindexedlocation[i]+12)/25
                //print SmallestKerr
                variable totalPoints=numpnts(outerkerrwave)
                variable MaxestKerr=sum(outerkerrwave,totalpoints-50,
                   totalpoints-1)/50 //these two now yield the largest
                   and smallest kerr rotation values
                //print maxestkerr                  ///
                                                The largest is imprtant
                    for anchoring, and the smallest is necessary to
                   make the normalizing factor

                variable NormFactor=(maxestkerr-smallestkerr)/2
                variable offset = (maxestkerr+smallestkerr)/2
```

```
        Duplicate/o/R=[reversalindexedlocation[i],numpnts(
           outerfieldwave)-1] outerfieldwave, \$(
           outerfieldreversalname)
        Duplicate/o outerkerrwave, \$("FULL_N_"+nameofwave(
           outerkerrwave))

        wave normedouter=\$("Full_N_"+nameofwave(outerkerrwave)
           )                                  // now the outer wave
            is sittin pretty and normalized.

        If(stringmatch(Scaleorshift, "Normandoffset")==1)
               normedouter=(normedouter-offset)/normfactor
                  //9/30/19
        else
               normedouter=(normedouter-offset) //If I am just
                  offsetting stuff, I don't need to normalize
                  it to 1, but centering it on the Y axis is "
                  useful"
        endif

        Duplicate/o/R=[reversalindexedlocation[i],numpnts(
           outerfieldwave)-1] Normedouter, \$("N_"+nameofwave(
           outerkerrwave))
        Display Normedouter vs Outerfieldwave
        doupdate
        i+=1
elseif(i!=0)
        variable rng1=Abs(floor(enoise(65534))) //make 3 random
            r-g-b's for graphs
        variable rng2=Abs(floor(enoise(65534)))
        variable rng3=Abs(floor(enoise(65534)))
        string outerfieldname=FieldlocationA+":"+stringfromlist
           (0,FieldListA)
        FieldName=FieldlocationA+":"+stringfromlist(i,
           FieldListA)
            if(strlen(stringfromlist(i,FieldListA))==0)
                   break
            endif

        kerrname=wavelocationB+":"+stringfromlist(i,waveListB)
        wave kerrwave=\$(kerrname)
        wave fieldwave =\$(fieldname)
        variable ReversalValue=sum(kerrwave,
           reversalindexedlocation[i],reversalindexedlocation[i
           ]+12)/13
        variable Maxvalue=sum(kerrwave,totalpoints-150,
           totalpoints-1)/150              //this is the value
           for the m(Hr) and m(Hmax)
        //stil need M(Hr) and M(Hmax)(which we have already as
           MaxestKerr)
        variable ReversalIndex = Reversalindexedlocation[i] //
           index where reversal happens for reversal field wave
```

```
variable ReversalfieldValue= fieldwave[Reversalindex]
      // the actual field value (Hr) of the reversal

If(stringmatch(Scaleorshift, "Normandoffset")==1)
       Duplicate/o/r=[0,numpnts(outerfieldwave)/2]
          Outerfieldwave, MinFinder   //all of this to
          find which kerr value is associated with the
          corresponding reversal field
       wave Minfinder
       Minfinder= (Minfinder-Reversalfieldvalue)
       wavetransform/o abs minfinder
       findvalue /v=(wavemin(minfinder)) minfinder //
          which index of the outer wave does the field
          reach Hr
       print "the reversal field of "+num2str(
          reversalfieldvalue)+" occurs at point " +
          num2str(v_value)+ " of the outer kerr wave"
       variable OuterreversalKerr=sum(outerkerrwave,(
          v_value)-3,v_value+3)/7//what kerr value is
          the outerloop at when we have Hr
//we have now M(Hr), M(Hmax), m(Hr), and m(Hmax)
       Variable IdealDelta = (MaxestKerr-
          OuterReversalKerr)
       Variable TrueDelta = (maxvalue-ReversalValue)
       Variable ScalingFactor=IdealDelta/TrueDelta
                                                    //
          all these variables go into normalizing. see
          notebook3, pgs 6-9
       variable ScaledMax = scalingfactor*maxvalue
       variable scaledMin= scalingFactor*ReversalValue
       variable Reversaloffset=((Scaledmax-maxestkerr)
          +(scaledmin-OuterReversalKerr))/2
else
       reversaloffset=Maxvalue-maxestkerr
       scalingFactor=1
       Normfactor=1
endif
string NormalizedReversal="N_"+nameofwave(kerrwave)
string ReversalFieldname="Field"+nameofwave(Fieldwave)
Duplicate/o Kerrwave, \$("FULL"+NormalizedReversal)
Duplicate/o/R=[reversalindexedlocation[i],numpnts(
   kerrwave)-1] Kerrwave, \$(NormalizedReversal) //
   Right here makes the normalized wave have just the
   reversal part of the loop
Duplicate/o/R=[reversalindexedlocation[i],numpnts(
   kerrwave)-1] fieldwave, \$(reversalfieldname)
wave Normalizedreversalwave= \$(NormalizedReversal)
Wave normailzedFullForc=\$("FULL"+NormalizedReversal)
Normalizedreversalwave=((((normalizedreversalwave*
   ScalingFactor)-Reversaloffset)-offset)/Normfactor)
normailzedFullForc=((((normailzedFullForc*ScalingFactor
   )-Reversaloffset)-offset)/Normfactor)
```

```
                //Normalizedreversalwave=((normalizedreversalwave*
                   ScalingFactor)-Reversaloffset)
                 appendtograph/C=(rng1,rng2,rng3)
                    Normalizedreversalwave vs \$(reversalfieldname)
                 doupdate
                i+=1
        endif
While(1)
        //Now, if we are on any other loop we need Kerr(Hr) for that
           loop and it's max value, similar to what we have just done
           above.
        //The tricky comes when finding the value of Kerr(Hr_outerloop
           ) needs to be found by finding the value where the field is
            closest to the Hr form the minor loop.
        //and we'll do the normalizing for each wave here
        string listofKerr=wavelist("*N_R*",";","")
        string listofFields=wavelist("*FieldR*",";","")
        variable k=0
DO
fieldname=stringfromlist(k,listofkerr)
kerrname=stringfromlist(k,listoffields)
        if(strlen(fieldname)==0)
                break
        endif

wave kerrwave =\$kerrname
wave fieldwave=\$fieldname
movewave kerrwave, Root:NormalizedReversals:
movewave fieldwave, Root:Normalizedreversals:
k+=1
while(1)
killwaves minfinder
end


Function CursorControlA(CtrlName) : ButtonControl
string ctrlName
//variable/G Ender=1
variable/g oldindex = pcsr(a,"TheGraph")
variable/g Lazy=0
killwindow NoMinHere
end

