UC San Diego

UC San Diego Electronic Theses and Dissertations

Title

Electromagnetic metamaterials : engineering the physics of light

Permalink

https://escholarship.org/uc/item/87d2v996

Author

Driscoll, Tom

Publication Date

2008

Peer reviewed|Thesis/dissertation

eScholarship.org Powered by the California Digital Library University of California

UNIVERSITY OF CALIFORNIA, SAN DIEGO

Electromagnetic Metamaterials:
engineering the physics of light.

A dissertation submitted in partial satisfaction of the
requirements for the degree of Doctor of Philosophy
in
Physics

by

Tom Driscoll

Committee in Charge

Dimitri N. Basov, Chair
Massimiliano Di Ventra
Shayan Mookerjea
Sia Nemat-Nasser
David R. Smith
Anthony Starr
Tom O'Neil

2008

Tom A. Driscoll

Copyright

2008

All rights reserved

The dissertation of Tom Driscoll is approved,
and is acceptable in quality and form for publication
on microfilm

____

____

____

____

____

____

____
Chair

Dedications

To my family, for all your support, thank you. To my father, who encouraged and nurtured my curiosity about the workings of nature from the beginning. To my mother, who taught me love and life which exist outside the reach of science. To my brother, who I am proud of. And to Megan, my love, and our many adventures yet to come.

Epigraph

"The goal of science is to amass such an enormous mountain of evidence that not even scientists can ignore it, and this is the distinguishing feature of a scientist; a non-scientist will ignore it anyway."

-Eliezer Yudkowsky

"In the life of every mad scientist, there comes a point between sane and mad when he is only half-mad. This is when he will throw his best parties."

-Colin W. Jemmott

The real beauty of science lies not in raw truth or the thrill of discovery, but in the ability to improve lives. Science is from humanity, to humanity. In our quest for progress we must often work just as hard not to lose to sight of this.

Table of Contents

List of Figures

List of Tables

Vita

(2001) Bachelor of Science in Physics. Harvey Mudd College, CA

(2001 to Present) Stampy consulting. CTO & laser systems consultant.

(2006) Master of Science in Physics University of California San Diego

(2008) Doctor of Philosophy in Physics University of California San Diego

PUBLICATIONS

T. Driscoll, S. Palit, M.M. Qazilbash, M. Brehm, F. Keilman, B.G. Chae, H.T. Kin, N. Marie-Jokerst, D.R. Smith, D.N. Basov. *Hybrid metamaterials for dynamic tuning.* Applied Physics Letters **93**, 024101 (2008).

T. Driscoll, G. O. Andreev, D. N. Basov, S. Palit, Tong Ren, Jack Mock, Sang-Yeon Cho, Nan Marie Jokerst, and D. R. Smith, *Tuned permeability in terahertz split-ring resonators for devices and sensors.* Applied Physics Letters **90**, 092508 (2007)

T. Driscoll, G. O. Andreev, D. N. Basov, S. Palit, Tong Ren, Jack Mock, Sang-Yeon Cho, Nan Marie Jokerst, and D. R. Smith, *Quantitative investigation of a terahertz artificial magnetic resonance using oblique angle spectroscopy.* Applied Physics Letters **90**, 092508 (2007)

T. Driscoll, D. N. Basov, W. J. Padilla, J. J. Mock, and D. R. Smith, *Electromagnetic characterization of planar metamaterials by oblique angle spectroscopic measurements.* Physical Review B **75**, 115114 (2007)

T. Driscoll, D. N. Basov, A. F. Starr, P. M. Rye, S. Nemat-Nasser, D. Schurig, and D. R. Smith. *Free space microwave focusing by a negative index gradient lens*. Applied Physics Letters **88**, 081101. (2006)

Major Field: Physics.

Studies in infrared metamaterials
Professor Dimitri N. Basov

Studies in microwave metamaterials
Professor David R. Smith
Professor Sia Nemat-Nasser

Abstract of the dissertation

Electromagnetic Metamaterials
Engineering the Physics of light.

by
Tom A. Driscoll
Doctor of Philosophy in Physics
University of California, San Diego, 2008
Professor Dimitri N. Basov, Chair

Structures engineered to give a specific response to light are certainly nothing new. The long history of engineering materials response to light encompasses seemingly disparate structures from antennas to stained glass, lighting rods to mirrors. It is only in the recent decade, however, that we have appreciated the full gamut of possibilities this field holds, and envisioned paths towards realizing these possibilities. The new field of electromagnetic metamaterials has given us the potential to create devices that manipulate light in nearly any way we can envision. The work of this thesis is involved principally with the study of metamaterials and their unique properties. Using a wide array of developed apparatus and techniques - spanning microwave frequencies through the infrared – we investigate metamaterial behavior, and the ways they differ from conventional materials. Applications are always kept in the forefront of thought. The demonstration of a graded negative-index lens, fabricated from metamaterial fiberglass composite, highlights the potential of these structures. Characterization procedures and instruments suitable for metamaterial samples, developed in the course of this work, enable not only our investigation of the physics of metamaterials, but also facilitate the full design cycle critical to engineering. Our demonstration of dynamic tuning directly addresses the role bandwidth plays as a major roadblock to metamaterial devices. Finally a demonstrated novel use as a sensor/detector adds to the growing list of metamaterial roles in emerging technology.

I. Introduction to the dissertation

The field of metamaterials began as something of a curiosity. The physics of materials possessing negative refractive index had been worked out by the Russian physicist Veselago in the 1970's, but had never been realized. In 1999, after the initial work had been nearly forgotten, such a material with negative index of refraction was first demonstrated. This material was a metamaterial, utilizing sub-wavelength geometric metallic inclusions to create the necessary magnetic response. Although sub-wavelength arrays existed previously, it was this use to create a material with properties that did not exist in nature which drew intense attention. Since then, the field has grown explosively, with publications and funding following a near-geometric yearly increase. This marked interest is not without merit – the number of revolutionary metamaterial applications and devices demonstrated in half a decade is remarkable.

In chapter 2, we demonstrate one such metamaterial application; a graded-index microwave lens with negative refractive index throughout. The lens is fabricated using a newly pioneered technique wherein multiple Printed Circuit Boards (PCB's) are fused together to form a metamaterial composite. This composite approach creates a metamaterial with superior mechanical properties to previous methods – many of which often relied heavily on scotch tape. Many of the innovations necessary to complete the design of the lens are highlighted. The performance of this microwave lens, ascertained through experiments designed and built for its testing, is discussed in the context of traditional lenses and other metamaterials.

In chapter 3, we begin to discuss some of the experimental difficulties inherent in working with metamaterials. In particular, we look at issues relating to characterizing metamaterial samples. Metamaterials are designed using finite-element electromagnetic simulation codes that attempt to predict the response. Even so, the final sample quite often does not behave as predicted due to errors in fabrication or limitations in simulation capabilities – which makes experimental verification critical. Unfortunately several factors combine to make normal approaches to characterization inappropriate – namely the large periodicity, the presence of a magnetic (as well as electric) response, and the tendency towards ‘thin-film’ planar samples. We thus develop and discuss a new technique for characterizing metamaterials. This technique is applied on two metamaterial samples, one at microwave frequencies and one at far-infrared frequencies. This latter case is the first time that retrieval of metamaterial optical constants directly from experimental data had been performed for any infrared metamaterial.

The work of chapter 4 addresses the issue of bandwidth, a sizeable hurdle when conceptualizing metamaterial applications or designing devices. The use of resonant structures to achieve the unique electromagnetic response necessarily imposes chromatic dispersion and bandwidth. In an attempt to assuage this drawback, we have realized a metamaterial capable of adjusting its electromagnetic properties in real-time response to external stimulus. We utilize a hybrid-metamaterial architecture which employs novel properties of Vanadium Dioxide thin films. The result is a frequency-agile metamaterial, with the ability to shift it's resonance by up to 20%. This approach also serves to illustrate the power inherent in a hybrid-metamaterial approach; combining unique properties found in condensed matter systems with the electromagnetic tailoring possible in metamaterials.

Chapter 5 introduces and explores a new application of metamaterials as sensors. The individual elements of a metamaterial array can behave as Resonant Electric Dielectric Sensors (REDS), responding to the presence of dielectric material by shifting their resonant frequency. Using a metamaterial for this purpose presents several advantages. The natural tendency of metamaterial resonators to exhibit field-concentration (directing applied energy into the small capacitive gap) concentrates the detection volume to an area much smaller than the wavelength, and allows us to observe minute quantities of material. Experimentally, our sensor array (using Split-Ring Resonators) detects the addition of 1nanogram/cm^2 of silicon nanocrystals – orders of magnitude less than is detectible in the same spectrometer without the aid of our metamaterial REDS sensor. The use of metamaterials also extends the range where REDS-type sensors may operate into the infrared spectrum where the detection of 'fingerprint' compounds is of great interest.

II. Free-space microwave focusing by a negative-index gradient lens.

II.A Background

Since their first realization in 1999 by Smith et al [1] - guided by the earlier theoretical work of Veselago [2]– understanding of negative index materials (NIMs) has advanced rapidly. Metamaterial-style structures with negative index-of-refraction ($n = \sqrt{\varepsilon}\sqrt{\mu}$) were initially constructed and tested at microwave frequencies [3, 4]. They have since advanced through infrared [5, 6]. Although there remains controversy about the ability to push these materials into the visible spectrum [7, 8], a wide array of uses at lower frequencies have been proposed [9-11]. At least in the microwave region where structure geometries can be easily manipulated, we are now at a stage where direct applications are within reach. To date, microwave metamaterial structures have largely been of the 'wine-crate' construction [12], which is an interleaved matrix of planes of split-ring resonators (SRR's) and wires. This construction, while useful in demonstrating the basic phenomenon, lacks structural integrity and is time-consuming to construct. Previous work with wine-crate type gradient-index (GRIN) lenses was restricted in design complexity to cylindrical lenses [13-15], largely to simplify design and construction. Additionally, the limitations in design tolerance and material choice in wine-crate metamaterials has hindered the progress of experimental positive-gain lensing using NIM's. Given the recent evidence showing the superiority of NIM lenses in many situations [16], there is much interest in pushing these lenses toward applications.

Here, we present a radial GRIN lens, built from elements similar to the SRR's and wires of previous experiments [17], but integrated into a fabrication technique using traditional printed circuit board (PCB) technology [18]. Our durable, lightweight, modular lens operates as a positive gain spherical lens, focusing in two dimensions to achieve a focal spot amplitude +7db over incident. This represents a significant advance towards applications, as well as favorable progress in design complexity and construction technique.

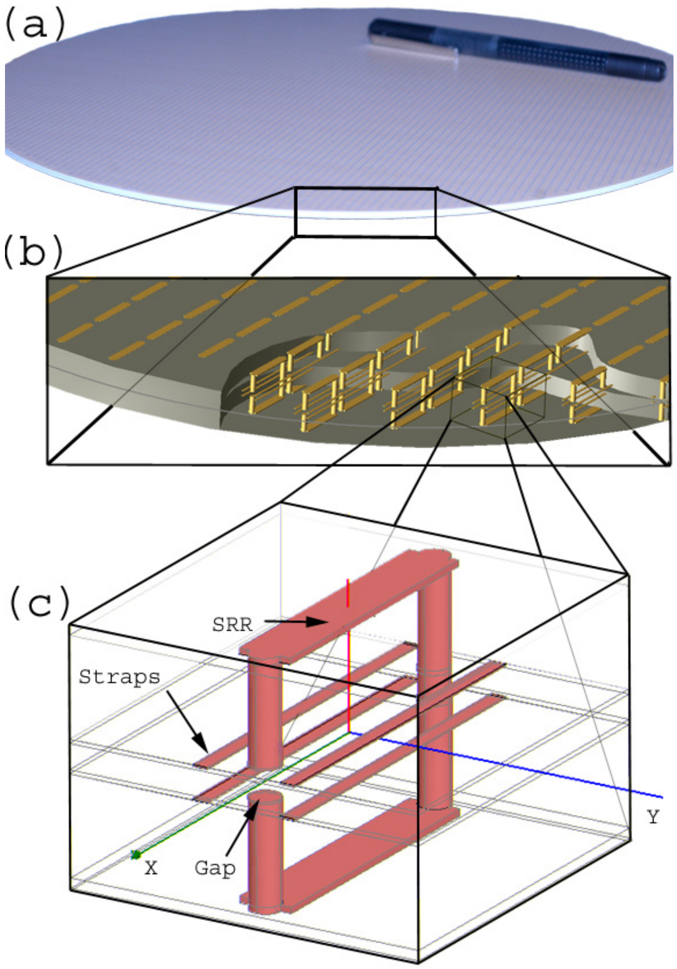

Figure II-1 Three tier diagram showing a) actual picture of a lens disc, b) blow-up illustrating unit-cell array, c) further blown up single unit cell with SRR and wire elements. Magnetic field is applied along Y direction, electric field is along X.

II.B Design and Fabrication

Using raytracing software written in Mathmatica, we designed a bi-planar (geometrically 'flat') lens with a radially varying gradient given in Eq.1.

$$\varepsilon(r,\omega) = \mu(r,\omega) = n(r,\omega) = -0.97 - 7.30r^2 + 0.18r^4 \quad (.1)$$

A 2mm thick disk, 15cm in radius, with this gradient is shown to behave as an f/9 lens. The continuous radial gradient is then mapped onto a Cartesian array 50 x 50 – giving each square in the index of refraction corresponding to Eq.II-1 at its center point. This array of squares is our metamaterial unit-cell layout. Each unit cell is a SRR and wire, shaped to have a specific magnetic and electric resonance. The result is a disc of ~8,000 unit cells (see Figure II-1), approximately ¼ of which are unique. This large number of steps gives an excellent approximation of a continuous gradient. The performance of metamaterial gradients such as this has previously been shown to be consistent with that of homogeneous-material continuous-gradients.[19]

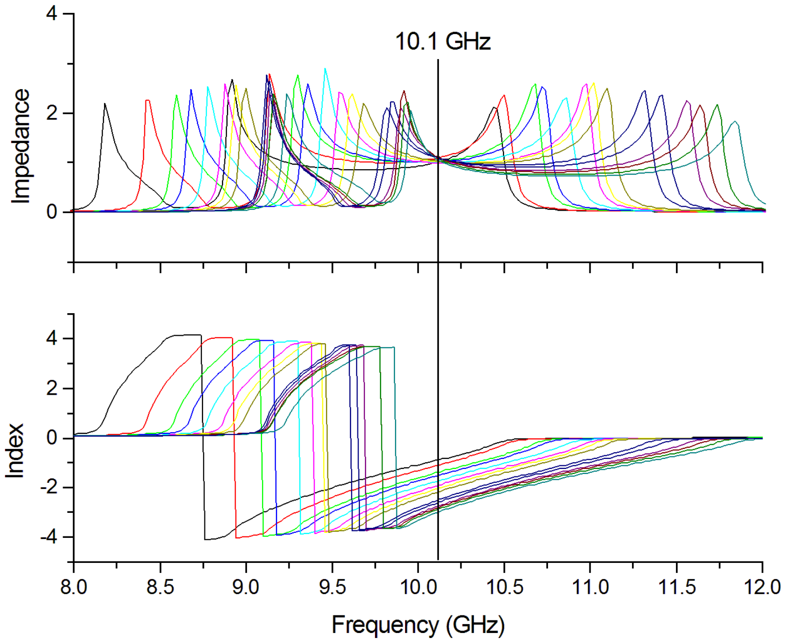

Figure II-2. Frequency dependence of the index and impedance for fourteen of our unit-cell geometries. Each curve is the generated output of a s-parameter inversion done on simulation data for one geometry.

To obtain the desired index of refraction at each unit cell we need to control the electric and magnetic resonance characteristics [17]. Changing the shape and spacing of our metamaterial constituents will do just this, and we explore many variations of the geometry shown in Figure II-1c. The geometry variations involve changing the wire thickness and location and the capacitor-gap pad radius. These geometries are simulated to retrieve the four scattering- (S) parameters in the commercial finite element software package HFSS by Ansoft. A standard retrieval method [18, 20] was used to invert the four s-parameters into real and imaginary electric permittivity $\varepsilon(\omega)$ and magnetic permeability $\mu(\omega)$. Ten geometries were selected with index of refraction spanning desired index range predicted by Eq.II-1. The optimization goals, in addition to correct index, included matching impedance to free space, low loss-tangents, and insensitivity to small geometric changes. Both the index and impedance are highly dispersive parameters in any metamaterial, due to the use of resonant structures. An operating frequency of 10.1 GHz was selected for the design. Figure II-2 shows the dispersion of the index and impedance around this design frequency.

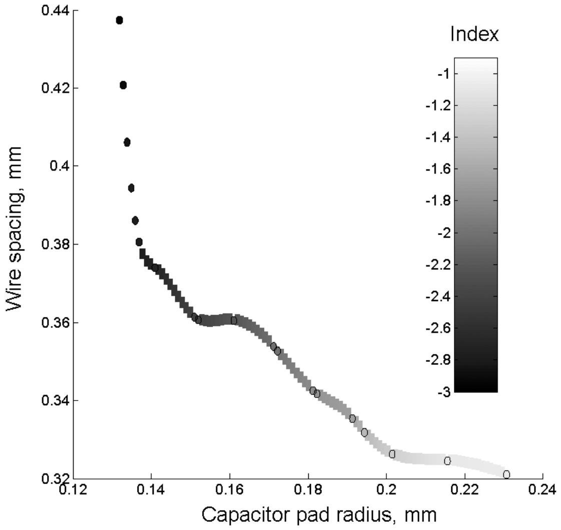

Figure II-3. Parametric plot of index-of-refraction as a function of wire spacing and capacitor pad radius. The wire thickness (not plotted) is varied to keep the impedance constant (z=1.06 ± 0.05) throughout. Simulation points shown in outlined circles, squares are interpolated.

The effects of wire thickness, wire location, and capacitor area on the electromagnetic parameters are intertwined, making the design a process relying largely on experience and iterative refinement. We can, however, still identify some variables as heavily influential on particular electromagnetic properties. The parametric graph in Figure II-3 helps to illustrate this for the dependence of index on wire spacing and capacitor pad size. Plotting the index as a function of both the wire spacing and the capacitor pad radius, we see the index decreases with increasing capacitor pad radius. The wire spacing also has some effect on the index - decreasing with a reduction in wire spacing - though less strongly than the capacitor pad. The circular data points in the graph are structures which were simulated in HFSS as described above, and the square data points are the result of an interpolation between simulation points. The impedance ($Z = \sqrt{\varepsilon / \mu}$) throughout is almost constant, at 1.06 ± 0.05 for the 10 designed cells. This is achieved by additionally varying the wire thickness, which predominantly affects the impedance. Each data point in Figure II-3 represents one unit cell design, like the one shown in Figure II-1c. These unit cells were then arranged in a two-dimensional Cartesian array (in the X-Y plane), as shown in Figure II-1b. This creates a PCB disc 2mm thick and 30cm in diameter (pictured in Figure II-1a.). The unit cells are aligned such that the wires run in the X direction, causing the resultant lens to respond correctly to a single linear polarization. These disks can be stacked up to build a more powerful lens, decreasing the lens f/# approximately linearly. This results in a highly modular design.

II.C Testing and performance

For testing, a microwave compact antenna test range (CATR) was designed and built using an off-axis parabolic reflector coupled with a standard feed horn. The polarized plane-wave coming off the dish shows moderately good amplitude uniformity - to within ~2db over the lens face. The lens is embedded in a Styrofoam barrier (with ε=1.03, μ=1) that minimizes the effects of the baffle on the observed fields, and is oriented normal to the plane wave in such a way that the incident electric and magnetic fields lie in the plane of the lens and are aligned to excite the wires and rings respectively. Our plane wave is measured to have very low cross-polarization (< -30dB) which allows us to neglect any effects of stray incorrect polarization states. The image side of the lens is scanned in volume using a dipole detector, guided by a xyz translation stage. This experimental setup is sketched as an insert in Figure II-4. The source and detector are connected to a Hewlett-Packard HP8510 Vector network analyzer, which performs a frequency sweep at each spatial point. Data is collected in a 4D array: in XYZ volume over 24cm by 30cm by 54cm every ½ cm, and between 8GHz and 12GHz every 0.04 GHz. This data is taken for various thicknesses of the lens.

In analyzing this data we take into consideration that at ~10GHz, our lens is only ~10 wavelengths in diameter - and diffractive edge effects may play an important role in its behavior. We wish to consider those lens thicknesses which will reduce diffractive effects by minimizing the difference in optical path length between two rays – one passing through the edge of the lens and the second through Styrofoam near the lens edge - written as

II.2
$$\Delta\phi_{OPL} = 2\pi d \tfrac{(n_0 - n_1)}{\lambda_0} .$$

At the lens edge, the index-of-refraction is n_1=-2.67. Solving equation 2, the first two thicknesses which result in integer 2ϕ phase difference are d_1=8.1mm and d_2=16.2mm. Each lens disc is 2mm thick, so a lens constructed of 4 or 8 layers will minimize the influence of diffractive effects in the focal field. Here we present these two thicknesses; the interesting behavior of highly diffractive small lenses will be considered in a later work.

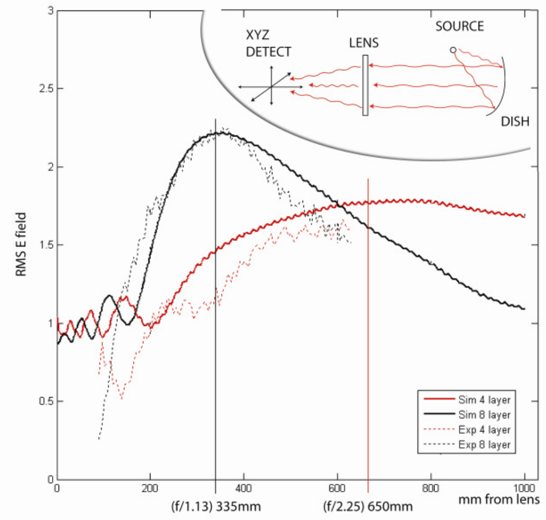

Figure II-4. RMS Electric field amplitude in both simulation (solid) and experiment (dash) along the optical axis for GRIN lenses comprised of 8 layers (black) and 4 layers (red). Data taken at 10.45 GHz

Figure II-4 shows data for an 8-layer lens (in black) and a 4-layer lens (in grey). Dashed lines are experimental data for the RMS field amplitude along the optical axis (x=0, y=0, translate z) at 10.45 GHz, normalized to incident field amplitude. Solid curves are Microwave Studio simulation results for the same 8layer and 4 layer lenses, the details of which will be discussed presently. We find good agreement between the experimental and simulation results for both the 8 and 4 layer lenses. We also find good agreement with the predictions of geometric optics, as determined by our ray-tracing software. Our 8 layer and 4 layer lenses have f-numbers of 1.13 and 2.25 respectively. The focal lengths predicted by these f-numbers are marked as vertical lines in Figure II-4.

The 8 layer lens focuses almost exactly where predicted, and though the physical limitation of our scanning stage does not allow enough z-axis translation to reach the 4-layer predicted focus, the line shape is still in good agreement with simulation.

Microwave Studio simulations are done in the form of a 2-dimensional 'slice' of the lens along its diameter. This reduces the number of meshcells necessary for our finite-difference time-domain solver to a threshold manageable on a high-end PC. As in our fabricated lens, our simulation uses 50 unit cells across its radius, where each unit cell has been assigned a Lorentzian dispersion in both ε and μ. This dispersion is designed such that both the index and loss-tangent at 10.1GHz are very close to the design parameters. To match our experimental setup, both Microwave Studio and ray-tracing simulations use a linearly polarized incident plane wave. We find only a ~3% difference between the design frequency of 10.1 GHz and what we determine to be the best operation frequency at 10.45 GHz. This small shift is likely due to tolerances in both the PCB construction, and the dielectric. Our unit cell designs call for a tolerance of 1 mil (25 μm), which is at the current limit of technology offered by board fabrication vendors. Fabrication errors such as registration misalignment would cause a shift in the parameters of all unit cells. Similarly, an increase in the board dielectric of only 5% would cause a shift in operation frequency near that observed.

The amplitudes in Figure II-4 are with respect to incident RMS amplitude 1, with simulation and experiment normalized separately and not relative to each other. As well as showing strong support for the accuracy of our design and fabrication, this illustrates another important feature of this lens: it produces field intensities at the focus that are much greater than that of the incident plane wave. Many researchers have been concerned that losses in existing left-handed materials would preclude their use in real applications. Here we have demonstrated this is not necessarily the case, at least for the microwave regime. The intentional impedance matching of each unit cell to air, and low loss-tangents, result in decreased back-reflection and absorption respectively. The peak amplitude of the 8-layer lens focus is nearly 7db greater than the incident wave amplitude.

A cross-sectional display of the field amplitude in the XY plane at the focal length (Figure II-5 checkered surface) nicely illustrates this gain-peak above the average incident amplitude (translucent plane). The solid black line is the beam focus 1-D profile from the Microwave Studio simulations. The slight tilt of the focus profile as well as the larger side lobes are attributed to non-uniform lens illumination due to the spherical falloff factor inherent in horn-feeding an off-center parabolic dish [21].

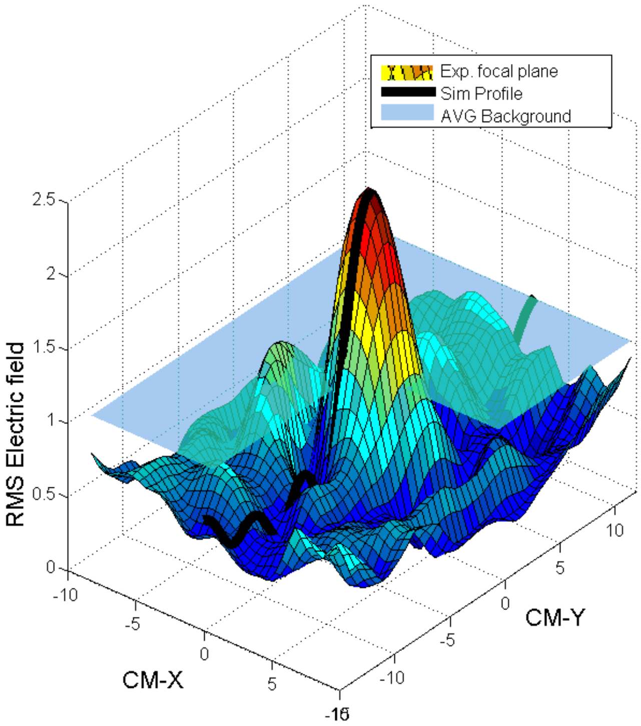

Figure II-5. 2-D profile of the E-field at the 8-layer lens focal plane (solid colored checker). Simulation 1-D focus profile overlaid (black line). No-lens averaged background (transparent plane) shown to emphasize magnification.

The dispersion of the unit cell index and impedance enforces an effective bandwidth on the useable frequencies of this lens. Even small variations in frequency may perturb our gradient index profile. In an attempt to maximize our useable bandwidth, we have designed the index dispersion such that small shifts around the operating frequency will cause a constant nearly linear change in the index of all cells. In reality the index dispersion curves of the 2000 geometries are unlikely to have identically sloped curves like we see in Figure II-2. The impedance is also quite sensitive to frequency. As a method of estimating the effective bandwidth of our lens, we plot the amplitude of the focus point (x=0, y=0, z=33.5 cm) as a function of frequency around 10.45 GHz, shown in Figure II-6. Looking at this plot, we see a region between 10.3 GHz and 10.6 GHz where the focal spot amplitude is relatively flat, and it falls off quickly away from this region. We consider this to be the useable bandwidth. Away from this region the behavior of the lens is quite hard to characterize or understand, as it degenerates into a pseudo-random grid of indexes and impedances. One way of further probing the dispersion of our parameters would be to do near-field microwave measurements of small unit-cell sized regions of the lens.

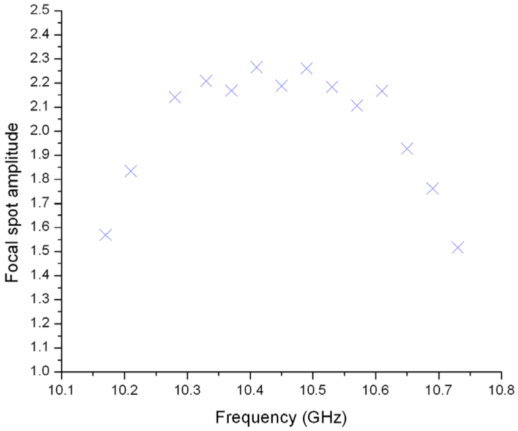

Figure II-6. The RMS electric field amplitude at the focus point (x=0, y=0, z=33.5 cm) as a function of frequency. The bandwidth is taken to be the flat region between 10.25 and 10.6 GHz.

Although to date we have only investigated the normal incidence behavior of this lens, there is reason to suspect the off-angle aberrations of negative index media may be superior to those of positive media in many cases [16]. So long as we retain the correct linear polarization of the electric field discussed above, there is nothing to prohibit operation at slight angles away from the normal. Deviation from normal incidence will, however, cause a shift in the optical parameters of the unit cells, and the off-angle behavior of anisotropic metamaterials such as ours is has been investigated by many groups [22-25]. The off-normal incidence behavior of metamaterials will be discussed further in chapter III.

II.D Summary of UCSD GRIN lens

In this work, we have designed, constructed, and tested a bi-planar gradient lens whose gradient ranges from n= -0.97 to n= -2.67. This microwave lens is one of the first negative-refractive-index lenses ever, and the first fabricated out of robust composite metamaterial. As well as providing experimental confirmation of the behavior of negative-index gradient lenses, the adaptation of conventional Printed Circuit Board fabrication to 3D metamaterials immensely simplifies the construction process. Commercial PCB companies offer tolerances in excess of what's needed for microwave metamaterials, and quick turnaround times are possible. Our gradient lens with nearly 2,000 unique unit cells represents the ultimate demonstration of the power of this recently pioneered construction technique[8]. The performance characteristics of the lens are in excellent quantitative agreement with simulation, giving us the tools necessary to repeat this process in the design of any desired optics, and one can envision realistically creating three-dimensional structures with finely tailored index-of-refraction throughout. This lens represented something of a milestone in the metamaterials community, and the experience is now being applied to even more complicated 3D microwave metamaterials, such as missile radomes.

III. Characterization and modeling of metamaterial thin films

III.A Introduction

Metamaterials derive much of their electromagnetic properties from their structure, as opposed to entirely from their material composition. This seemingly simple fact has resulted in experimental complexities when measuring metamaterial samples. Large anisotropy and spatial dispersion make metamaterial samples quite unlike other condensed matter samples, and many usual measurement techniques are ineffective. In this chapter, we discuss the development of a technique suitable for retrieving the permittivity and permeability of metamaterial samples. The technique combines *ad-hoc* ellipseometry with modeling; fitting oscillator functions to transmission and reflection data taken at multiple angles of incidence and polarizations. In this manner additional data sets can be added as necessary to adapt for complicated metamaterials and/or samples with multiple layers - ensuring the accuracy of the model-fit.

III.B Background

In the absence of magnetoelectric coupling, the electromagnetic response of a continuous material can be entirely described solely in terms of its three-by-three electric permittivity $\underline{\underline{\varepsilon}}(\omega)$ and magnetic permeability $\underline{\underline{\mu}}(\omega)$ tensors which together represent the material's response to light at any frequency along any direction in three dimensions. These tensors are macroscopic bulk parameters that average the responding motion of microscopic dipoles and currents [26]. Once these material property tensors are determined, the Fresnel equations can be derived from Maxwell's equations by solving the appropriate electromagnetic boundary-value problem at the various material interfaces. The Fresnel equations describe the reflection and transmission of polarized light incident at any angle to a planar layer or stack of layers of one or more materials.

Metamaterials have recently garnered much attention because of their ability to achieve unusual electromagnetic responses, such as negative index, not typically found in natural materials [27-29]. Loosely defined, a metamaterial is an artificial crystal in which mesoscopic inclusion structures replace the microscopic atoms or molecular structures of natural materials. So long as the scale of the periodicity of the metamaterial is significantly smaller than the wavelengths of interest, the metamaterial is electromagnetically indistinguishable from a continuous material. Metamaterials have a periodicity (**P**) that is usually on the order of $\mathbf{P} \sim \lambda /10$, which contrasts with natural crystals for which the Wigner-Seitz unit-cell may be 5A, corresponding to $\mathbf{P} \sim \lambda/1\text{x}10^8$ at microwave frequencies. Nevertheless, it has been shown that macroscopic approximations like the permittivity and permeability tensors remain largely valid for these metamaterials [30], and the permittivity and permeability description has shown considerable success in describing the behavior of electromagnetic scattering from metamaterial samples.