Function LazyMin(CtrlName) : ButtonControl
string ctrlName
variable/g Lazy =1
killwindow NoMinHere
end
```

A.4 FORC Analysis Programs

From a normalized set of curves, we must actually take derivatives to achieve the forc diagram. The first is done with the FirstD() program which takes a derivative of the reversal with respect to the applied field. This program prompts the user with a panel which allows for the individual selection of reversals from a dropdown menu. As shown in figure 7.45, the user can select a number of points on either side of the point being differentiated.

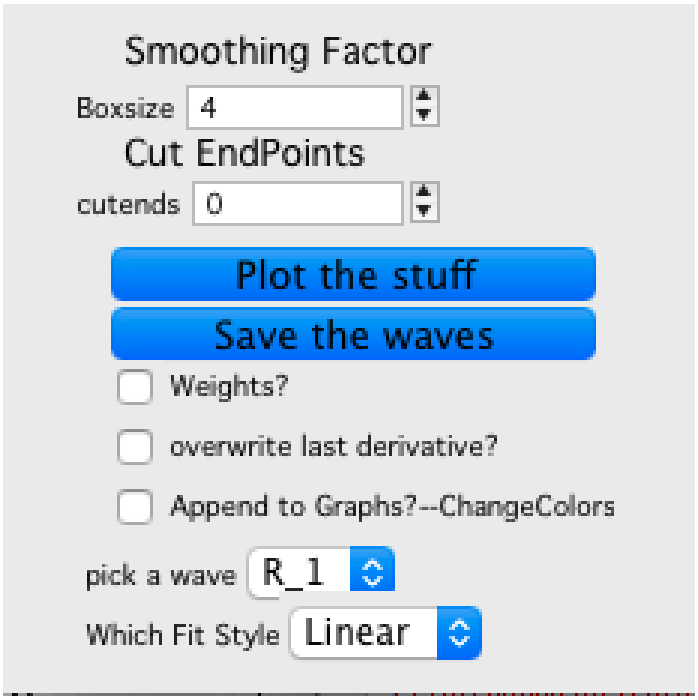

Figure A.2 An example of the panel prompted when taking the first derivative of raw data.

Confirming the selection produces a graph which is a derivative, a graph of the integral of that derivative, and a graph showing the difference between the integral and the orginal reversal. This step is done for each reversal by hand to ensure the best looking derivitive for each reversal. Following the first derivitive, we run SecondDPanel(), which, once again, will prompt the user with a panel with which they can interact.

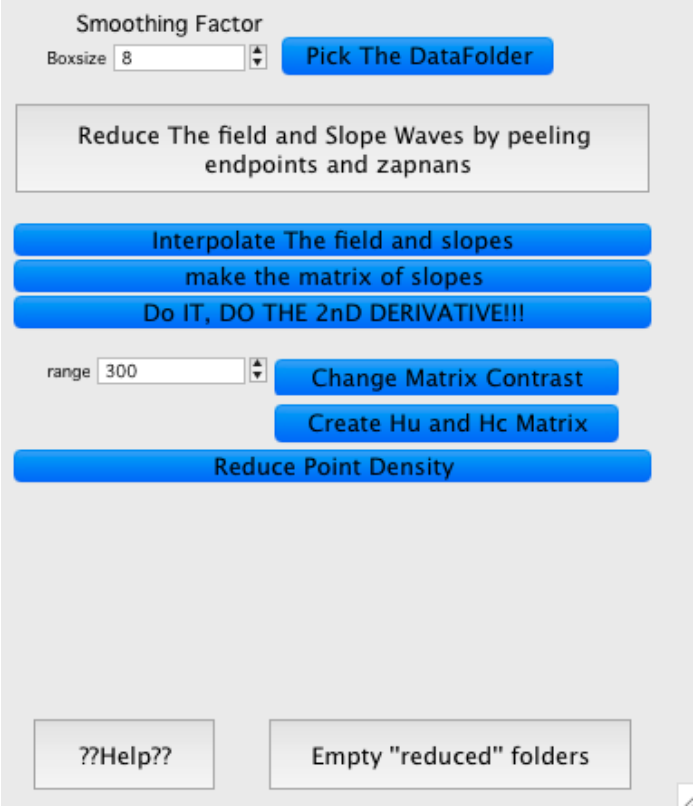

Figure A.3 An example of the panel displayed when taking the second derivative of our raw data for FORC.

Just like the FORC preparation, the second derivative panel also requires a workflow. From the top, the first input allows the user to select a window size (for differentiating) and data folder. The second button from the top will remove all NaN values if they are remaining as artifacts from the first derivative and cut all values beyond a predetermined field value, chosen within the program itself. Cutting endpoints helps to remove noisy behavior from excessive points near saturation. "Interpolate the field and slopes" will ask the user to pick a reduced amount of points to fix the field spacing, and interpolate all reversals to only have values at the same field points so that we can remove field as an axis as shown in figure 7.46. We remove field as an axis and create a matrix representation of our array of first derivatives, which is the next button. This version of Igorpro (v6.38B01) is unable to have unevenly spaced matrix indices so we insisted in the previous step that

they are identical. We now use all the values from the first derivatives of the reversals to build a matrix with axes H_r and H_a. From this, we proceed to the next button, which takes the derivative $\frac{\partial}{\partial H_a}$ which fits weighted linear fits along each of the equalized field values on the matrix (along the H_r direction). This will yield a matrix which is defined as a FORC diagram. We finally use the "create H_u and H_c matrix to make these coordinates for every point on the matrix so we can properly plot our FORC diagram with the literature-suggested axes, as mentioned and shown in equation 7.19.

A.4.1 First Derivative Panel

```
#pragma rtGlobals=3          // Use modern global access method and
   strict wave access.
function FirstD()
setdatafolder Root:
variable/g Boxsize =4
variable/g CutEnds =0
variable/g Ender=0
variable/g doweights=0
string/g FieldPathName
string/g Fitype = "Linear"
variable Edge=0 //-1,0,1 for left middle right
variable i=0
string theonewindow=winlist("Buttonpanelname",";","")
if(whichlistitem("Buttonpanelname", theonewindow)!=-1)
       killwindow Buttonpanelname
endif
       Nvar Boxsize=root:Boxsize
       Nvar cutends=root:cutends
       Execute "CreateBrowser prompt=\"find the folder full of waves
          to be fit (y-Waves)!\", showWaves=0, showVars=0, showStrs
          =0" //asks you to find a wave, who'se path is stored as
          S_browserlist
       SVAR S_BrowserList=S_BrowserList //sets a global variable
       variable pathlength =strlen(s_browserlist) //length, in
          numbers, of the path name.
       string foldername= s_browserlist[0,pathlength-2] //cuts the
          semicolon from the path name
       Execute "CreateBrowser prompt=\"find the folder full of Fields
           (X axis)\", showWaves=0, showVars=0, showStrs=0" //asks
          you to find a wave, who'se path is stored as S_browserlist
       SVAR S_BrowserList=S_BrowserList //sets a global variable
       variable pathlength2 =strlen(s_browserlist) //length, in
          numbers, of the path name.
       string foldername2= s_browserlist[0,pathlength2-2] //cuts the
          semicolon from the path name
```