If metamaterials are to be treated on the same basis as natural materials, it is necessary to define a procedure by which the permittivity and permeability tensors can be determined for a metamaterial. To date, two methods have been introduced to retrieve the material parameters from simulated or measured metamaterial structures: the field averaging method and the S-parameter retrieval method. In the field averaging method, which can be readily applied to simulated metamaterial structures, the local electric and magnetic fields are averaged over surfaces and edges to arrive at macroscopic values for the **E**, **D**, **B** and **H** fields. From these macroscopic averaged fields values, the tensor elements for the permittivity and permeability tensors can be found. In the S-parameter retrieval method, the scattering parameters (or reflection and transmission amplitudes and phases) for a finite length of metamaterial are either computed or measured.

The S-parameters retrieval method has now been widely applied for the characterization of metamaterial structures at RF and microwave frequencies [18, 20, 30, 31]. S-parameters contain both amplitude and phase information, and so a single frequency S-parameters measurement yields four values that can in turn be inverted to find the real and imaginary permittivity and permeability components transverse to the given direction of propagation. While the method does not provide unique solutions due to the nature of the required numerical inversion, typically erroneous solutions can be excluded by studying the retrieved parameters over a large enough frequency band that unphysical branch jumps can be identified.

The S-parameters retrieval method suffers from several drawbacks: The difficulty in obtaining phase information at THz and higher frequencies renders the S-parameter retrieval method impractical. Therefore, characterization of metamaterial samples at infrared frequencies has relied on the consistency of transmission and reflection magnitudes with simulation results. This approach has enabled the assignment of magnetic or electric responses to fabricated metamaterials, but has not resulted in a direct quantification of material properties for THz and higher frequency structures [7, 32]. In addition, most work to date has been restricted to waves incident along one of the three primary axes, since this orthogonal geometry is far more convenient for both numerical simulation and analytical expressions.

However, since the Fresnel equations can be generated for oblique incidence waves; measurement of the reflected or transmitted wave magnitude for multiple angles provides additional information that can be used to compensate for the absence of phase data. In particular, the reflection and transmission coefficients corresponding to waves at varying incident angles contain different relative contributions of the material parameter tensor elements; and thus by collecting transmission/reflection data at a number of angles, it is possible to arrive at an over-determined system of equations for $\underline{\underline{\varepsilon}}(\omega)$ and $\underline{\underline{\mu}}(\omega)$ that can be solved by known methods. The Fresnel formulation thus provides a rigorous method for determining $\underline{\underline{\varepsilon}}(\omega)$ and $\underline{\underline{\mu}}(\omega)$ of a structure from experimental data - in any material where $\underline{\underline{\varepsilon}}(\omega)$ and $\underline{\underline{\mu}}(\omega)$ can be safely defined. While it should be possible to perform this characterization at every given frequency, a great computational simplification occurs if the material is modeled as a collection of electric and magnetic oscillators whose functional forms are known. By fitting the parameters for these oscillators to the measured data, fewer experimental frequency points need be acquired and calculation complexity for fitting is considerably reduced. This approach is common and well documented in the field of optical spectroscopy. The software package WVASE [33], for example, is a common tool to do this analysis in a non-magnetic ($\underline{\underline{\mu}}(\omega)$ =1) crystal [34, 35].

In this chapter, we attempt to use the Fresnel approach to describe the transmission and reflection through a metamaterial at a variety of angles. The frequency-dependent functional form for the double-SRR structure has been previously investigated and reported upon [36], so we apply a spectroscopic approach here, modeling the metamaterial as a collection of magnetic and electric oscillators. The success of the Fresnel method applied to a metamaterial relies upon the entirety of the metamaterial electromagnetic scattering response being governed by a well-defined permittivity and permeability. This has been shown to be largely true, although new revisions to effective medium theory continue to account for artifacts such as periodicity and diffraction [20, 30].

One last point to be made is that for infrared and higher frequencies, metamaterial test samples usually comprise a single plane of lithographically patterned conducting elements, with thickness much less than the wavelengths of interest, i.e. ~ λ/300. The scattering problem, then, should properly be formulated as a boundary value problem for which electric and magnetic polarizabilities can be defined for the intervening metamaterial layer. Here, we introduce an effective *electromagnetic* thickness (*d*) for the metamaterial layer that will facilitate the assignment of effective bulk permittivity and permeability. Physically, this effective thickness may be thought of as a representation of the distance which the local fields extend out away from the metamaterial plane. If *d* is on the order of the unit cell dimension in the plane, then the resulting effective medium parameters generally predict the scattering behavior of a composite medium formed by stacking multiple planar samples together spaced *d* apart. However this holds true only asymptotically - if the expected composite sample consists of planes stacked with an intervening distance that is less than **P**, then cell-to-cell interactions between planes invalidate the use of a single layer to predict the response of the crystal [37]. While this ambiguity is of concern, we note that it is inherent to other retrieval techniques (e.g., S-parameters retrieval) for which the effective medium parameters are also often derived from a single layer of metamaterial unit cells. The s-parameter method, even when also relying on the effective thickness of a single layer, has still proved to be quite accurate [18].

To show support for the use of an effective thickness parameter in our case, we analyze numerical results for the resonant field distribution in a typical metamaterial structure, and show inter-layer interactions can be safely neglected for our structure. Figure III-1 plots the in-plane electric and out-of-plane magnetic fields extending away from a SRR structure for two cases: in a single layer planar array, and in an infinite crystal. We notice in both cases, the induced fields have almost entirely died out by the edge of the unit cell. Also of importance is that the field distributions for the single layer and infinite crystal are quite similar. The smaller overall amplitudes of the induced fields for the single layer are likely a result of energy absorption inherent at the PML cell-boundary – not present in the stacked layer. These results suggest a single layer taken to have effective thickness *d* will closely approximate the permittivity and permeability of many layers placed with spacing *d*. An experimental probe of the unit-cell fields and effective thickness is planned for the near future to further investigate this.

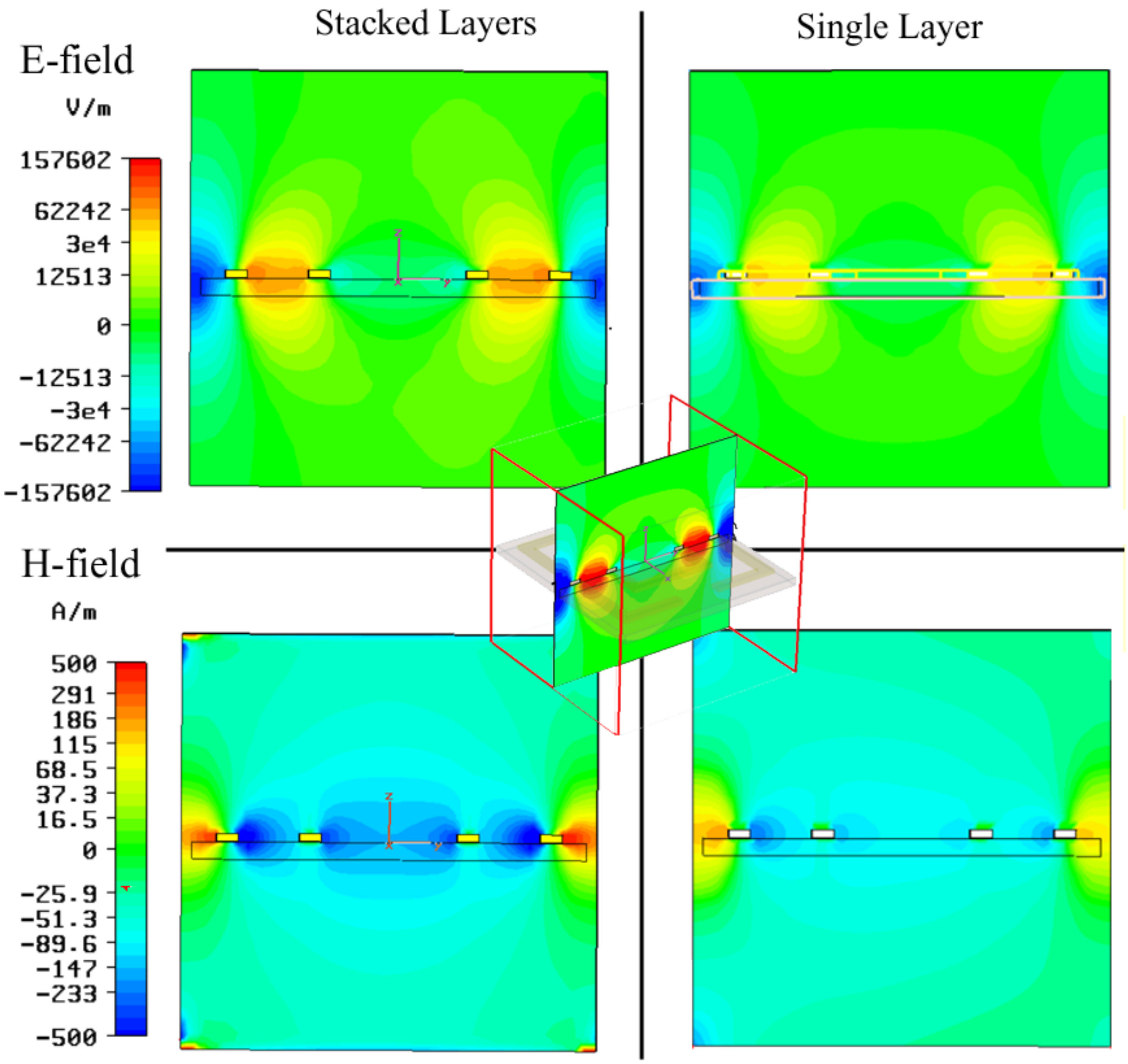

Figure III-1. Numerical FDTD results for a cross section of the electric and magnetic distributions in a SRR structure for both a single layer and infinitely stacked layers. The fields plotted are **E** in the sample plane, and **B** normal to the sample plane. The center insert shows the geometry of the 2D field slice relative to the SRR.

III.C Characterization of a Microwave Metamaterial

III.C.1 Microwave Experiment

We fabricated a planar array of SRR's designed to have distinguishable electric and magnetic resonances. This copper-on-FR4 [38] array was etched using lithography techniques typical to printed circuit boards, which have been described elsewhere. We opt for the Dual-SRR configuration shown Figure III-2 and Figure III-3, which has been used in previous experiments [1, 39, 40]. This planar sample is placed in a microwave transmission / reflection setup using polarized horn-lens assemblies driven and analyzed by a vector-network-analyzer. Transmission and Reflection amplitude data was collected for s-polarized microwaves from 0˚ (normal) to 50˚ in transmission and from 25˚ to 50˚ in reflection. Figure III-2 illustrates this setup. The reflection data is limited in angle to a minimum of 25˚ due to the physical size of the horns. Data is collected in frequency over the horn/lens useable range of 10 GHz to 20 GHz.

Figure III-4a shows the collected transmission data for angles 0˚, 20˚, and 40˚ (symbols). Figure III-4b shows the reflection data for angles 25˚ and 40˚ (symbols). In both spectra sets, we can identify one predominate feature: a narrow angle-dependent resonance near 14 GHz. There is also a broad trend in each data – decreasing transmission (increasing reflection) with frequency. As discussed in previous literature [41], an individual SRR element exhibits in-plane electric and out-of-plane magnetic responses - as well as magnetoelectric coupling. The magnetoelectric coupling can be eliminated in a bi-isotropic array such as ours - with reflection symmetry about the mirror lines Γ and ζ (see Figure III-2) [20]. The magnetic response of the SRR involves magnetic flux passing through its nearly-closed loops. This flux is zero for normal incidence (0˚) light and increases as the incidence becomes more oblique. The electric response of a SRR comes from the sides acting as cut-wire dipole antennae.

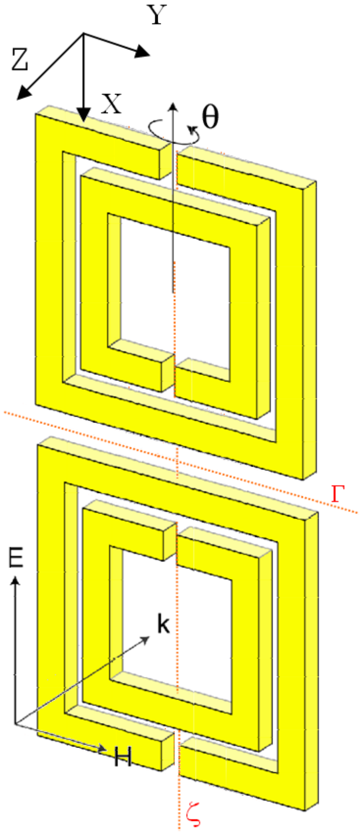

Figure III-2 The geometry of the symmetric SRR array in s-polarization

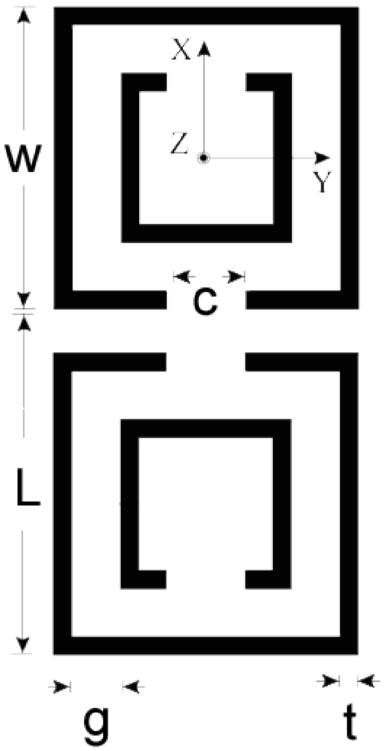

Figure III-3 SRR dimensions in mm: w=2, t=0.13, g=0.35, c=0.47, L=2.4. The vertical thickness of the copper is ~ 37μm (1oz. copper)

Returning to the experimental data in Figure III-4, we attribute the narrow feature at 14 GHz to the magnetic response of the SRR. The feature is absent in the transmission data at 0˚ where the applied magnetic field is in-plane, and grows as the incident angle becomes more obtuse and we include normal components of **B** which react with the

SRR. The broad trend of decreasing transmission (increasing reflection) is part of the electric interaction – it is the onset of the electric oscillator which at higher frequencies will look much like a stronger broader version of the magnetic response.

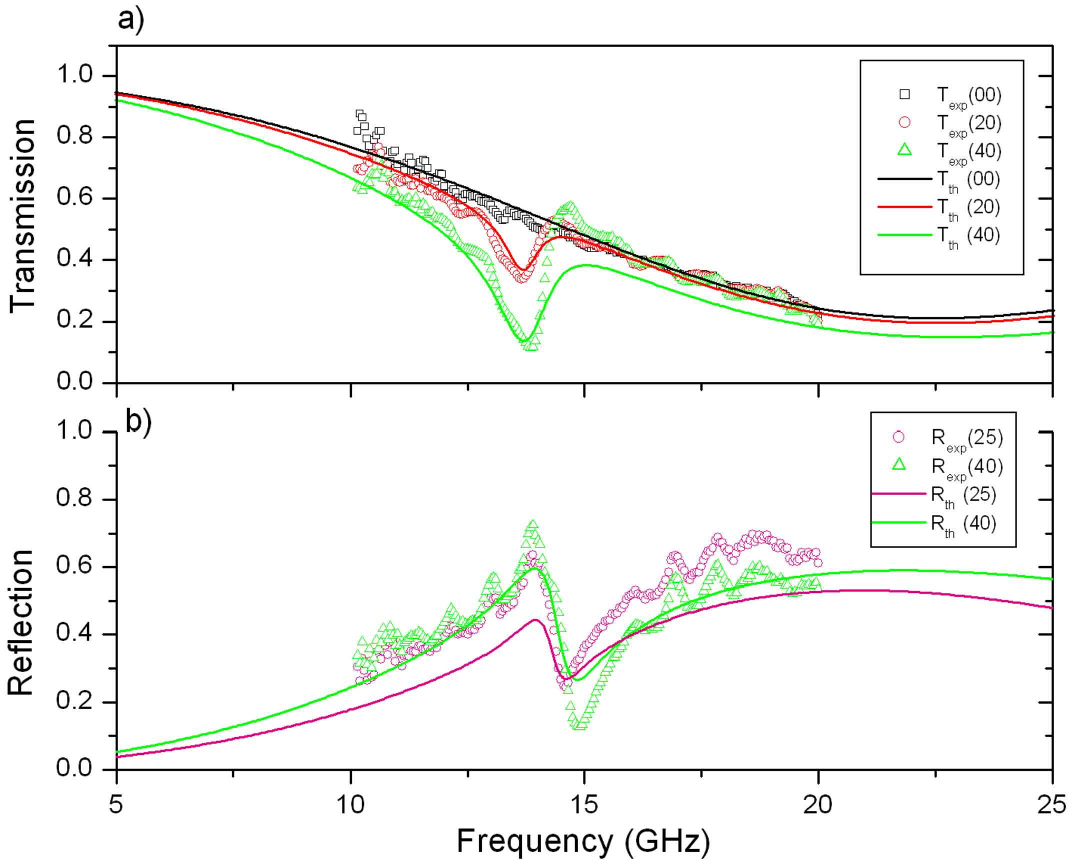

Figure III-4 Experimental data at selected angles for transmission (0°, 20°, 40°) and reflection (25°, 40°) with a fit of the Fresnel theory. The oscillator parameters corresponding to the two fits are shown in Table 1.

III.C.2 Theory

To describe and characterize the angle-dependant response of the SRR, we turn to an effective medium picture, and solve for the Maxwell's equations boundary value problem for transmission and reflection through our sample. The electromagnetic response of the resulting composite planar sample is then described as follows: Electric fields along the two sides of the composite induce an electric dipolar response (in the $\hat{x}$ and $\hat{y}$ directions), and magnetic fields that penetrate the SRR loops (in the $\hat{z}$ direction) induce a magnetic dipolar response. We then assign the tensors

III.1 $$\underline{\underline{\varepsilon}}(\omega)=\begin{pmatrix}\varepsilon_{xx}(\omega) & 0 & 0\\ 0 & \varepsilon_{yy}(\omega) & 0\\ 0 & 0 & 1\end{pmatrix}\quad \underline{\underline{\mu}}(\omega)=\begin{pmatrix}1 & 0 & 0\\ 0 & 1 & 0\\ 0 & 0 & \mu_{zz}(\omega)\end{pmatrix}.$$

The response of ε_{yy} is considerably more complicated than that of ε_{xx} due to the asymmetry of SRR sides lying in the $\hat{y}$ direction, so we confine E to the $\hat{x}$ direction (s-polarization) using linearly-polarized incident light, as indicated by Figure III-2.

Working from these tensors, we now summarize the Fresnel equations for the reflection and transmission of light incident along vacuum wavevector **k** on a planar layer of stratified material, where k is at an incidence angle ⍰ from the normal ($\hat{z}$ direction). Although we restrict our analysis to s-polarized light, the procedure for p-polarization follows from the duality of **E, H**. This is a three region boundary value problem [26, 42], with the continuity of $\mathbf{E_x}$ and $\mathbf{H_y}$. Maxwell's equations are written as

III.2 $$\nabla\times\mathbf{H}=-i\omega\underline{\underline{\boldsymbol{\varepsilon}}}\varepsilon_0\mathbf{E}\qquad \nabla\times\mathbf{E}=+i\omega\underline{\underline{\boldsymbol{\mu}}}\mu_0\mathbf{H}\,.$$

Combining these to get the wave equation and assuming sinusoidal dependence of all solutions, we arrive at the dispersion relation inside the medium,

III.3
$$\frac{q_y^2}{\mu_{zz}}+\frac{q_z^2}{1}=\omega^2\varepsilon_{xx}\,,$$

where **q** is the wavevector in medium. Setting up forward- and backward-propagating waves in each region, and matching boundaries at each interface, we solve the system of 4 linear equations (2 generated from each boundary) to find

III.4
$$\begin{gathered}t=\frac{e^{-ik_zd}}{\cos\left(q_zd\right)-\frac{i}{2}\left(\frac{\mu_{zz}k_z}{q_z}+\frac{q_z}{\mu_{zz}k_z}\right)\sin\left(q_zd\right)}\\ r=\frac{\frac{i}{2}\left(\frac{q_z}{\mu_{zz}k_z}-\frac{\mu_{zz}k_z}{q_z}\right)\sin\left(q_zd\right)}{\cos\left(q_zd\right)-\frac{i}{2}\left(\frac{\mu_{zz}k_z}{q_z}+\frac{q_z}{\mu_{zz}k_z}\right)\sin\left(q_zd\right)}\end{gathered}\quad .$$

These are the coefficients of reflection and transmission for our system. The incident angle θ is related to the components of the vacuum wavevector **k** and medium wavevector **q** as

III.5
$$\sin\theta=\frac{k_y}{k_0}=\frac{q_y}{k_0}\,.$$

Let us now assume some frequency dependent form for ε_{xx} and μ_{zz} to simplify the calculation, as discussed in the background section. The most accurate form for the electric and magnetic response for the symmetrized SRR configuration is: [19, 43, 44]

III.6
$$\varepsilon_{xx}(\omega)=\varepsilon_s-\frac{A_e\omega_p{}^2}{\omega^2-\omega^2{}_{e0}+i\omega\gamma_e}\ .$$
$$\mu_{zz}(\omega)=1-\frac{A_m\omega^2}{\omega^2-\omega_{m0}^2+i\omega\gamma_m}$$

Where A are oscillator amplitudes, ω_p are plasma frequency, ω_0 are resonance center frequency, γ are damping-constants and ε_s is the static (zero-frequency) dielectric constant of the metamaterial. This expression for μ_{zz} has been shown to be valid over the limited frequency ranges where the composite can be considered an effective medium, despite the unphysical fact that it does not approach unity at large frequencies. None of these expressions will be valid for frequencies at which the wavelength approaches the periodicity, so this is of no concern.

III.C.3 Analysis

We now wish to determine how well the transmission and reflection features seen in Figure III-4 are captured by this simple Fresnel model, and to evaluate the fitted values for μ_{zz} and ε_{xx}. To do this, Eq.III-4 was incorporated into a MATLAB program which uses a combination of graphical user-feedback and least-squares minimization to choose the material parameters in Eq.III-6. The fitting procedure is roughly as follows: First we guess appropriate values for the center frequencies of the oscillators based on the observed data. Values are first selected by hand so the fit roughly matches the features of the data. All parameters of the oscillator are then repeatedly adjusted to find the values with the smallest residue that still qualitatively matches the data.

Because the forms of Eq.III-6 are only a good first approximation to the actual response of our material it is necessary to do this fitting by hand to avoid a-physical solutions.

We find a relatively good qualitative agreement between the experimental data and the modeled transmission/ reflectance curves fit to the data. This fit is plotted as the solid lines in Figure III-4. The general trend of decreasing transmission (increasing reflection) with increasing frequency is captured by the higher frequency in-plane electric oscillator. The attributes of the magnetic modeled transmission dip match well to the data. The final fit parameters to transmission and reflection are given in Table III-1, and plotted in Figure III-5 Also given is an average residue measure of the fit: $\frac{\sum_{i=0}^{n}\left|y_{th(i)}-y_{data(i)}\right|}{n}$. We notice some disagreement in the region immediately following the magnetic resonance, where experimental transmission levels 'bounce' back up - increasingly so at larger angles. The cause of this is unknown, although we suspect it is from unaccounted oscillator strengths such as may arise from extended array interactions.

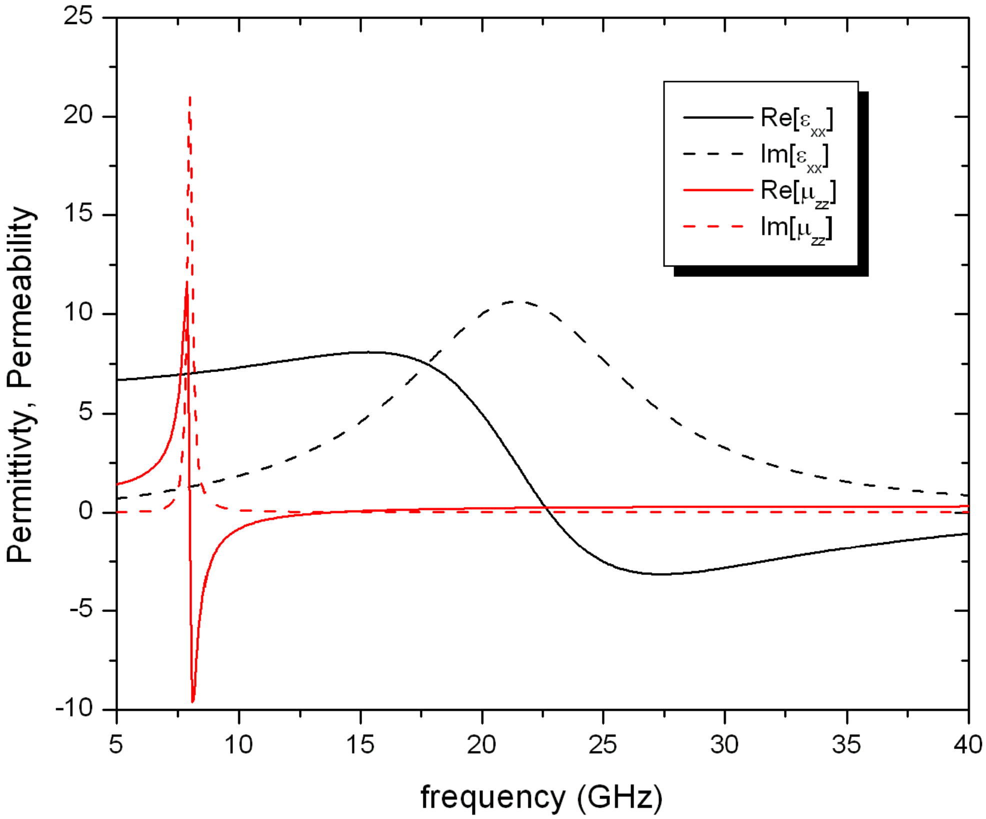

Figure III-5 Permittivity (black) and permeability (red) dispersion curves corresponding to the transmission fit.

In reflection, we also see good qualitative matching of the features in the data. The largest discrepancies in reflection are just after the magnetic dip, where levels in the model curve don't drop far enough. This corroborates the area of largest transmission discrepancy noted above, and suggests this area may be important for additional study. The disagreement at frequencies above 17 GHz is perhaps to be expected since wavelengths shrink and our effective-medium assumptions break down. As seen in Table III-1, the optimal fits for the transmission and reflection agree quite well for most oscillator parameters, but differ notably in the damping factors. The reflection and transmission experiments were conducted on separately fabricated samples, so we should not expect the fitted material parameters be identical.

Table III-1 Parameters from Eq.6 for Optical fit to transmission and reflection data.

Parameter	Transmission fit	Reflection fit
D	0.30 cm	0.30 cm
A_e	0.7	0.7
ω_{e0}	22.2 GHz	19.9 GHz
ω_p	62 GHz	62 GHz
γ_e	11.6 GHz	5.4 GHz
A_m	0.66	0.66
ω_m	8.0 GHz	8.3 GHz
γ_m Avg. Residue	0.25 GHz 0.037	0.19 GHz 0.081

In both transmission and reflection fits we identify a sharp magnetic resonance with ω_{m0} = 8 GHz, and a broad electric resonance around 22 GHz. These two optical parameters of interest, $\mu_{zz}(\omega)$ and $\varepsilon_{xx}(\omega)$, retrieved from our model fit, are plotted in Figure III-5. It is interesting to note that our model predicts the magnetic transmission/reflection response to occur at the zero-crossing of the permeability. This is the magnetic plasma frequency, and the response of the metamaterial at this point is best described as a collective longitudinal mode. There is a direct analogy to the electric response of anisotropic crystals in spectroscopy, known as the Berreman mode [45]. This differs from previous work, where the magnetic response of a SRR plane was assumed – although not shown – to occur at the center frequency ω_0. Certainly this feature warrants further investigation. The amplitude of the fitted permeability reaches a minimum of μ_{zz}= -9. This is somewhat larger, but still close to, the maximally negative permeability values taken from previous retrievals on similar SRR structures [18, 46].

One important factor that is left out by the method presented here is the spatial dispersion associated with the finite unit cell of the metamaterial structure. This limitation is not inherent to the overall method, but is related to our assumption of ideal frequency dependent forms for the material parameters (Eq. III-6). We expect that, especially in the vicinity of the resonance, the actual values of the permittivity and permeability will be somewhat different from the values derived here. In particular, the finite size of the unit cell is known to limit the range of possible permeability values associated with the magnetic resonance [20], causing significant distortions to the shape of both the real and imaginary parts of the resonance curves. For the SRR structure we have presented here, the retrieval suggests that the permeability should achieve values as large as $|\mu| \approx 9$; however, were the effects of spatial dispersion included we would expect the maximum permeability values to be significantly smaller. Work must be done to adapt this method to effective medium descriptions which include spatial dispersion.

The model-fitting procedure we have outlined can be further refined by incorporating a many-oscillator routine into the algorithm, thereby enabling frequency-by-frequency fitting rather than relying on specific forms for the effective medium parameters. Such a direct retrieval will allow us to investigate the smaller features seen in the experimental data, which may not be captured by the forms in Eq. III-6. In the context of a frequency-by-frequency inversion, the Fresnel equations may be used for a precise retrieval of the experimentally measured permittivity and permeability – rather than relying on the assumptions the form of Eq. III-6 make. During our analysis we found that the best-fit parameter for the effective sample thickness was 3mm, which is near enough to the unit-cell size of 2.4mm that we expect multiple layers to behave as a good continuous medium. Future work will include investigating the accuracy of this theory on thicker, multi-layer metamaterial crystals and higher frequency structures, as well as the development of a more rigorous point-by-point retrieval routine.

III.C.4 Summary of microwave characterization

We have shown that macroscopic permeability and permittivity tensors for a metamaterial can be used to derive Fresnel expressions for transmission and reflection from an interface at any incident angle, and that these expressions do a good job of predicting the qualitative spectroscopic features of the metamaterial. This description is quite useful, as it reduces the interaction at an angle of incidence to those fundamental magnetic and electric oscillators which the metamaterial structure mimics. Using these Fresnel expressions, experimental data for transmission (and/or reflection) at a number of angles across a broad frequency range can be used in a multiple-oscillator fitting routine to retrieve the effective-medium material parameters. The redundancy of the data at multiple angles increases the accuracy of this fitting; and it has been shown here to give reasonable values for the permittivity and permeability of a SRR 2D array in the microwave range.

In the near future, we hope to implement a direct retrieval procedure that will avoid the use of fixed, frequency dependent forms for the material parameters. The use of such forms for metamaterials having significant spatial dispersion constrains the retrieved parameters and can yield incorrect values in the region of the resonance. Still, direct inversion adds new difficulties, and the method we have presented is an important step towards quantifying the electromagnetic properties of planar metamaterial samples. Having demonstrated the procedure at microwave frequencies, we will now apply the technique to infrared frequency investigations.

III.D Characterization of a Far-infrared metamaterial

III.D.1 Introduction

The rapidly growing field of metamaterials is based around the ability to create tailored electric and magnetic responses in a host material by the inclusion of designed geometric, usually conducting elements. If these elements are positioned with a periodicity **P,** sufficiently smaller than the wavelength of applied light λ, ($P < \lambda/5$), they can form an effective medium electromagnetically indistinguishable from a continuous material [30]. Artificial materials have generated particular interest for their ability to exhibit magnetic response at frequency ranges where magnetic response is usually absent. Magnetism in natural materials is a quantum mechanical phenomenon, occurring whenever a net electron spin imbalance or an orbital spin imbalance creates a magnetic moment. In contrast, metamaterial tailored magnetism is quite different, resulting from a time-varying magnetic flux inducing resonant oscillatory electric currents along structured mesoscopic metallic inclusions.