```
        string/g pathtoKerr = foldername+":"
        string/g pathtoField = foldername2+":"
        svar pathtokerr = pathtokerr
        svar pathtofield=pathtofield
//      setdatafolder $foldername2 CZN changed 2/18/20
        setdatafolder $pathtofield //changed from the above line
        string/g listofFields= wavelist("*field*",";","") //this might
            be a problem? pre norm they have the same name, else i
           think i append "Field" as a prefix
        svar listofFields=listofFields
        setdatafolder root:
        string/g GListoffields = listoffields//I don't think this
           works
        //later, make a button that opens the data browser and lets me
            pick a new folder for both of these ^^^^ up there

NewPanel /W=(150,50,440,280)
DoWindow/C Buttonpanelname// set to an unlikely name

DrawText 43,57,"Cut EndPoints"
SetVariable setvar1,pos={27,59},size={126,17},limits={-Inf,Inf,1}
SetVariable setvar1,value= root:cutends
DrawText 43,23,"Smoothing Factor"
SetVariable setvar0,pos={27,27},size={126,17},limits={-Inf,Inf,1}
SetVariable setvar0,value= root:Boxsize
Button button0,pos={40,80},size={160,20}
Button button0,proc=Plotwaves ,title="Plot the stuff"
Button button1,pos={40,100},size={160,20}
Button button1,proc=SaveWaves,title="Save the waves"
Checkbox Check0,pos={40,120}
checkbox check0, Proc=weighting,title="Weights?"
Checkbox Check1,pos={40,140}
checkbox check1, Proc=Overwrite,title="overwrite last derivative?"
Checkbox Check2,pos={40,160}
checkbox check2, Proc=append,title="Append to Graphs?--ChangeColors"

setdatafolder $(foldername) //I dont like this, but unless i come up
   with another way to construct the list of waves, im sol.
//string listofKerrs = wavelist("!*field*",";","")
//string SortedKerrs=Sortlist(listofKerrs,";")
popupmenu popup1,pos={30,180},size={150,20}, value=WaveList("!*field
   *",";","")
//popupmenu popup1,pos={30,180},size={150,20}, value=SortedKerrs
popupmenu popup1,proc=Pickwave, title="pick a wave"
popupmenu popup2,pos={30,200},size={150,20}, value="Linear;Polynomial"
popupmenu popup2,proc=Fitstyle, title="Which Fit Style"

end


Function/S PickWave(ctrlName,popNum,popstr) : PopupMenuControl
        String ctrlName
        variable popnum
```

```
        string popstr
        string/g pathtokerr
        string /g pathtofield
        string /g listofFields
        variable/g cutends
        svar pathtokerr =root:pathtokerr
        svar pathtofield = root:pathtofield

        svar glistoffields =root:glistoffields

        string KerrName
        kerrname =nameofwave($(pathtokerr+popstr)) //when we select
           the kerr rotation, we also need the corresponding X wave.
        //If i had time to fuck with this, I would have a check box to
            optionally let you just pick the X wave from a dropdown
           menu...
        print kerrname
        variable StartTheName=strsearch(kerrname,"R_",0) // attempts
           to pull out when R_ starts in the string. now we have to
           look at R_XXX and descriminate
        string NametoMatch= kerrname[Startthename,strlen(kerrname)-1]

        print "the name to match is " + nametomatch
               variable i=0
               DO //NEED TO LOOK IN THE WAVE FOLDER AND Try to find
                  the corresponding X wave (field wave) based on the
                  nametomatch

                      string listitem = stringfromlist(i,glistoffields
                         )
                      //if the ideal match string matches the simlar
                         part---> the end of the name of the x wave,
                         they match???
                          if(stringmatch(listitem,"*"+nametomatch)
                             ==1)
                               print "The matching field is " +
                                  nameofwave($listitem)
                               break
                          else
                               i+=1
                          endif

               WHILE(1)
        string/g fullkerrname = pathtokerr+popstr //full path to the
           wave that we are looking at
        string/g fullfieldname = pathtofield+listitem
        svar fullfieldname =fullfieldname
        svar fullkerrname =fullkerrname
        print "the field is at " +pathtofield+listitem
        print "the kerr is at "+pathtokerr+popstr

end

Function/S Fitstyle(ctrlName,popNum,popstr) : PopupMenuControl
```

```
        String ctrlName
        variable popnum
        string popstr
        string/g root:Fitype
        svar fitype=root:fitype
if(stringmatch(popstr,"Linear")==1)
        fitype="Linear"
elseif(stringmatch(popstr,"Polynomial")==1)
        fitype="Polynomial"
endif

end



Function Plotwaves(ctrlname) : Buttoncontrol //Within this wave, we
   will calculate slopes and plot them
        string ctrlName
        string/g fullfieldname
        string/g fullkerrname
        string/g fitype
        nvar  boxsize = root:Boxsize
        nvar weightvalue = root:doweights
        svar fitype=root:fitype
        variable i=0
        variable width = boxsize
        variable totalpoints = numpnts($(fullkerrname))
        Variable WidthR= ceil(Width/2)
        Variable WidthL = ceil(Width/2)
        variable edge=0

        Nvar cutends = root:cutends


        If(datafolderexists("Root:FirstDerivative")!=1)
                newdatafolder Root:FirstDerivative
        endif
        if(weightvalue==0)
                String DsName="Root:FirstDerivative:D1_"+num2str(Width)
                   +"pts_"+Nameofwave($fullkerrname) //prefix as is so
                   that when the actual wave is made, it is put into
                   that folder
        else
                DsName="Root:FirstDerivative:D1_W_"+num2str(Width)+"
                   pts_"+Nameofwave($fullkerrname) //prefix as is so
                   that when the actual wave is made, it is put into
                   that folder

        endif
        Make/o/n=(totalpoints) $DsName=nan //makes a wave with above
           prefix (D1, indicating the first derivative) into the
           folder differentiating, for easier cleanup later
        Wave DsWave = $DsName
```

```
        Wave TheWave=$fullkerrname //the original wave that will be
           differentiated
        wave THeFieldWave = $fullfieldname
        //print width
Do
//Make/o/n=((2*Width)+1) NewFittywave //this is the wave where the to-
   be-fit-by-line points will be and this will be overwritten every
   interation


        if(i<widthR&&i<totalpoints/2)
                  //if there are not enough points to the LEFT of the
           point we are looking at, start at 0

                Duplicate/o/r=[0,i+widthR] Thewave Realfittywave
                Duplicate/o/r=[0,i+widthR] TheFieldwave RealfittyField
                edge=-1
        elseif(i+cutends<widthR&&i>totalpoints/2)
                           //if there are not enough points to the
           Right of the point that we are looking , end at the last
           point


                Duplicate/o/r=[i-widthL, totalpoints-1] Thewave
                   Realfittywave
                Duplicate/o/r=[i-widthL, totalpoints-1] TheFieldwave
                   RealfittyField
                edge=1
        else

                edge=0
                Duplicate/o/r=[i-widthL,i+widthR] Thewave Realfittywave
                Duplicate/o/r=[i-widthL,i+widthR] TheFieldwave
                   RealfittyField
        endif



        wave NewFittywave= Realfittywave//Wave variable within the
           code containing the to be fit potion of points
        wave newfittyField = Realfittyfield //x axis