Artificial magnetism has been well explored at microwave frequencies [18, 20, 27], but a rigorous quantitative retrieval of the full optical constants has not yet been done at THz and infrared frequencies. This is due, in part, to the difficulty in fabricating 3D bulk metamaterial structures; as well as difficulty in obtaining the complete phase information needed to retrieve the complex permittivity ε and permeability μ by way of direct inversion (the s-parameters method). At microwave frequencies, network analyzers are capable of obtaining the phases of both transmitted as well as reflected waves, enabling a retrieval procedure in which ε and μ can be determined. At optical frequencies, phase information can be obtained by interferometry or by time domain methods [47]. In neither of these optical frequency methods, however, can the phase of the reflected wave easily be determined, so that a true material characterization cannot be obtained - although measurements of just the refractive index do appear to be feasible [8, 48].

Here, we apply an alternative technique (referred to here as the Fresnel method) to characterize a planar SRR medium, using transmission amplitude data taken at multiple angles of incidence relative to the plane of the SRRs[22]. In this technique, the Fresnel formulas for reflection and transmission from a continuous material interface at an angle of incidence are derived. Multiple material layers can be stacked up, and the transmission through the entire stack can be found. By varying the material properties corresponding to each layer, we fit our modeled transmission spectra to the measured spectra. Fitting transmission datasets at *multiple-angles simultaneously* reduces the possible ambiguity of multiple best-fit solutions, thus obviating the need for phase information.

III.D.2 Far-Infrared Experiment

Our metamaterial sample consists of a planar layer of gold SRRs photo-lithographically patterned onto an n-doped, 30 Ω–cm resistivity, double-side polished silicon wafer, spin-coated with a layer of benzocyclobutene (BCB). BCB was chosen for its moderate dielectric constant, low losses in the THz range, and planarizing properties. It also helps with gold adhesion during the liftoff process used for the photolithography. The measured thickness of the BCB layer is 6 μm. The resultant sample geometry is shown in the insert of Figure III-6. Transmission measurements are performed in a Bruker 66v FTIR spectrometer, with the sample oriented such that s-polarized light is incident at angle θ from the surface normal. The sample dimensions are 1cm x 1cm (an array of 277 x 277 split rings). Because each SRR element is inherently bi-anisotropic, we symmetrized the SRR array (shown in the insert of Figure III-6) by making nearest neighbors mirror each other - so as to reduce magneto-electric coupling [23]. The resulting metamaterial layer can then be described by an in-plane permittivity $\varepsilon_x(\omega)$ and an out-of-plane permeability $\mu_z(\omega)$. The transmitted light amplitude is measured as a function of frequency from 0.61 THz to 6.06 THz, and at angles from θ=0˚ to θ=45˚. These transmission spectra are plotted in Figure III-6, in a low resolution (4cm^{-1} [0.12 THz])[49] which excludes Fabry-Perot (FP) fringes that arise from the Si substrate. In these spectra, there are two dominant features: an angle-dependant magnetic resonance at 1.27 THz, and an angle-independent stronger electric resonance at 3.03 THz.

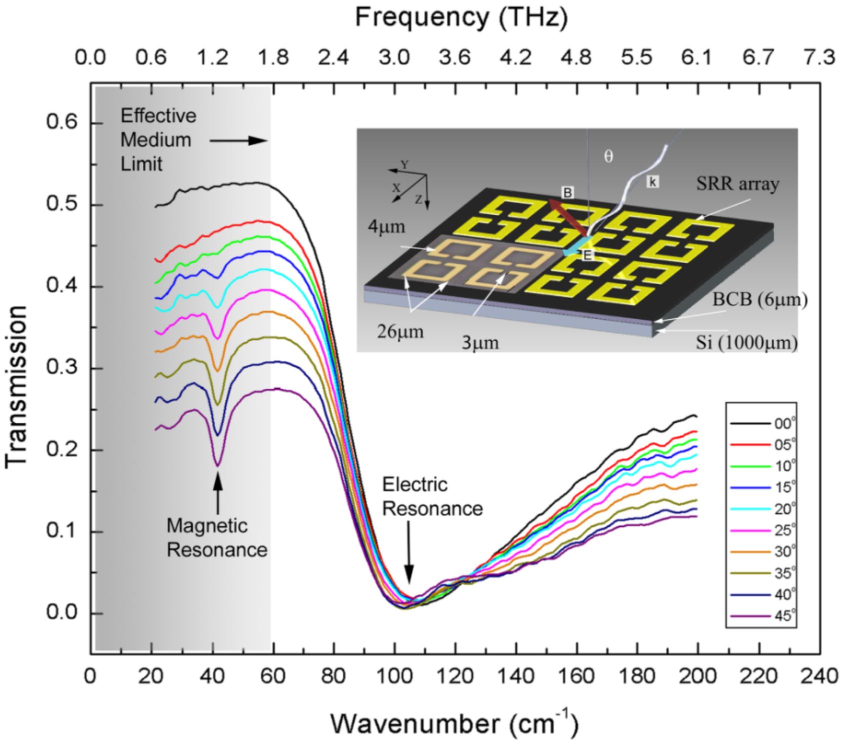

Figure III-6. Low resolution (4cm^{-1} [0.12 THz]) transmission spectra at angles of incidence ranging from 0° to 45°. Insert: Sample geometry and polarization orientation. The lower left 4 SRR's are an insert from an optical photograph, with included actual measured dimensions.

The assignment of electric and magnetic to the two features is straightforward as the polarization setup necessitates zero magnetic response - but strong electric response - at 0º. This assignment is corroborated by previous work wherein the electric resonance frequency always appears above the magnetic resonance frequency, and by simulation results identifying the position of the magnetic resonance (discussed below). The low-resolution spectra [50] allow the full range of the dataset (including the higher frequency electric resonance) to be plotted with clarity. High-resolution (0.25 cm^{-1} [0.008 THz]) spectra which do include the FP oscillations are plotted over a zoomed-in frequency range of 0.91 THz to 1.52 THz in Figure III-7 (color lines) for the angles of incidence θ=0˚,30˚,45˚. This frequency range bounds the magnetic resonance of interest, which manifests at 1.27 THz as an angle-dependant reduction in the amplitude of the FP oscillations.

By restricting our analysis to the range over which the effective medium approximation for metamaterials can be applied[1] (up to $\lambda > 5*$**P**, corresponding to ~1.7 THz in our array), we treat this sample as a simple multi-layer system: ~1000 μm of Silicon, 6 μm of BCB and a thickness of metamaterial. There is a caveat here: although the actual gold thickness of the SRR metamaterial is sub-micron (100nm gold layer), the *effective thickness* (normal to sample surface) of the metamaterial layer has been shown to be nearly equal to the (in plane) periodicity of the SRR array [20, 37, 51]. This effective thickness is a manifestation of the distance SRR induced electromagnetic fields extend above and below the SRR plane (**P**=36μm → z_{eff} = ±18 μm). This also means there is an apparent overlap of the naively defined metamaterial and BCB layers. In truth, BCB in the immediate vicinity of the gold rings directly modifies the local fields of the SRR, and thus becomes *part* of the optical constants of the metamaterial layer – rather than acting as its own separate layer. The effect of the BCB layer should not be "double counted", and we perform our analysis as such.

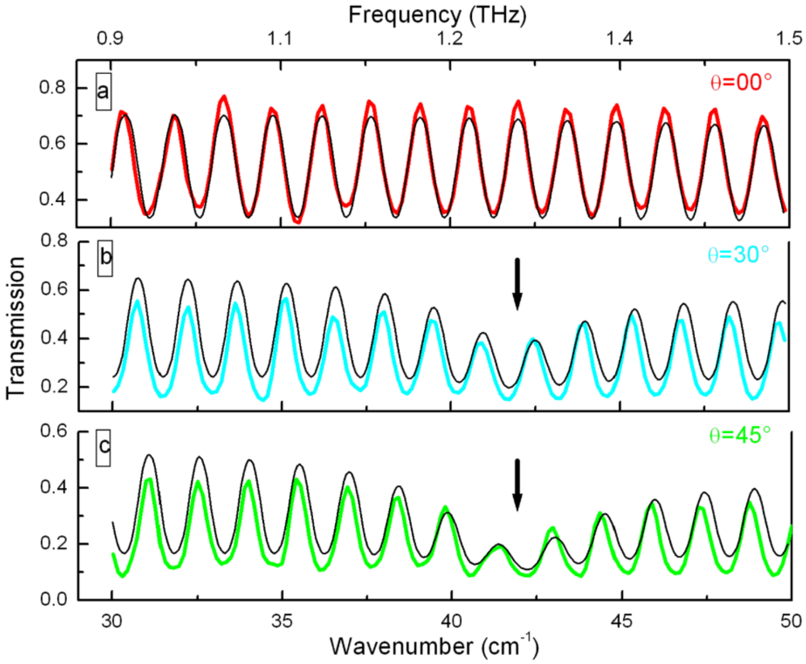

Figure III-7 High-resolution (0.25cm^{-1} [0.008 THz]) experimental transmission spectra (colored bold lines) zoomed on the magnetic resonance for θ=0° (a), 30° (b), and 45° (c). Fresnel transmission fit (thin black lines) for each angle. The position of the magnetic oscillator interaction is indicated by a black arrow.

We can find an analytical expression for the transmitted power through the multilayer sample by enforcing the continuity of the tangential components of the electric field, E, and magnetic intensity at the boundaries of each of the layers. The full expression for transmitted power is quite long, but the procedure for derivation can be found in [Ref. 22], and in other standard texts [42, 52]. The total transmission is a function of the layer thicknesses, the isotropic dielectric constant of the silicon, and of the in-plane electric and out-of-plane magnetic response of the metamaterial+BCB layer [17]. Far infrared values for the dielectric value of lightly doped silicon ($\varepsilon_{(si)}=11.9+0.01\cdot\mathbf{i}$) [53] and BCB ($\varepsilon_{(BCB)}=2.5+0.01\cdot\mathbf{i}$) [54] can be readily found in literature. A control measurement on a silicon only sample using these literature values shows that our derived Fresnel transmission formula accurately models the observed transmission through natural material layers. For the frequencies treated in our analysis (those shown in Figure III-7), the higher-frequency dynamic aspects of the electric resonance can be ignored, and we reduce its full interaction $\varepsilon_x(\omega)$ to a static dielectric constant of the metamaterial layer ε_x.

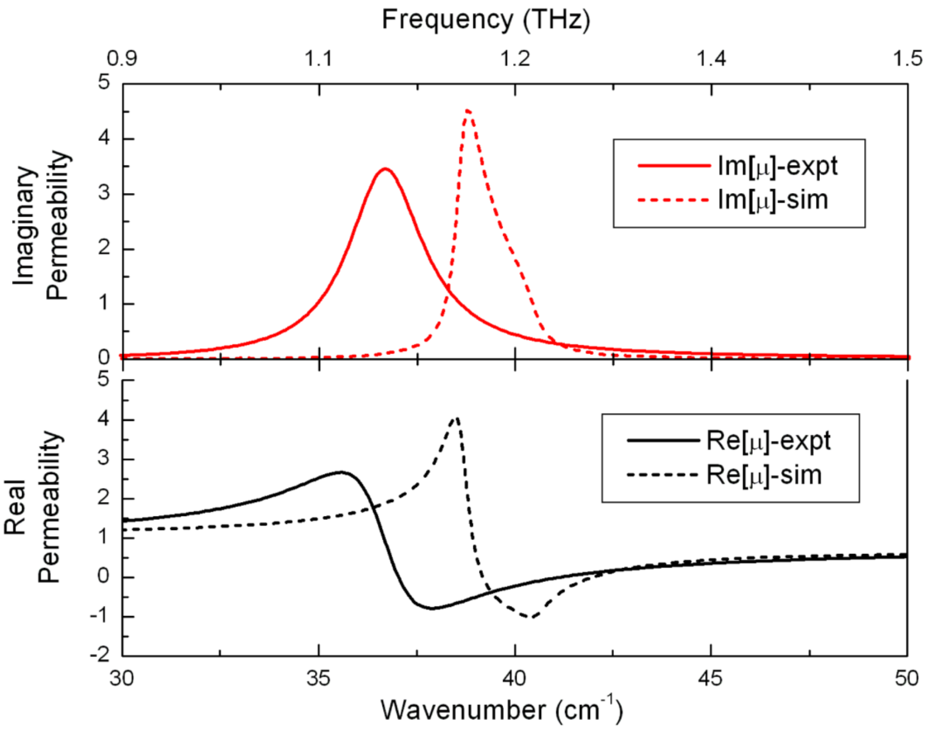

Figure III-8 Metamaterial out-of-plane magnetic permeability for the Fresnel fit to experimental data (solid lines), and S-Parameter inversion of the simulated structure (dashed).

The spectral features arising from the metamaterial resonances are intertwined with the substrate fringes, and extracting the metamaterial ε_x and μ_z would in principle require fitting to all parameters of the system simultaneously across the angle datasets, and repeating this at each frequency. Fortunately, in this work our primary interest is limited to the magnetic out-of-plane resonance, whose form for the SRR medium is known to be well approximated by the modified lorentzian-oscillator function given in Eq. III-6 [19, 43, 44].

This allows us to make a great computational simplification: collapsing a fitting calculation at every frequency point into a single fitting for the oscillator parameters in this frequency dispersion relation. Additionally, $\mu_x = \mu_y = 1$, since only z-components of the B-field couple to the metamaterial. To generate a fit of our model to the data, we iteratively vary the fitting parameters for layer thicknesses, substrate and metamaterial dielectric values, and parameters in Eq.III-6 to find the set of values which gives the best *simultaneous* fit to the data at 0^o, 15^o (not plotted), 30^o and 45^o. A rough fit is found by hand, and then a chi-squared method is used to optimize. The real and imaginary metamaterial permeability generated from the fitted values of Eq. III-6 are shown in Figure III-8 (solid lines). This result for permeability has a standard resonant form, becoming negative over a short range and reaching a maximally negative value of μ_z=-0.8 at 1.15 THz. The fitted values for the metamaterial and silicon layer thicknesses are d_{MM}=37μm, d_{Si}=970μm, respectively.

To check that our fit-resultant permeability curve is reasonable, we perform a finite element simulation of our metamaterial, using the commercial software Microwave Studio. The simulation generates reflection/transmission phase/amplitude as output, and as such is suitable for the standard S-parameters retrieval [55]. The possible geometry of the simulation restricts **k** to one of the XYZ axes, so we simulate the equivalent of θ=90º, though this cannot in practice be performed experimentally. The permeability dispersion recovered from the S-parameter retrieval of the simulation results is overlaid in Figure III-8 (dashed lines). We find that the simulation results are in general agreement, but exhibit a sharper resonance than in our experimental sample. This is not surprising given the dampening effect of small sample defects [56]. The simulation-retrieved permeability peak also occurs at a higher frequency, by a couple wavenumbers. Some of this discrepancy may be due to the necessary assumptions made in finite element simulations, such as partial substrate inclusion. More importantly, it is known that the finite size of our unit cells will cause problems here. Our Fresnel formulation, and thus our experimentally retrieved permeability fit, relies on the validity of our assumption that the SRRs act together as a continuous material which supports collective modes – rather than as individual antenna-responses which occur in sparse arrays. Our sample has a periodicity only 7 times smaller than the resonant wavelength. Although this is typical for a metamaterial, previous work has indicated that the discrete medium only becomes truly indistinguishable from a continuous material at ~λ/30 [30]. Our model thus neglects any effects due to spatial dispersion in our metamaterial layer – which are certainly present given the known phase inhomogeneities [20] within the unit cells. The perfect modeling approach in these conditions would be a hybrid of the Fresnel theory appropriate for continuous materials and of the antenna theory used for discrete elements.

III.D.3 Summary of infrared characterization

To summarize, we have measured far-infrared transmission through a split-ring-resonator metamaterial sample with an artificial (tailored) magnetic resonance. Using the Fresnel formulas and assuming an effective medium, we have formed an expression for transmission through this metamaterial at any acute angle of incidence in terms of the magnetic resonance and the other material parameters. These parameters are simultaneously fit to experimental data at four angles, which helps to compensate for the lack of phase information. The experimentally measured permeability reaches a value of negative 0.8 on resonance. We see moderately good agreement for the retrieved permeability with that predicted by finite element simulation of the structure; there are known inaccuracies in both methods, and the true response may lay somewhere inbetween. The ability to couple quantitative experimental measurements with simulation predictions is an important tool in the ongoing research into metamaterials, and the Fresnel analysis presented here helps make this possible in frequencies ranges or situations where phase information may not be available.

IV. Dynamic tuning using a vanadium-dioxide hybrid-metamaterial

IV.A Introduction to hybrid metamaterials

Advances in the recently emerging field of metamaterials include the development and demonstration of devices for sub-wavelength imaging [57], cloaking [9], ultrafast optoelectric switching [58], and more. So far, these devices have most often relied on geometrically-fixed electromagnetic resonances which restrict operation to a single frequency [11, 59] or narrow band [27]. Real-time tuning of the resonant response is one possible way to overcome limitations of bandwidth. Demonstrations of tuning have been made for microwave frequencies using integrated RF electrical components [60]. At infrared and higher frequencies such components are unavailable, and other means of implementing tuning must be found [61]. Our hybrid Split Ring Resonator Vanadium Dioxide (SRR-VO_2) device accomplishes this, granting a resonance tuning range of 20% or more.

Our device is made of 100nm thick gold Split Ring Resonators (SRRs) lithographically fabricated on a 90nm layer of VO_2 Figure IV-1b. The VO_2 is grown on a sapphire substrate, and thoroughly characterized by electrical [62] and optical measurements [63]. VO_2 undergoes a thermally-triggered insulator-to-metal phase transition [64] that corresponds to a four orders-of-magnitude change in conductivity. The SRR is the most common and best characterized implementation of electromagnetic metamaterials [22, 65]. It responds resonantly to in-plane electric fields, and out-of-plane magnetic fields. The way which the VO_2 and SRR layers interact is what makes this hybid-metamaterial interesting. The thickness of the lithographic gold comprising the SRR's and of the VO_2 film, are both much less than the in-plane periodicity of the SRR array (20 μm). Metamaterial arrays like this have been shown to form *effective* material layers whose electromagnetic thickness is approximately the period of the array [22, 23] – rather than the physical thickness of the lithographic gold. In this arrangement, the local electromagnetic fields of the SRR overlap the VO_2 layer (see Figure IV-1c) and intertwine with the VO_2 material response. The VO_2 film thus becomes part of this effective material layer - due to its proximity to the SRRs and thin size compared to the array periodicity. Together they form a hybrid metamaterial – blending the properties of the VO_2 film with those of the discrete SRR array.

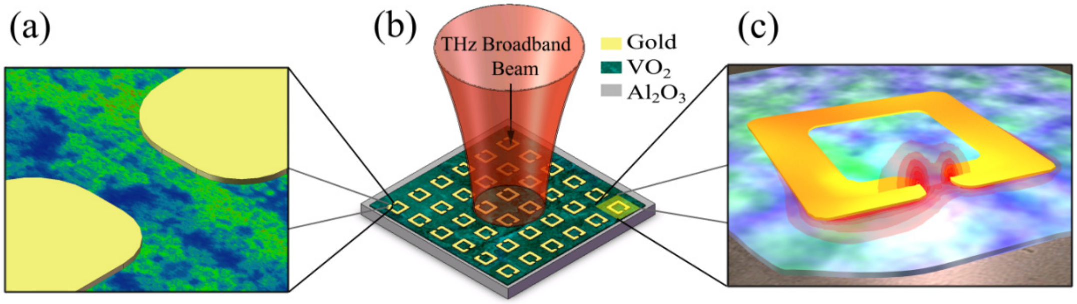

Figure IV-1 Sketch of the Vanadium Dioxide Split Ring Resonator hybrid-metamaterial. **a,** Close-up of the SRR gap, sketched on top of a near-field image of a VO_2 film during phase transition. This comparison illustrates that the VO_2 percolating metallic grains (green) which emerge from the insulating host (blue) are much smaller than SRR gap. The sSNIM data is taken at 342 Kelvin. **b,** Device layout and experimental setup. Gold Split-Ring Resonators of period 20 μm are lithographically fabricated above a 90nm thick VO_2 layer, which has been grown on Sapphire. The resonance of this hybrid-metamaterial device is probed in transmission. **c,** Blow-up of a single SRR with electric field total amplitude results from a numerical solver, illustrating the overlap of the SRR fields with the VO_2 film.

The resonance frequency of the SRR metamaterial is highly sensitive to the dielectric property of any material placed nearby - especially in the vicinity of the SRR gaps [66, 67]. This circumstance, along with the following distinctive dielectric property of VO_2, enables the realization of a dynamically tunable SRR-VO_2 hybrid metamaterial. Near its insulator-to-metal phase transition, VO_2 exhibits a divergent *bulk* permittivity (see Figure IV-2c). This modifies the local fields of the SRR within and around the gap-region, acting like a tunable dielectric inside a capacitor. The resonance frequency of the SRR-VO_2 hybrid system is expected to decrease as VO_2 permittivity increases: a behavior anticipated from a simplified RLC-circuit model of SRR response. Experimental results as well as numerical simulations - reported below - substantiate these expectations.

The observed divergent permittivity of the bulk VO_2 is understood to be due to the percolative nature of the phase transition. During its phase transition, VO_2 exhibits the emergence and growth of tiny (5-10nm) metallic puddles in the insulating host. Figure IV-1a illustrates this, showing near-field composition data obtained via scattering Scanning Near-Field Infrared Microscope (sSNIM), overlaid on a sketch of the 3μm wide SRR gap. The *effective medium* response of the bulk VO_2 material comprised of these metal and insulating puddles is described by the Bruggeman model [63]. This arrangement is interesting as it utilizes one nanometer-scale effective-medium (formed by metallic puddles in the VO_2 insulation host) within another micron-size effective-medium (formed by the periodic SRR array).

IV.B Experiment and performance

To probe this hybrid metamaterial, we perform Fourier Transform Infrared (FTIR) transmission spectroscopy of our device. We focus normally incident linearly polarized light from a mercury lamp onto a $3mm^2$ spot on the $1cm^2$ sample. The SRR array is oriented such as to electrically excite the SRR by taking advantage of the asymmetrical in-plane dipole moment created in the long and short legs of the SRR [47, 68]. In Figure IV-2, we display transmission spectra for our hybrid device. The room temperature spectra reveal a strong resonance with a peak at 55cm^{-1} (see Figure IV-2a). As we increase the temperature of the device, VO_2 begins its transition and the observed resonance peak frequency decreases. The tuning of the resonance through the VO_2 transition can be mapped by plotting the peak frequency versus temperature (see Figure IV-2b). The resultant curve shows a sharp onset of tuning as the phase transition begins, and has a tuning range of 20%. At temperatures above 342 Kelvin, the resonance becomes heavily damped due to the increasing conductivity of the VO_2 layer. Eventually, the interconnecting VO_2 metallic puddles electrically short the SRR entirely, giving us the ability to turn the resonance off at temperatures above 343 Kelvin.

We substantiate our experimental observations numerically - simulating the metamaterial using the finite-integration time-domain code package Microwave Studio by CST, Inc. All three constituents of the device (SRR, VO_2, Al_2O_3 substrate) are included. These numerical results agree very well with our experimental data - showing a room temperature resonance at 55cm^{-1}. Simulations for elevated temperatures (green tuning curve shown in Figure IV-2b) use Bruggeman permittivity values for VO_2 taken from Figure IV-2c [63], and also agree well with experimental data. The accuracy of Microwave Studio for the prediction of the resonance tuning of our SRR-VO_2 hybrid is important, since metamaterial design advances are largely reliant on such numerical simulators.

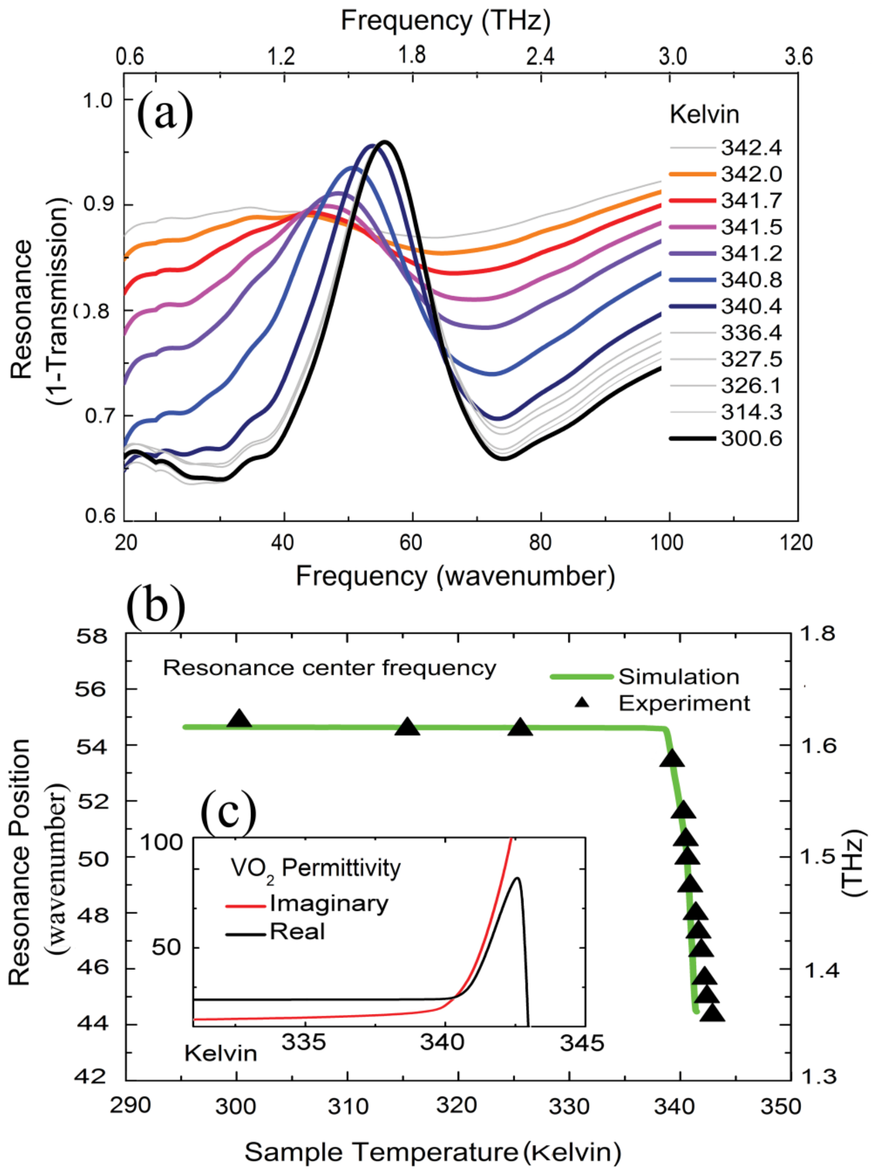

Figure IV-2 Dynamic tuning of the SRR ω_0-resonance. **a**, Transmission spectra through the hybrid metamaterial device at increasing sample temperatures. The resonance frequency decreases by nearly 20% as the vanadium dioxide passes through its metal insulator transition. **b**, Resonance frequency as a function of temperature. **c**, (insert) VO_2 Bruggeman effective-medium permittivity.

In order to evaluate the parameter space where this hybrid-metamaterial device enables tunable electromagnetic properties, we retrieve permittivity and permeability values for the metamaterial layer. This is done by modeling the transmission through a 2-layer device (hybrid SRR+VO_2 and Al_2O_3 substrate) using the Fresnel equations. Electromagnetic oscillators are assigned to the material of each layer, and then fit to the observed spectra [22]. Literature values for the permittivity of Al_2O_3 are used for the substrate [69]. The oscillator used for the hybrid metamaterial layer is a modified Lorentzian - incorporating effects arising from the spatial dispersion present in the SRR array [70]. This recent advance in our fitting procedure gives a noticeable improvement in fit over previous oscillator models [71]. Figure IV-3 shows the retrieved real permittivity and permeability. The room-temperature permittivity exhibits an expected strong resonance. Room temperature permeability also shows a weak anti-resonance, even in our electric excitation configuration. This is an artifact of the SRR array periodicity; spatial dispersion acts to couple the permittivity resonance to a permeability anti-resonance. This effect has been routinely observed for metamaterials with periodicity such as ours - in the range of one-tenth of a wavelength [72].

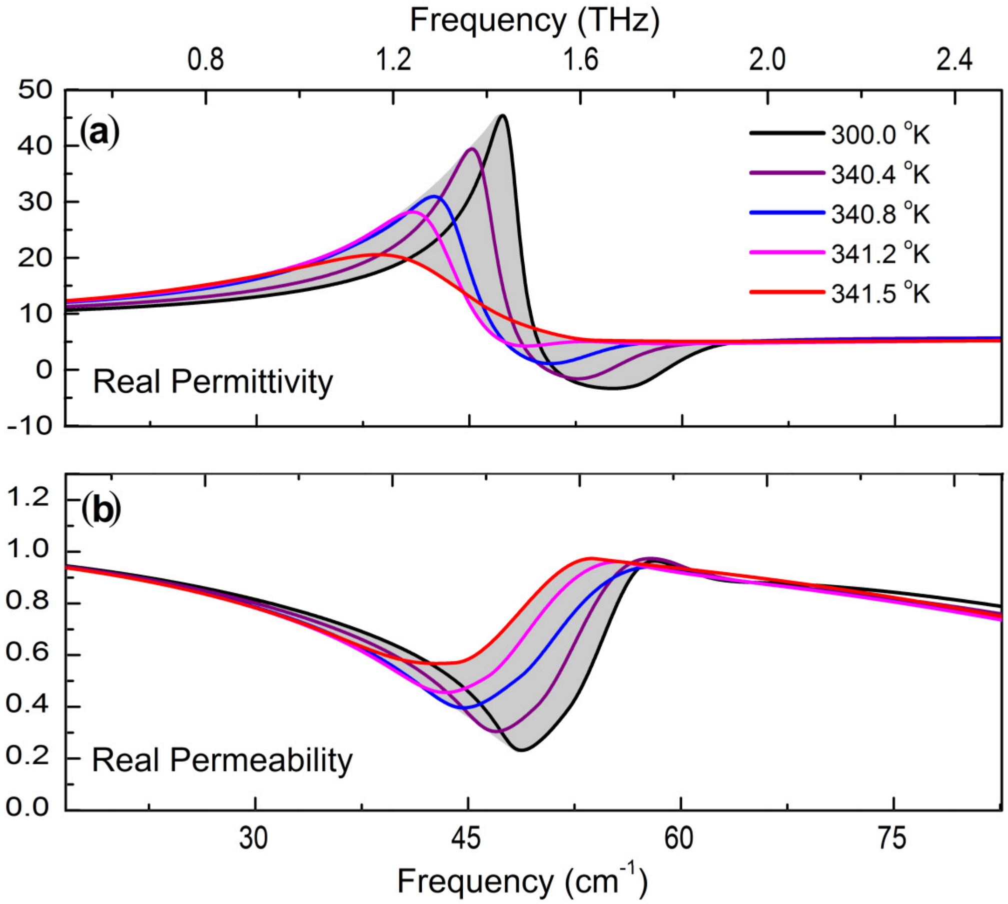

Figure IV-3 Experimentally retrieved permittivity and permeability bandwidth for the hybrid metamaterial. **a**, Permittivity values for the SRR-VO_2 hybrid layer in our device. **b**, Permeability values. The shaded area illustrates the complete range of values accessible with the help of our hybrid SRR-VO_2 device

At temperatures above 340 Kelvin, the retrieved permittivity and permeability resonances both red-shift. This frequency shift follows the transition from insulator to metal with temperature in VO_2. Losses in the VO_2 metallic puddles also dampen the resonance, decreasing the amplitude. Any non-tunable metamaterial allows only a *single* pair of permittivity and permeability curves. In contrast, the shaded area in Figure IV-3 illustrates a range of permittivity and permeability values accessible with the help of our hybrid SRR-VO_2 device. Through careful control of the sample temperature, we can select any permittivity-permeability curve-pair in the shaded region. Electromagnetic flexibility of this kind is immensely valuable in device design and operation.