If(stringmatch(fitype,"Linear")==1)
        if(weightValue==0)
                Curvefit/N/Q line newfittywave /x=newfittyfield
        elseif(weightValue==1)
                //middle point should just always be (numpnts(
                   realfittywave)-1)/2 since numpnts is always odd
                //eg. if there are 5 points, than middle index (0 1 2 3
                    4) is 2, (5-1)/2=2
                //first index is always 0.
                variable n=0 //index for finding radii
                variable Middleindex=(numpnts(realfittywave)-1)/2
```

```
        Variable R
        if(edge==-1) //use unweighted becasue weird
           edgebehaviour
               variable middley=newfittywave[0] //if we are
                  leftbound
               variable middlex=newfittyfield[0]

        elseif(edge==1)
                middley=newfittywave[numpnts(realfittywave)-1]
                   //if we are rightbound
                middlex=newfittyfield[numpnts(realfittywave)-1]
        elseif(edge==0)
                middley=newfittywave[middleindex] //if we are
                   in the middle
                middlex=newfittyfield[middleindex]
        endif
        variable Rmin,Rmax, Stdev

        Do //find the R values from each point to the middle
               if(edge==-1) //suspected messy edgebehaviour
                  messing up integration, use no weight at edge

                       break
               endif

//R=(((newfittywave[n]-middley)^2)+((newfittyfield[n]-middlex)
   ^2))^1/2
R=((newfittyfield[n]-middlex)^2)^1/2 //just the abs value of x
    distances
               if(n==0)
                      make/o/n=(0) root:RadialDistances //have
                         this made somewhere nice.....
                      wave radialdistances = root:
                         radialdistances
                      radialdistances[0]={R}
               //     display RadialDistances
                      //doupdate
               else
                      radialdistances[n]={R}
                      //doupdate
               endif

        n+=1

        while(n<numpnts(realfittywave))

        if(edge!=-1)
               duplicate/o newfittywave, weights
               weights=0
               Rmin = wavemin(radialdistances)
               Rmax=wavemax(radialdistances)
               stdev=Sqrt((-(Rmax-Rmin)^2)/(2*ln(.1))) //using
                  the distribution of radial distances to
```

```
                        create a resized gaussian.
                            //the farthest point is assigned to the
                               location of the gaussian where we have
                                10% of the total value
                        n=0

                        do
                                Weights[n]=Exp((-(RadialDistances[n]-Rmin
                                   )^2)/(2*(stdev^2)))
                                n+=1
                        while(n<numpnts(radialdistances))

                                Curvefit/N/Q line newfittywave /x=
                                   newfittyfield/i=0/w=weights

                else
                                Curvefit/N/Q line newfittywave /x=
                                   newfittyfield
                endif
        endif

        //if(i==floor(1302))
        //break
        //endif

wave w_coef
variable slope = w_coef[1] //variable slope for the linear fit

elseif(stringmatch(fitype,"Polynomial")==1)
        if(weightValue==0)
                Curvefit/N/Q poly_Xoffset 3, newfittywave /x=
                   newfittyfield //polyfit to the subset of the wave.
                //Need to take "differentiate" of the wave at the point
                     of interest. Since the curvefit uses 200? points by
                     default, i have to locate the right
                //point within the differentiated wave and save that
                   data to the
        elseif(weightValue==1)
        endif
        // slope= //the dot thing from the differentiate funtion
endif

        //wave w_coef

        Dswave[i]=slope
i+=1

while(i<(totalpoints-cutends))

//gotta make evevrything append to the same graph
//killwaves realfittyfield, realfittywave w_coef//,w_sigma
dowindow /k Derivative
display/N=Derivative/w=(0,0,1000,250) Dswave vs thefieldwave
```

```
//gotta get the integrated wave to be offset to be ontop of the
   original wave so i dion't have to drag it manually

setaxis bottom *,Wavemax(thefieldwave)*.95 //so the graph doesn't get
   autoscalled to the awful end behavior
setaxis/a=2 left
string integralname= "I_"+nameofwave(dswave)
integrate/meth=1 Dswave/x=thefieldwave/d=$integralname
dowindow/k integral
wave integralwave=$integralname


display/n=integral/w=(0,279,1000,750) $integralname vs thefieldwave
String visibleTraces=TraceNameList("",";",1+4) // only visible normal
   traces
string tracename = stringfromlist(0,visibletraces)
doupdate /w=integral
modifygraph rgb($tracename) =(0,0,65000)
appendtograph thewave vs thefieldwave
dowindow/k difference //kills trace on graph if it is there..i think
wave integratedwave = $integralname
duplicate/o integratedwave,Thedifference
duplicate/o/r=[0,numpnts(integratedwave)-1] thewave, reducedWave
thedifference= integratedwave-reducedwave
display/n=difference/w=(0,752,1000,1000) thedifference vs thefieldwave



end

function overwrite(ctrlname,checked) : checkboxcontrol
        string ctrlname
        variable checked
        if (checked==1)
                print "ka-check"
        else
                print "ka-uncheck"
        endif
end

function weighting(ctrlname,checked) : checkboxcontrol
        string ctrlname
        variable checked
        variable/g Doweights
        nvar doweights= root:doweights
        if (checked==1)
                print "ka-check, Radial Gaussian Weighting is enabled!"
                Doweights=1
        else
                print "ka-uncheck, Radial Gaussian Weighting is
                   Disabled"
                Doweights=0
        endif
```

```
end

Function Savewaves(ctrlname) : Buttoncontrol //Save the stuff that is
   currently Displayed into a folder
       string ctrlName
       string/g fullfieldname
       string/g fullkerrname
       string/g fitype
       string nameofgraph ="Derivative"

       if(datafolderexists("Root:DerivativesForForcAnalysis")==0)
              Newdatafolder/o Root:DerivativesForForcAnalysis
       endif

       string Tracename=Tracenamelist(nameofgraph,";",1+4)
       tracename=tracename[0,strlen(tracename)-2]

       wave wavepath=Tracenametowaveref(nameofgraph,Tracename)
       string fullpath =Getwavesdatafolder(wavepath, 2)


       Duplicate/o $fullpath, root:DerivativesForForcAnalysis:
          $tracename //at this point we will have thigs saving, but
          they might not necissarily be a nice naming


       Print tracename + " Has been saved"//convention, so we can
          mess with that later.
end
//
```

A.4.2 Second Derivative Panel

```
#pragma rtGlobals=3          // Use modern global access method and
   strict wave access.