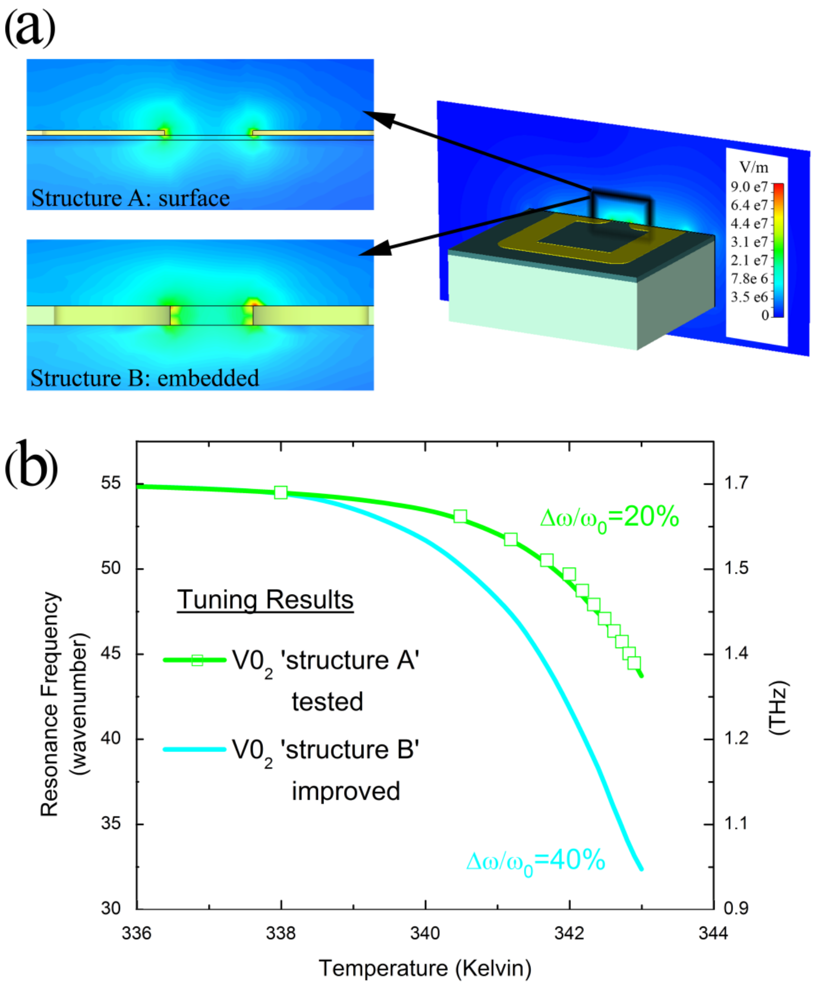

Figure IV-4 Improved design geometries for greater dynamic tuning range. **a,** RMS electric field cross-section inside the SRR split-gap for the two structures: Structure A as fabricated and tested with 100nm gold SRRs lithographically patterned on top of 90nm VO_2 film. The incident field strength is 1e6 V/m. Proposed improved structure B with gold SRRs embedded into 500nm VO_2 film. **b**, Comparison of the dynamic tuning range for devices A and B. Device B exhibits nearly twice the tunable range of device A, resulting from the thicker VO_2 and embedding - both of which increase the proportion of SRR fields contained in the VO_2 volume.

The accuracy of our oscillator-fitting model in replicating the observed experimental spectra highlights the essence of the hybrid-metamaterial approach. Physically, our gold SRR array and VO_2 film comprise two distinct layers, each only ~100nm thick. Electromagnetically, however, these two layers are exceedingly well represented by one single hybrid-material layer with combined properties of SRR and VO_2. As mentioned, the electromagnetic thickness of a SRR array is approximated by the array periodicity, which can be quite large compared to the physical thickness of the SRR. It is this large electromagnetic thickness which allows us to easily combine the properties of other nearby materials with those of the metamaterial, forming a hybrid-metamaterial.

Taking advantage of advanced device architectures can also expand the range of tuning. Figure IV-4 illustrates this, showing Microwave Studio numerical results for a proposed improved device employing embedded SRRs in thicker VO_2. This geometry places the VO_2 directly in the critical SRR split-gap region where the capacitive fields are most intense (see Figure IV-4a). This maximizes the impact each volume of VO_2. Simulation results for this improved 'structure B' show twice the useable tuning range - up to 40% resonance shift across the VO_2 transition (see Figure IV-4b).

We stress that the phase transition in VO_2 may be triggered optically[73] or electrically[74] as well as thermally, thus enabling photonic or electric control of the resonance in this hybrid metamaterial. In these cases, one can envision locally triggered sections of VO_2 for pixel-like tuning of the metamaterial.

IV.C Summary and Future Outlook

Our demonstration of dynamic tuning in this SRR-VO_2 configuration suggests the potential for rich physics and interesting effects in other hybrid-metamaterial devices employing magnetic, nonlinear, or other materials. Combining the wide ranging phenomena found in natural materials with the electromagnetic design control offered by metamaterials may act to expand the usefulness of each. The many demonstrations tunable metamaterials in a variety of incarnations which have been published since this work underscores the likelihood that tuning will become an essential part of metamaterial devices. Even for single-frequency operation, some form of tuning control exists in almost all resonant systems to facilitate dealing with temperature drift, shifting input, or other similar factors.

.

V. Metamaterial resonant-enhanced dielectric-sensors

V.A Overview of tuning and metamaterial sensors

In Chapter IV, we discussed the utility of tuning metamaterial response. There are many approaches to such tuning, and in this chapter we introduce a method for passive tuning via addition of nanocrystals to the metamaterial surface. This was developed as a tactic for adjusting the metamaterial resonance post-lithography, allowing fine adjustments to the response without requiring complete re-design of the metamaterial. This experiment in tuning also presented an unexplored use for metamaterials – as Resonant Enhanced Dielectric Sensors (REDS). Because the resonance frequency of our SRR elements is so sensitive to the presence/absence of any dielectric material (especially in the capacitive split-gap) we found the metamaterial enabled us to detect minute quantities of nanocrystal which were otherwise undetectable. In addition to tuning results, the application of metamaterials as REDS sensors is explored in this chapter.

V.B Tuning via dielectric inclusions.

Device design is quickly becoming a large part of metamaterial research. In the short half-decade since its conception, understanding of the physics behind tailored electromagnetic responses in metamaterials has progressed far enough to where application demonstrations are surfacing. Prime examples include diffraction beating lenses [28, 75], frequency selective cloaks [9, 11, 76], advanced optics [16, 27], and improved radomes [77]. Many of these devices have even been demonstrated at the microwave frequencies, due to the comparable ease of fabricating millimeter scale structures. Operation at higher frequencies - infrared and above - requires smaller structures which are usually patterned photo-lithographically. The photo-lithographic step is often a hurdle to creating new infrared devices, as it requires designing and purchasing an expensive lithography mask - which then permanently fixes the metamaterial structures. To make matters more difficult, design relies on numerical finite-element solvers; so even the best efforts in design can incorrectly predict the resonant response frequency of a structure by 5%.

Given all these considerations, it is clear that there is significant utility in developing the ability to alter the response of any metamaterial structure without the need to design a new lithography mask. Active tuning based on modification of the electromagnetic properties of materials is one option being pursued [41, 47], with clear advantages for switchable devices [61]. Despite advantages, the architecture of initial active-tunable devices has been quite complicated, and the tunable frequency range limited. In this chapter, we demonstrate a simple procedure for passive tuning of a Split-Ring Resonator (SRR) response. Although the tuning processes are certainly not limited to SRR's, we consider them exclusively here for simplicity and familiarity [71]. We use the same SRR array as in the work of [Ref. 71], wherein gold SRR's are patterned on a 1mm thick Silicon substrate coated with a thin 6μm layer of BenzoCycloButane

(BCB). The BCB is used as a low-loss adhesive aid in performing the gold lithography. A photograph of this array is shown in Figure V-1a. This metamaterial is designed to have a magnetic resonance at 1.20 THz, which experimentally manifests as a sharp dip in sample transmission, shown in Figure V-2 as the black curve. Transmission is probed in a FTIR spectrometer using s-polarized 45° incident light (see insert in Figure V-3). The details of this style of measurement, and a more in depth discussion of how this dip is known to be the magnetic mode of the SRR, can be found in [Refs 22, 71] One improvement on the method in [Ref. 71] is explored in this work: sample transmission spectra exhibit large Fabrey-Perot fringes which arise from multiple reflections within the thick 1mm silicon substrate. These fringes make precise determination of the resonance frequency difficult. The fringes have a known periodicity which is set by the thickness of the substrate, so in our analysis we pass the transmission spectra through a digital notch-filter which effectively removes this substrate periodicity from our transmission spectra. Because the notch filter is sharply targeted at the substrate fringes, no other features of the spectra are significantly affected.

Passive tuning in this chapter is accomplished by adding dielectric material to alter the capacitance of the SRR. A simplified model of the SRR can be thought of as an inductive-capacitive circuit element [78, 79]. The self-inductance of each loop battles the capacitance of the split-gap to determine the resonance frequency (and all other attributes of the resonance) following

$$\omega_0 \propto \frac{1}{\sqrt{LC}} = \frac{1}{\sqrt{L}\sqrt{\varepsilon_0 \int^{v} \varepsilon(v)E(v)dv}}.$$

The electric field falls of quickly away from the metamaterial layer so the volume integral need only be considered in vicinity of the SRR. Cross-sections of the electric field distribution for our SRR are displayed in Figure V-1g, as solved by finite element. The high intensity of the electric field **E(*v*)** near the SRR also makes the resonance frequency sensitive to small changes in the dielectric ε. To take advantage of this, we prepare a 0.2% solution of silicon nanospheres (50nm diameter - prepared by weight), suspended in ethanol using an ultrasonicator. The solution is then applied to the surface of our sample in small drops of 30μl (average drop volume as measured by weight). The sample is heated to 60° C, so shortly upon contact with the sample the ethanol evaporates leaving behind only the silicon nanospheres. This procedure produces a fairly uniform layer (a video of this puddle application and drying process is made available online at physics.ucsd.edu/~tdriscol/SiPuddle.mpeg).

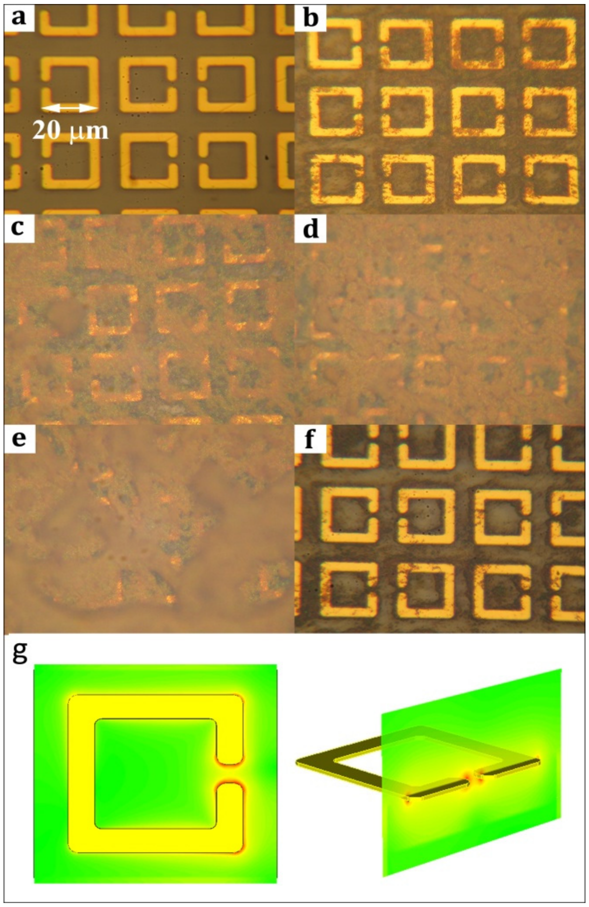

Figure V-1 Photographs of the gradual addition of silicon nanospheres by solution. Panel (f) shows the sample after removing most of the silicon by ultrasonics. Panel (g) shows cross-sections of the electric field intensity (solved by finite-integration time-domain). The scale is logorithmic, green is the incident field intensity, red is 100x incident.

Figure V-1b shows a photograph of the SRR array after this first application. The introduction of the silicon material onto the SRR effectively increases the capacitance and thus decreases the resonant frequency, and we observe that the transmission dip decreases by ~0.05 THz (Figure V-2, blue curve). Repeated applications of 30μl drops of this solution deposit additional silicon spheres (pictured in Figure V-2c - Figure V-2e) which continue to decrease the resonance in steps. Saturation of this effect begins onset by picture/line e, and further applications reduce the resonance frequency in diminishing increments. A weaker solution of nanospheres could give even finer step-sizes, and an exact resonance frequency can be created via a guess and check methodology. Another interesting effect of the silicon addition is an observed sharpening and deepening of the magnetic resonance dip. This is qualitatively sensible, as the Q factor of a LC-circuit scales inversely proportional with the resonance frequency. This sharpening is also an indicator that addition of the silicon nanospheres does not appreciably add to the damping of the SRR resonator (the nanospheres are nearly intrinsic silicon and so have a very small imaginary permittivity at THz).

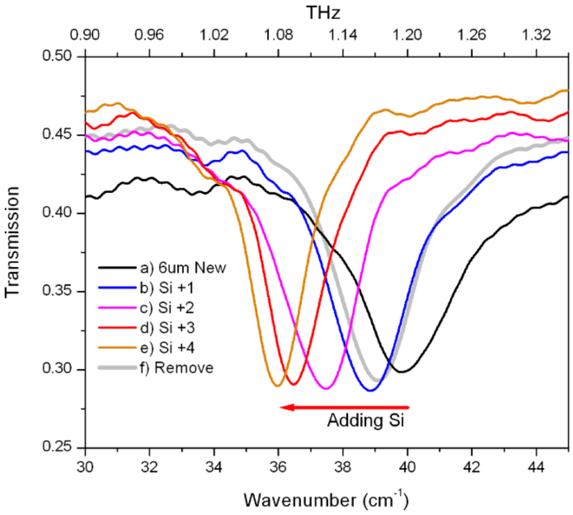

Figure V-2 Fine tuning: Addition of silicon nanospheres increases the average dielectric of the SRR capacitance and shifts the magnetic resonant frequency downwards. Thick

grey (line f) shows near-restoration of the original response by removal of the nanospheres in an ultrasonicator.

The accumulation of the silicon spheres is also demonstrated to be reversible. We briefly (30 seconds) submerge the sample in an ultrasonicated ethanol bath, and most of the silicon is removed (Figure V-2f). The resonance frequency also returns to nearly its original position. Not entirely all of the spheres come off, we see that the spheres clustered near the edges of the SRR's remain. This reversibility is an important attribute; if the desired resonant frequency is overshot we can reset and begin the application process anew.

Fine tuning by addition of silicon nanospheres gives control over the resonance frequency to within a change of ~10%, as we see in Figure V-2. We can effectively extend this tuning range by also varying the thickness of the BCB spacer used. Since BCB has a lower dielectric than silicon, a thinner BCB layer lets more of the SRR fields penetrate the silicon substrate, and lowers the resonance frequency. Figure V-3 demonstrates this, showing that using a thinner BCB layer creates a lower resonance frequency, acting as a method for coarse tuning. Finite element solvers do a moderately good job of predicting the result of such homogeneous material inclusions, but combined with imprecision in the BCB spin-coating process we are limited to a coarse tuning accuracy of 5-10%. By combination of the coarse and fine tuning methods, however, we can feasibly create a metamaterial with a precise magnetic resonance frequency anywhere over the octave from .70 THz to 1.20 THz – all without needing to change the feature sizes.

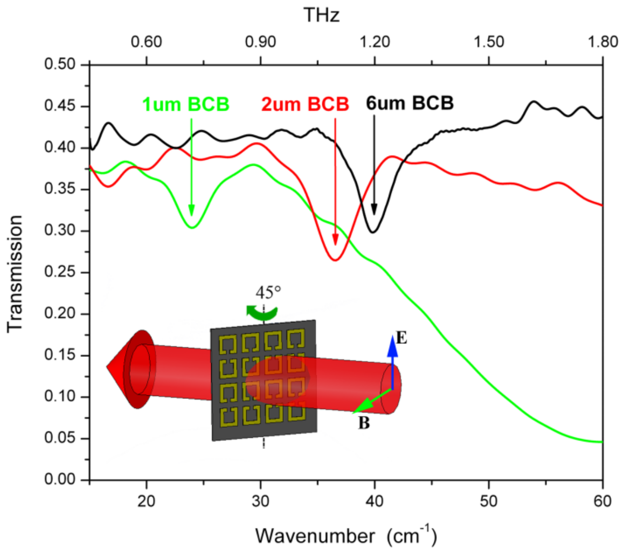

Figure V-3 Coarse tuning: Three values of the BCB spacer thickness showing thinner spacers decrease the SRR magnetic resonance position.