Function SecondDPanel()
setdatafolder root:
string listoWindows= winlist("secndDPanel",";","")
variable windowexists =strsearch(listowindows,"secndDPanel",0)
if(windowexists!=-1)
killwindow secndDPanel
endif
variable/g range= 300
variable/g Boxsize =4
NewPanel /W=(150,50,540,500)
DoWindow/C secndDPanel// set to an unlikely name
//DrawText 43,23,"Smoothing Factor"
//SetVariable setvar0,pos={27,49},size={126,17},limits={-Inf,Inf,1}
//SetVariable setvar0,value= root:Boxsize
Button button0,pos={10,60},size={350,50}
```

```
Button button0,proc=SecondDprep ,title="Reduce The field and Slope
   Waves by peeling\r endpoints and zapnans"
Button button1,pos={10,125},size={350,20}
Button button1,proc=Samesteps,title="Interpolate The field and slopes"

Button button3,pos={10,145},size={350,20}
Button button3,proc=matrixUp,title="make the matrix of slopes"

button button5, pos={10,165},size={350,20}
Button button5,proc=SecondDman, title = "Do IT, DO THE 2nD DERIVATIVE
   !!!"
DrawText 43,23,"Smoothing Factor"
SetVariable setvar0,pos={27,27},size={126,17},limits={-Inf,Inf,1}
SetVariable setvar0,value= root:Boxsize
button button6, pos={157,22},size={150,22}
Button button6,proc=Setdatafolderdude, title = "Pick The DataFolder"
DrawText 43,180,"(+/-)ContrastRange"
SetVariable setvar1,pos={27,200},size={126,17},limits={-Inf,Inf,1}
SetVariable setvar1,value= root:range
button button7, pos={153,200},size={190,22}
Button button7,proc=contrast, title = "Change Matrix Contrast"
button button8, pos={153,225},size={190,22}
Button button8,proc=changeaxestest, title = "Create Hu and Hc Matrix"
Button button9,pos={10,250},size ={350,20}
button button9, proc=ReducePointNumber, title="Reduce Point Density-
   Not functional"
Button button10,pos={10,270},size ={350,20}
button button10, proc=GizmoGussy, title="Gussy up the Gizmo"
Button button2,pos={150,400},size={200,40}
Button button2,proc=killwavez,title="Empty ''reduced'' folders"
button button4, pos={20,400},size={100,40}
Button button4,proc=HELPME, title = "??Help??"
//setdatafolder $(foldername) //I dont like this, but unless i come up
    with another way to construct the list of waves, im sol.
//popupmenu popup1,pos={30,180},size={150,20}, value=WaveList("!*field
   *",";","")
//popupmenu popup1,proc=Pickwave, title="pick a wave"
//Procedure "pickwave" is in th "Firstderivativefromslopes.ifp, which
   is executed with firstd()
//pickwave requires a path to a folder containinf field_xxxx and one
   containting non-fields
//The field path is the same as the path from firstd(), so we continue
    to use the Glistoffields global variable.
end

function secondDprep(ctrlname):buttonControl

string ctrlname
string/g Glistoffields // = this is the wavelist of fields from the
   Firstd() program
string/g pathtofield
svar pathtofield=root:pathtofield
//svar fieldlist=glistoffields
```

```
//need a path to the fields
//selecta folder with the slopes in it
Execute "CreateBrowser prompt=\"find the folder full of slopes (y-
   Waves)!\", showWaves=0, showVars=0, showStrs=0" //asks you to find
    a wave, who'se path is stored as S_browserlist
       SVAR S_BrowserList=S_BrowserList //sets a global variable
       variable pathlength =strlen(s_browserlist) //length, in
          numbers, of the path name.
       string foldername= s_browserlist[0,pathlength-2] //cuts the
          semicolon from the path name
       string/g path2slopes= foldername+":"
       setdatafolder $(path2slopes)
       string listofslopes = wavelist("!*Field*",";","")
       setdatafolder root:
Execute "CreateBrowser prompt=\"find the folder full of field waves (x
   -Waves)!\", showWaves=0, showVars=0, showStrs=0"
       SVAR S_BrowserList=S_BrowserList //sets a global variable
       pathlength =strlen(s_browserlist) //length, in numbers, of the
           path name.
       foldername= s_browserlist[0,pathlength-2] //cuts the semicolon
           from the path name
       string/g path2fields= foldername+":"
       setdatafolder $(path2fields)
       string fieldlist = wavelist("*field*",";","")
       setdatafolder root:
if(datafolderexists("Root:reducedfieldwaves")==0)
newdatafolder Root:reducedfieldwaves
endif
if(datafolderexists("Root:reducedslopes")==0)
newdatafolder Root:reducedslopes
endif



//At this point i think i am going to go through every slope wave and
   replace the points past 3.6 kG with nan, then zap em
//zapnans on all waves in slopes....count the number of points that
   you had to cut.
//insist that the new field wave will have the same number of points
   as the zapped wave
variable i=0
do
       variable p=0
       variable m=0
              if (strlen(stringfromlist(i,fieldlist))==0) //ends
                 program once we don't have more waves
                     break
              endif


       do //this loops looks like it looks through the name to see if
           it has "R_" in it, which indicates that it's part of the
          actual forc data.
```

```
        string fieldname=stringfromlist(i,fieldlist)
        variable StartTheName=strsearch(fieldname,"R_",0) //
           attempts to pull out when R_ starts in the string.
           now we have to look at R_XXX and descriminate
        if(strlen(fieldname)==0)

               variable Breaker=1 //secondary break condition,
                  beccause of nested loops
               break
        endif

        if(startthename==-1)
               i+=1
        endif

while (startthename==-1) //keep going while it doesn't match

if (breaker==1)
        break
endif

string FullpathtoFieldWave=pathtofield+stringfromlist(i,
   fieldlist) //we have moved i a bit from the previous loop,
   so now what is the wave taht is actually R_something
wave fieldwave=$(path2fields+stringfromlist(i,fieldlist)) //if
    there are extra waves in the folder that don't have
   r_something in them
fieldname=nameofwave(fieldwave)
string pathtoReducedWave ="Root:reducedfieldwaves:"+nameofwave
   (fieldwave)
duplicate/o fieldwave, $(pathtoreducedwave)
wave reducedfield=$(pathtoreducedwave)
        print "the name to match is "+ fieldname
        DO //NEED TO LOOK IN THE WAVE FOLDER AND Try to find
           the corresponding X wave (field wave) based on the
           nametomatch

               string NametoMatch= fieldname[Startthename,
                  strlen(fieldname)-1] //I believe this just
                  matches the R_XXX since all my waves are
                  Prefix_R_XXX

               string listitem = stringfromlist(m,listofslopes)
               //if the ideal match string matches the simlar
                  part---> the end of the name of the x wave,
                  they match???
                      if(stringmatch(listitem,"*"+nametomatch)
                         ==1) // now that we have a field
                         selected that has R_xxx , this loop
                         searches through the list of slopes
                         for a matching
                             print "The matching kerr is " +
                                listitem                  //
```

```
                                        suffix. If there are multiple,
                                         it will grab the first one.
                                    break
                            else
                                    m+=1
                            endif

                WHILE(1)


        string FullpathtoSlopeWave=path2slopes+listitem
        wave slopewave=$(fullpathtoslopewave)
        string pathtoReducedslope ="Root:reducedslopes:"+listitem
                                //delete the same index - worth of
           points so that both wave contain data UNTIL .36 kg
        duplicate/o slopewave, $(pathtoreducedslope)
        wave reducedslope=$(pathtoreducedslope)
        if (i==12)
                print num2str(i)
        endif
        do
                if(reducedfield[p]>=.36||stringmatch(num2str(
                   reducedfield[p]),"NaN")==1||stringmatch(num2str(
                   reducedfield[p]),"nan")==1)
                       reducedfield[p]=nan
                       reducedslope[p]=nan

                endif
        p+=1
        while(p<numpnts(reducedfield))
        wavetransform/o zapnans, reducedfield
        wavetransform/o zapnans, reducedslope
i+=1
while(1)
//Now, using the spacing variable, take each fieldwave and do the
   thing to it.
//idealfieldspacing
end




function samesteps(cooteroool) : Buttoncontrol
string cooteroool
variable i=0
Execute "CreateBrowser prompt=\"find the folder full of slopes (y-
   Waves)!\", showWaves=0, showVars=0, showStrs=0" //asks you to find
    a wave, who'se path is stored as S_browserlist
       SVAR S_BrowserList=S_BrowserList //sets a global variable
```

```
        variable pathlength =strlen(s_browserlist) //length, in
           numbers, of the path name.
        string foldername= s_browserlist[0,pathlength-2] //cuts the
           semicolon from the path name
        string/g path2slopes= foldername+":"
        setdatafolder $(path2slopes)
        string slopelist= wavelist("!*Field*",";","")
        setdatafolder root:
Execute "CreateBrowser prompt=\"find the folder full of field waves (x
   -Waves)!\", showWaves=0, showVars=0, showStrs=0"
        SVAR S_BrowserList=S_BrowserList //sets a global variable
        pathlength =strlen(s_browserlist) //length, in numbers, of the
            path name.
        foldername= s_browserlist[0,pathlength-2] //cuts the semicolon
            from the path name
        string/g path2fields= foldername+":"
        setdatafolder $(path2fields)
        string fieldlist = wavelist("*field*",";","")
        setdatafolder root:



string firstwavename=stringfromlist(0,fieldlist)
variable NumberofSteps = numpnts($(path2fields+firstwavename))
        variable idealfieldspacing
Prompt IdealFieldSpacing, "Your first field wave has "+num2str(
   numberofsteps)+ " points in it, how many equally spaced steps
   would you like?"
Doprompt "Select Field Spacing",idealfieldspacing

string destfolder= "root:equalizedslopes"+num2str(idealfieldspacing)
newdatafolder/o $(destfolder)
//idealfieldspacing =2500
wave firstwave=$(path2fields+firstwavename)

//user is prompted to input number of points desired for spacing, we
   then use the min and max of the firstwave's fieldwave to set the
   ideal steps

Make/o/n=(idealfieldspacing) EFS //(maxfieldsteps)
EFS=wavemin(firstwave)+((wavemax(firstwave)+abs(wavemin(firstwave)))/
   idealfieldspacing)*x

variable stepsize =((wavemax(firstwave)+abs(wavemin(firstwave)))
variable newstart = 0
        Do
                //i=39
                variable j=0
                j=newstart
                string fieldname=stringfromlist(i,fieldlist)
                string kerrname=stringfromlist(i,slopelist)
                Print fieldname
                        if(strlen(fieldname)==0)
```

```
                break
        endif
    wave fieldwave = $(path2fields+fieldname)
    wave KerrSlope=$(path2slopes+kerrname)



        make/o/n=(numpnts(fieldwave)) Dummy=nan

    string EFSSpecific= "EFS"+ "_"+num2str(
       idealfieldspacing)+"points"
    string ESpacedName="E_"+kerrname
    duplicate/o EFS, $(espacedname)
    duplicate/o EFS, $(efsspecific)
    wave EqDkerr= $(espacedname)
    EqDKerr=kerrslope[0]

    variable pointdifference = idealfieldspacing-numpnts(
       fieldwave)
        //if lesss
        //elseif more
        //endif


//waves are :
//EFS : Equally spaced ideal field steps
//Dummy : when looking for closest point, use dummy as a filler.
//eqDkerr: when the interpolated value is found, makea new field wave
   with these steps.
//fieldwave: the actual wave being investigated
//kerrslope : the wave with the kerr slopes that match up with
   corresponding fieldwaves

    do


    if(EFS[j]<fieldwave[0])
        do
            j+=1

        while(EFS[j]<fieldwave[0])
        newstart=j
        //once it finds the start point on the first one
           , maybe save it so we don't have to count
           from 0 every time?
    elseif(efs[j]>fieldwave[numpnts(fieldwave)-1])
            //the last point of the new wave is the
               last point of the slope wave
    endif


    dummy = fieldwave-EFS[j]
    wavetransform/o abs, dummy
```

```
variable scaleFraction=.005 //startpoint for honing in
   on proper tolerance

do

        findvalue/t=(scalefraction*(stepsize))/v=(
           wavemin(dummy)) dummy //if we can't
           consistantly find these points, or if it is
           wrong, can make a loop where the tolerace
           slowly increases while checking for the point
           , compare to what we know the minpoint would
           be and if the index we find returns the same
           value
               // the closest value in EFS to the field
                  value is minwave(dummy) at point '
                  v_value'
               //Is this particular point closer to EFS
                  j-1 or j+1
               //print wavemin(dummy)
               //print dummy[v_value]

        scalefraction/=2 //makes tolerance smaller every
            time it gets the wrong point

while(wavemin(dummy)!=dummy[v_value])

        if(v_value!=numpnts(fieldwave)-1)
               variable InterpolationDifference    =
                  fieldwave[v_value]-EFS[j]

                   if(interpolationDifference<0)
                          variable endpoint=v_value
                             +1
                          variable startpoint=
                             v_value
                          variable endbound=
                             fieldwave[endpoint


                                    variable
                             startBound=fieldwave[
                             startpoint]
                   elseif(interpolationDifference>0)
                          startpoint=v_value-1
                          endpoint=v_value
                          endbound= fieldwave[
                             startpoint]
                          startBound=fieldwave[
                             startpoint]
                   endif

        //variable InterpolationPercent = (fieldwave[j]-
           startbound)/(endbound-startbound)
```

```
                        //at this point, now we can just plug into the
                           formula on pg 44 of my notebook :D
                               variable InterpolatedSlope = (((kerrslope
                                  [endpoint]-kerrslope[startpoint])/(
                                  fieldwave[endpoint]-fieldwave[
                                  startpoint]))*EFS[j])- (((kerrslope[
                                  endpoint]-kerrslope[startpoint])/(
                                  fieldwave[endpoint]-fieldwave[
                                  startpoint]))*fieldwave[startpoint])+
                                  kerrslope[startpoint]
                               EqDkerr[j]=interpolatedSlope

                        elseif(v_value>=numpnts(fieldwave)-1)

                               do
                                      EqDkerr[j]=kerrslope[numpnts(
                                         kerrslope)-1]
                                      j+=1
                               while(j<numpnts(EqDkerr)-1)

                        endif
               //print "the % that the fieldpoint is between idealpoints is "
                   + num2str(interpolationpercent)



                      j+=1
while(j<numpnts(efs))

               string pathtonewfolder=destfolder+":"+nameofwave(eqdkerr)
               duplicate/o eqdkerr,$Pathtonewfolder
               killwaves eqdkerr


i+=1
while(1)
end


function killwavez(ctrlname):buttonControl

string ctrlname

setdatafolder root:reducedfieldwaves
killwaves/a
setdatafolder root:reducedslopes
killwaves/a
setdatafolder root:
end

function matrixup(cooteroool) : Buttoncontrol
string cooteroool
```

```
 Execute "CreateBrowser prompt=\"find the folder full of slopes (y-
    Waves)!\", showWaves=0, showVars=0, showStrs=0" //asks you to
    find a wave, who'se path is stored as S_browserlist
       SVAR S_BrowserList=S_BrowserList //sets a global variable
       variable pathlength =strlen(s_browserlist) //length, in
          numbers, of the path name.
       string foldername= s_browserlist[0,pathlength-2] //cuts the
          semicolon from the path name
       string/g path2slopes= foldername+":"
       setdatafolder $(path2slopes)
       string listofslopes = wavelist("!*matrix",";","")

 variable i=0



string nameofslope= stringfromlist(i,listofslopes)
wave slopewave=$nameofslope
duplicate/o slopewave,finalmatrix
Concatenate/o listofslopes, finalMatrix
Newimage/k=1 FinalMatrix
ModifyImage finalMatrix ctab= {-10.089592,10,VioletOrangeYellow,0}
ColorScale/C/N=text0/A=RT image=finalMatrix
setdatafolder root:


end

function secondDman(ctrlname):buttonControl

string ctrlname
//variable/g boxsize
nvar boxsize = root:boxsize
print boxsize
//this is it folks.
//Need to grab that Ecks Axis. it's in root as Actual reversal Field
   Values. I think im just going to put a copy in a folder to be safe
   .
variable i=0
variable Row=0//use this for matrix row
variable m=0 //use this for matrix column index
variable widthR=ceil(boxsize/2)
variable widthL=ceil(boxsize/2)
wave thematrix = finalmatrix
variable edge=0
variable numberofcolumns=dimsize(thematrix,1) //number of reversals
variable Numberofrows= dimsize(thematrix,0)
wave RWave=root:ReversalFieldValues:ReversalEcksAxis //path to the
   reversal spacing wave. this will be our X axis for the derivative
//make/o/n=(dimsize(finalmatrix,1)) wavename here, dimsize(name,1)
   counts how many columns are in the matrix finalmatrix.
//wavename = finalmatrix[2188][p] this format will pull all of row
   2188 and asssign it to wave toot
```

```
//this should work pretty similarly to the other FirstD, which i may
   just cannibalize that code, weighting and all. nomnomnom.
//Should i do 1 line at a time, or just do the whole thing, and if
   need be, i can go back and switch the singluar weird stuff?
Duplicate/o thematrix, secondDmatrix
wave secondDmatrix
make/o/n=(dimsize(thematrix,1)) TheWave
wave thewave
do
i=0
        Do
                variable lookatme
                lookatme = thematrix[row][i]
        if(row==2212&&i==100)
                print "WAIT"
        endif
                        if(stringmatch(num2str(lookatme),"nan")||
                           stringmatch(num2str(lookatme),"NaN"))
                               print "sucka"
                               thematrix[row][i]=thematrix[row+1][i]
                        endif
                i+=1
        while(i<numberofcolumns)
row+=1
while(row<numberofrows)
row=0
i=0


Do
        thewave=thematrix[Row][p]
        i=0
        Do
        //Make/o/n=((2*Width)+1) NewFittywave //this is the wave where
            the to-be-fit-by-line points will be and this will be
           overwritten every interation


        if(i<widthR&&i<numberofcolumns/2)
                          //if there are not enough points to the LEFT
           of the point we are looking at, start at 0

                Duplicate/o/r=[0,i+widthR] Thewave Realfittywave
                Duplicate/o/r=[0,i+widthR] Rwave RealfittyField
                edge=-1
        elseif(i<widthR&&i>numberofcolumns/2)
                     //if there are not enough points to the Right of
           the point that we are looking , end at the last point


                Duplicate/o/r=[i-widthL, numberofcolumns-1] Thewave
                   Realfittywave
```

```
        Duplicate/o/r=[i-widthL, numberofcolumns-1] Rwave
           RealfittyField
        edge=1
else

        edge=0
        Duplicate/o/r=[i-widthL,i+widthR] Thewave Realfittywave
        Duplicate/o/r=[i-widthL,i+widthR] Rwave RealfittyField
endif



wave NewFittywave= Realfittywave//Wave variable within the
   code containing the to be fit potion of points
wave newfittyField = Realfittyfield //x axis



        //middle point should just always be (numpnts(
           realfittywave)-1)/2 since numpnts is always odd
        //eg. if there are 5 points, than middle index (0 1 2 3
            4) is 2, (5-1)/2=2
        //first index is always 0.
        variable n=0 //index for finding radii
        variable Middleindex=(numpnts(realfittywave)-1)/2
        Variable R

        if(edge==-1) //use unweighted becasue weird
           edgebehaviour
               variable middley=newfittywave[0] //if we are
                  leftbound
               variable middlex=newfittyfield[0]

        elseif(edge==1)
                middley=newfittywave[numpnts(realfittywave)-1]
                   //if we are rightbound
                middlex=newfittyfield[numpnts(realfittywave)-1]
        elseif(edge==0)
                middley=newfittywave[middleindex] //if we are
                   in the middle
                middlex=newfittyfield[middleindex]
        endif

        variable Rmin,Rmax, Stdev

        Do //find the R values from each point to the middle
               if(edge==-1) //suspected messy edgebehaviour
                  messing up integration, use no weight at edge

                      break
               endif

//R=(((newfittywave[n]-middley)^2)+((newfittyfield[n]-middlex)
```

```
   ^2))^1/2
R=((newfittyfield[n]-middlex)^2)^1/2
                if(n==0)
                        make/o/n=(0) root:RadialDistances //have
                           this made somewhere nice.....
                        wave radialdistances = root:
                           radialdistances
                        radialdistances[0]={R}
                //      display RadialDistances
                        //doupdate
                else
                        radialdistances[n]={R}
                        //doupdate
                endif

        n+=1

        while(n<numpnts(realfittywave))

        if(edge!=-1)
                duplicate/o newfittywave, weights
                weights=0
                Rmin = wavemin(radialdistances)
                Rmax=wavemax(radialdistances)
                stdev=Sqrt((-(Rmax-Rmin)^2)/(2*ln(.1))) //using
                   the distribution of radial distances to
                   create a resized gaussian.
                        //the farthest point is assigned to the
                           location of the gaussian where we have
                            1% of the total value
                n=0

                do
                        Weights[n]=Exp((-(RadialDistances[n]-Rmin
                           )^2)/(2*(stdev^2)))
                        n+=1
                while(n<numpnts(radialdistances))

                        Curvefit/N/Q line newfittywave /x=
                           newfittyfield/i=0/w=weights

        else
                        Curvefit/N/Q line newfittywave /x=
                           newfittyfield
        endif


wave w_coef
variable slope = w_coef[1] //variable slope for the linear fit


        //wave w_coef
```

```
        secondDmatrix[row][i]=slope
i+=1

while(i<(numberofcolumns))
print "row " + num2str(row) + " complete"
row+=1

while(row<Numberofrows)

end



function setdatafolderDude(cooteroool) : Buttoncontrol
string cooteroool

Execute "CreateBrowser prompt=\"find the folder full of slopes (y-
   Waves)!\", showWaves=0, showVars=0, showStrs=0" //asks you to find
    a wave, who'se path is stored as S_browserlist
       SVAR S_BrowserList=S_BrowserList //sets a global variable
       variable pathlength =strlen(s_browserlist) //length, in
          numbers, of the path name.
       string foldername= s_browserlist[0,pathlength-2] //cuts the
          semicolon from the path name
       string/g path2slopes= foldername+":"
       setdatafolder $(path2slopes)

end


function contrast(cooteroool) : Buttoncontrol
string cooteroool
//nvar contrast=root:range
string listofTraces =Imagenamelist("",";")
string TraceName = stringfromlist(0,listoftraces) //assumes only one
   trace on graph
wave tracepath = imagenametowaveref("", tracename)
//wave thewave =$tracepath
variable minuscontrast =wavemin(tracepath)
variable pluscontrast=wavemax(tracepath)
ModifyImage '' ctab= {minuscontrast,pluscontrast,RainbowCycle,1}
print minuscontrast
print pluscontrast

end



function changeaxestest(cooteroool) : Buttoncontrol
string cooteroool
wave reversals= root:ReversalFieldValues:ReversalEcksAxis
wave appliedfield =root:EFS_2500Points
variable columns= numpnts(reversals)
```

```
variable rows= numpnts(appliedfield)

make/o/n=(rows,columns) Hcs
make/o/n=(rows,columns) Hus
variable r=0
variable c=0
Do
r=0
        do
        Hcs[r][c]=(appliedfield[r]-reversals[c])/2 //h-Hr/2
        Hus[r][c]=(appliedfield[r]+reversals[c])/2 //H+Hr/2
        r+=1
        while(r<rows)
c+=1
while(c<columns)

end
//###################################################
function reducepointnumber(cooteroool) : Buttoncontrol
string cooteroool

//()Identfy waves that have matching suffixes
//() create new waves with different names
//() reduce the waves so that they have less dense data points


string path2reducedreversal
variable i=0
variable k=0
variable p=0
wave reversalfieldvalues = root:ReversalFieldValues:ReversalEcksAxis
   //this is made from the sas program, and should already contain
   all the revresal field values


variable rng1,rng2,rng3,multiplier
string ReducedFoldername

Prompt ReducedFoldername, "What do you want the folder name suffix to
   be for the reduced Waves"
DoPrompt "Reduced Folder Name", ReducedFoldername

Prompt Multiplier, "what multiple of the rebersal number do you want
   as the largest point number?"
DoPrompt "input Fraction",multiplier

curvefit/q line, reversalfieldvalues //produces a ...Hopefully, linear
    fit.
wave w_coef
variable Spacing = W_coef[1]/multiplier //w_coef[1] is the slope of
   the fit line
make/o/n=1 IdealSteps

Execute "CreateBrowser prompt=\"Find the folder with the Post-
```

```
   NormReversals\", showWaves=1, showVars=0, showStrs=0" //asks you
   to find a wave, who'se path is stored as S_browserlist
SVAR S_BrowserList=S_BrowserList //assigning the location of the
   global variable
variable pathlength =strlen(s_browserlist) //length, in numbers, of
   the path name.
string FieldLocation= s_browserlist[0,pathlength-2] //cuts
   thesemicolon from the path name

do
        variable onestep=reversalfieldvalues[0]+i*Spacing
        Idealsteps[i]={onestep}
        i+=1

while(i<numpnts(reversalfieldvalues)*multiplier)


path2reducedreversal="root:Normalized_"+num2str(i)+"steps"+
   reducedfoldername

if(datafolderexists(path2reducedreversal)==0)
        newdatafolder $(Path2reducedreversal)
endif

        setdatafolder $(fieldlocation)
        string listofFields=wavelist("*Field*",";","")
        string listofKerr=wavelist("*N_*",";","")
        setdatafolder root:

Do
        if(p==0)
                //Identify prefix. sometimes it will be S_N_" " and
                   sometimes it will just be N_ so we should have this
                   figure out
                //which it is.

                if(stringmatch(stringfromlist(0,ListofKerr),"*Full*")
                   ==1)
                        String Prefix = Stringfromlist(1,Listofkerr)
                else
                        Prefix = Stringfromlist(0,Listofkerr)
                endif

                Variable PrefixLocation= strsearch(prefix,"R_",1)
                Prefix=prefix[0,Prefixlocation+1]
                Print Prefix
        endif
        if(stringmatch("*FULL*",stringfromlist(p,listoffields))==1)
                p+=1
        endif
        string reducedfieldname="Reduced"+Stringfromlist(p,
           listoffields)
        Make/n=0/o $(reducedfieldname)
```

```
wave reducedwave=$(reducedfieldname)
string fieldname = stringfromlist(p,listoffields)



if (strlen(stringfromlist(p,listofFields))==0)
        break
endif

wave Fieldwave = $(fieldlocation+":"+fieldname)

variable startofR= strsearch(fieldname,"R_",1)
string searchstring= fieldname[startofR+2,strlen(fieldname)-1]
    //identifies the numerical suffix (index) of the wave

variable theindex=whichlistitem(Prefix+searchstring,listofkerr
   )

string kerrname=Prefix+searchstring
Wave Kerrwave=$(fieldlocation+":"+kerrname)
string reducedkerrname="Reduced"+Stringfromlist(theindex,
   listofkerr)
Make/n=0/o $(reducedKerrname)
wave reducedKerr=$(reducedKerrname)
k=0

Do


        if(k==0)
                variable localmin= wavemin(fieldwave)
                duplicate/o idealsteps, IdealstepMovingMin
                wave IdealstepmovingMin
                IdealstepmovingMin=idealsteps-localmin
                wavetransform/o abs idealstepmovingMin
                variable IdealMin=wavemin(idealstepmovingmin)
                Findvalue/V=(idealmin) idealstepmovingmin
                variable StartingINdex=v_value
        endif

                Duplicate/o fieldwave, Minfinder
                wave Minfinder
                Minfinder= Fieldwave-idealsteps[k+StartingIndex]
                wavetransform/o abs minfinder
                variable Minvalue= wavemin(minfinder)
                findvalue/v=(minvalue) minfinder
                reducedwave[k]={fieldwave[v_value]}
                reducedkerr[k]={kerrwave[v_value]}
                //Now i need to go to use these idnex values
                   found to pull the same indexed values from
                   the kerr wave
                k+=1
while(StartingINdex+k< numpnts(idealsteps))
```

```
        if(p==0)
                Display ReducedKerr vs ReducedWave
        else
                rng1=Abs(floor(enoise(65534)))
                rng2=Abs(floor(enoise(65534)))
                rng3=Abs(floor(enoise(65534)))
                appendtograph/c=(rng1,rng2,rng3) ReducedKerr vs
                   Reducedwave
                doupdate
        endif
        p+=1
While(1)
wave reduced
killwaves minfinder,Reduced, IdealstepMovingMin
end
```