V.C Sensing dielectric inclusions.

The observed sensitivity of the SRR metamaterial resonance frequency to the introduction of microliters of our weak solution of silicon nanoparticles also brings to mind the possibility of using such devices as sensors. The 30μl drop contains an amount of silicon so small (less than 1nanogram) that we cannot detect it via standard transmission-amplitude experiment [80]. The Silicon layer which we add to the SRR's is many times thinner than the skin depth of THz radiation. Using our SRR array as a probe offers a significant advantage, as we transform detection of the silicon nanospheres from a transmission-amplitude level measurement to a resonance frequency position measurement, which is often much more accurate [81]. The resonance frequency of the SRR is sensitive to very small quantities of material as the concentrated fields within the split-gap most strongly interrogate a volume of only $(\sim 3\text{um})^3$. A cleverly designed sensor might thus be able to detect quantities of material several orders of magnitude less than that demonstrated in this chapter, and we are in the process of designing and investigating such devices.

Appendix A. MATLAB metamaterial optical-constants retrieval program

A.1 Program-flow outline

1. **AnisoFresnel.m** Starts GUI, parses input strings containing experimental data
    a. *Calls* **InteractiveEpMu.m** and passes all control.
2. **InteractiveEpMu.m.** contains all GUI callbacks. Handles fitting calculations and bulk of routine
    a. Initializes global variables and structures.
    b. *Calls* **Initialize_Material_Tensors.m** which sets up Permittivity and Permeability global tensors, and oscillator structure pointers.
    c. Recieves setup parameters (# layers, polarization) from GUI.
    d. Recieves user-input for number and type of oscillators, and first guess at parameters.
    e. *Calls* **OscLookup.m** which builds and returns the requested dispersion relation for an oscillator.
        i. Repeats 2.e for each oscillator needed
    f. *Calls* **TELookup.m** which returns the functional form for transmission and reflection in the given setup.
    g. Solves for numerical transmission and reflection values by inserting permttivity and permeability dispersion-tensors for each layer into transmission/reflection equations.
    h. Plots solved numerical values with experimental values.
        i. 2.d through 2.h are repeated until user guesses are close.
    i. Performs least-squares fitting, adjusting oscillator values (2.d), and repeating 2d-2i.
3. **InteractiveEpMu** returns control and final solution to **AnisoFresnel.m,** which parses the global handles structure, and saves needed information.

A.2 Full Code.

function handles = AnisoFresnel

```
    global i deg c;
    i = sqrt(-1);
    deg = pi/180;
    c = 3e10;
    Pi=3.14159;
% definitions and setups
    Trans00deg = 0;     wAHz = 0;
%    Trans00degDEFAULT = load('Smoothed00.DAT');
      wAHz = load('Freq.DAT');              %frequency data
%    wAHz = LoadData('Load Frequency Data')             % prompted load

    ExpArray{1,1} = '00';   %angle of data
    ExpArray{2,1} = load('VO2t2950k.DAT');
    ExpArray{3,1} = 'T';    %type (R or T)

    ExpArray{1,2} = '00';   %angle of data
    ExpArray{2,2} = load('VO2t3404k.DAT');
    ExpArray{3,2} = 'T';    %type (R or T)

    ExpArray{1,3} = '00';   %angle of data
    ExpArray{2,3} = load('VO2t3408k.DAT');
    ExpArray{3,3} = 'T';    %type (R or T)

    ExpArray{1,4} = '00';   %angle of data
    ExpArray{2,4} = load('VO2t3412k.DAT');
    ExpArray{3,4} = 'T';    %type (R or T)

    ExpArray{1,5} = '00';   %angle of data
    ExpArray{2,5} = load('VO2t3415k.DAT');
    ExpArray{3,5} = 'T';    %type (R or T)

    ExpArray{1,6} = '00';   %angle of data
    ExpArray{2,6} = load('VO2t3417k.DAT');
    ExpArray{3,6} = 'T';    %type (R or T)

%    ii=2;
%     while(0) %change this to while(1) to re-enable this block of data-loading.  as of
April'06 this routine is outdated
%         selection = questdlg('Add Another Angle Data?','More Data?','Yes','No','Yes');
%         if strcmp(selection,'No')
%             break;
%         else
%             while (1)
%                 ThetaPick=inputdlg('Input Single Angle', 'Theta', 1);
%                 if isnan(str2double(ThetaPick))
%                     set(hObject, 'String', 0);
%                     errordlg('Input must be a number','Error');
%                 else
%                     break;
%                 end
%             end
%             ii=ii+1;
%             ExpArray{ii}=ThetaPick;
%             ii=ii+1;
%             ExpArray{ii} = LoadData(strcat('Load  :', ThetaPick{1}, ' DegreesData'));
%         end
%     end
%

    d=1;
%        [PermittivityPars,Permittivity] = FitEpxToZeroDeg(wAHz,Trans00deg);
    handles = InteractiveEpMu('EpxMuzInteractive','wAHz',wAHz,'ExpArray',ExpArray);

%loads data with GUI and interfaces
```

```
function out = LoadData(PopupString)
    global i deg
    PopupNotice('Popup','Text',PopupString,'buttontext','Ready');
    out = uiimport                    % import Frequency array


function Trans = TransSingle(w,theta,Amu,Gmu,wpmu,womu,Aep,Gep,wpep,woep,epxinf,d)
    % this function computes transmission using a standard lorentzian for epx and muz all
in one go.
    global i deg c

function [PermittivityPars,Permittivity] = FitEpxToZeroDeg(wAHz,Trans00deg,d)
    % this function uses a special muz=1 form of Trans to fit a epx response to normal-
indecence
    % Replaced c=>3.0e+10, muz=>1, theta=>0,
    % PopupNotice('Popup','Text','Load Freq Data','buttontext','Ready');
    % first we need to get good initial guesses.  the best way to do this
    % is to open a window where the user can play with the parameters
    varargout = InteractiveEpx('EpxInteractive','wAHz',wAHz,'Trans00deg',Trans00deg)
    permittivity = varargout.epx;
    LorentzPermittivity = varargout.Lorentz;
```

function handles = Initialize Material Tensors(hObject, eventdata, handles);

```
% ----------------------------------------
    %initialization of GUI parameters so _Callback isn't necessary.

    %These hold the material parameters.  the 3x3 cell contains the 9
    %tensor elements, Epxx,Epyy,Epxy,etc.
    % within this tensor element, if there is  more than one oscillator
    % the tensor element is a cell array, containing the multiple
    % oscillators.

    % The oscillator parameters available are (A,G,wo,wp,inf), and values for these are
the first 5 entries in any material parameter cell element.
    % The last (sixth) entry in a material parameter cell element is a
    % anonymous function which gives an anonymous function for how these 5 parameters
used,
    %    ie. a Lorentzian would read:  @(w,A,G,wo,wp,inf) inf + A*wp^2 /(w^2 - wo^2 +
i*G*w).
    % Transmission function also knows about standard terms 'Lorentzian'.

    % SETUP OSCILLATORS AND OSCILLATOR SELECTOR
    % ------------------------------

    handles.L1.EpOscPar = cell(3,3);
    handles.L1.MuOscPar = cell(3,3);
    handles.L2.EpOscPar = cell(3,3);
    handles.L2.MuOscPar = cell(3,3);
    handles.L3.EpOscPar = cell(3,3);
    handles.L3.MuOscPar = cell(3,3);

    OscDefault   = {0,1,1,1,1,1,0 ,0,0,0, 0,1,1,1,1,1,0};
    MuOscDefault = {0,1,1,1,1,1}
    for ii=1:3
        for jj=1:3
            handles.L1.EpOscPar{ii,jj} = OscDefault;
            handles.L1.MuOscPar{ii,jj} = OscDefault;
            handles.L2.EpOscPar{ii,jj} = OscDefault;
            handles.L2.MuOscPar{ii,jj} = OscDefault;
            handles.L3.EpOscPar{ii,jj} = OscDefault;
            handles.L3.MuOscPar{ii,jj} = OscDefault;
        end
    end
    handles.L1.d = 0.0020;
    handles.L2.d = 0.1;
    handles.L3.d = 0.1;
    handles.ResLimit = 0.008;
    handles.L1.Periodicity = 20e-4;  %in centimeters
    handles.L2.Periodicity = 0;
    handles.L3.Periodicity = 0;
    handles.Dispersion.Logical = 0;
    set(handles.L1d, 'String', handles.L1.d);
    set(handles.L2d, 'String', handles.L2.d);
    set(handles.L3d, 'String', handles.L3.d);

                                % default values set above.  only need to modify those
osc's which are of
                                % particular interest to have a non-trivial initial value
                                % [A,      G,     wo,     wp, inf, 'function']   sets
default values.
%% Layer 1, Oscillator setup
    handles.L1.EpOscPar{1,1}(1:6) = {  1,   5e11,  1.5e12,   3e12,    4,  6}   % L1.Epxx
    if (get(handles.EL1isotropic, 'Value'))
        handles.L1.EpOscPar{2,2} = handles.L1.EpOscPar{1,1};
        handles.L1.EpOscPar{3,3} = handles.L1.EpOscPar{1,1};
    else
        handles.L1.EpOscPar{2,2}(1:6) = {   0,      1,      1,      1,    1,  1}   %
L1.Epyy
        handles.L1.EpOscPar{3,3}(1:6) = {   0,      1,      1,      1,    1,  1}   %
L1.Epzz
    end                         % [A,      G,     wo,     wp, 'function']   sets default
values.
    handles.L1.MuOscPar{1,1}(1:6) = {   0,      1,      1,      1,    1,  2}   % L1.Muxx
    if (get(handles.ML1isotropic, 'Value'))
        handles.L1.MuOscPar{2,2} = handles.L1.MuOscPar{1,1};
        handles.L1.MuOscPar{3,3} = handles.L1.MuOscPar{1,1};
    else
        handles.L1.MuOscPar{2,2}(1:6) = {   0,      1,      1,      1,    1,  2}   %
L1.Muyy
        handles.L1.MuOscPar{3,3}(1:6) = {   0,      1,      1,      1,    1,  2}   %
L1.Muzz
    end

%% Layer 2, Oscillator setup
    handles.L2.EpOscPar{1,1}(1:6) = {   0,      1,      1,      1,  9.9,  1};   % L2.Epxx
    if (get(handles.EL2isotropic, 'Value'))
        handles.L2.EpOscPar{2,2} = handles.L2.EpOscPar{1,1};
        handles.L2.EpOscPar{3,3} = handles.L2.EpOscPar{1,1};
    else
        handles.L2.EpOscPar{2,2}(1:6) = {   0,      1,      1,      1,    1,  1};   %
L2.Epyy
        handles.L2.EpOscPar{3,3}(1:6) = {   0,      1,      1,      1,    1,  1};   %
L2.Epzz
    end
    handles.L2.MuOscPar{1,1}(1:6) = {   0,      1,      1,      1,    1,  1}   % L2.Muxx
    if (get(handles.ML2isotropic, 'Value'))
        handles.L2.MuOscPar{2,2} = handles.L2.MuOscPar{1,1};
        handles.L2.MuOscPar{3,3} = handles.L2.MuOscPar{1,1};
    else
        handles.L2.MuOscPar{2,2}(1:6) = {   0,      1,      1,      1,    1,  1};   %
L2.Muyy
        handles.L2.MuOscPar{3,3}(1:6) = {   0,      1,      1,      1,    1,  1};   %
L2.Muzz
    end

%% Layer 3, Oscillator setup
    handles.L3.EpOscPar{1,1}(1:6) = {   0,      1,      1,      1, 11.9,  1}   % L3.Epxx
    if (get(handles.EL3isotropic, 'Value'))
        handles.L3.EpOscPar{2,2} = handles.L3.EpOscPar{1,1};
        handles.L3.EpOscPar{3,3} = handles.L3.EpOscPar{1,1};
    else
```

```
        handles.L3.EpOscPar{2,2}(1:6) = {   0,      1,      1,      1,    1,  1};   %
L3.Epyy
        handles.L3.EpOscPar{3,3}(1:6) = {   0,      1,      1,      1,    1,  1};   %
L3.Epzz
    end
    handles.L3.MuOscPar{1,1}(1:6) = {   0,      1,      1,      1,    1,  1}   % L3.Muxx
    if (get(handles.ML3isotropic, 'Value'))
        handles.L3.MuOscPar{2,2} = handles.L3.MuOscPar{1,1};
        handles.L3.MuOscPar{3,3} = handles.L3.MuOscPar{1,1};
    else
        handles.L3.MuOscPar{2,2}(1:6) = {   0,      1,      1,      1,    1,  1};   %
L3.Muyy
        handles.L3.MuOscPar{3,3}(1:6) = {   0,      1,      1,      1,    1,  1};   %
L3.Muzz
    end

    set(handles.OscAmpMu, 'String', handles.L1.MuOscPar{3,3}{1});
    set(handles.GMu,      'String', handles.L1.MuOscPar{3,3}{2});
    set(handles.omega0Mu, 'String', handles.L1.MuOscPar{3,3}{3});
    set(handles.omegapMu, 'String', handles.L1.MuOscPar{3,3}{4});
    set(handles.OscAmpEp, 'String', handles.L1.EpOscPar{1,1}{1});
    set(handles.GEp,      'String', handles.L1.EpOscPar{1,1}{2});
    set(handles.omega0Ep, 'String', handles.L1.EpOscPar{1,1}{3});
    set(handles.omegapEp, 'String', handles.L1.EpOscPar{1,1}{4});
    set(handles.epinf,    'String', handles.L1.EpOscPar{1,1}{5});

    handles.EActive = 'handles.L1.EpOscPar{1,1}';
    handles.MActive = 'handles.L1.MuOscPar{3,3}';

    handles.Fit.MaxIter = str2double(get(handles.FitIter,'String'));
    handles.Fit.Bound = str2double(get(handles.Bound,'String'));
    handles.Fit.FitType = 'T';
    handles.Plot.EpxMax = str2double(get(handles.EpxPlotMax, 'String'));
    handles.Plot.EpxMin = str2double(get(handles.EpxPlotMin, 'String'));
    handles.Plot.MuzMax = str2double(get(handles.MuzPlotMax, 'String'));
    handles.Plot.MuzMin = str2double(get(handles.MuzPlotMin, 'String'));
    handles.Plot.TransMax = str2double(get(handles.TransPlotMax, 'String'));
    handles.Plot.TransMin = str2double(get(handles.TransPlotMin, 'String'));
    handles.Plot.ColorArray = {'black', 'red', 'blue', 'magenta','green', 'cyan',
'yellow'};
    handles.Plot.Toffset = str2double(get(handles.Toffset, 'String'));
    handles.Setup.Layers = 3;
    set(handles.L2d, 'visible', 'on');
    set(handles.text22, 'visible', 'on');
    set(handles.L3d, 'visible', 'on');
    set(handles.text23, 'visible', 'on');
    handles.Setup.Pol='spol';
    handles.wCalcMin = 0.5*min(handles.wExp);
    handles.wCalcMax = 2*max(handles.wExp);
    wAHzSTEP = (max(handles.wExp)-min(handles.wExp))/length(handles.wExp);  % this ensure
matching steps to data
    wCalcMin = handles.wCalcMin - mod(handles.wCalcMin - handles.wExp(1),wAHzSTEP);
    handles.wCalc = [wCalcMin : wAHzSTEP/2 : handles.wCalcMax ];
    set(handles.wSetMin,'Value',handles.wCalcMin);
    set(handles.wSetMax,'Value',handles.wCalcMax);
    set(handles.wSetMin,'String',num2str(handles.wCalcMin,'%11.2g'));
    set(handles.wSetMax,'String',num2str(handles.wCalcMax,'%11.2g'));
    handles.Setup.TorR = {'T','T','T','R','R'};

    set(handles.EpsBreakSlider, 'Min', 0);  set(handles.EpsBreakSlider, 'Max', 1);
    set(handles.MueBreakSlider, 'Min', 0);  set(handles.MueBreakSlider, 'Max', 1);
    handles.Dispersion.EpsBreakPoint = handles.wCalcMax * 0.5;
    handles.Dispersion.MueBreakPoint = handles.wCalcMax * 0.5;
    set(handles.EpsBreakSlider, 'Value', 0.5);
    set(handles.MueBreakSlider, 'Value', 0.5);
    handles.L1.Disorder.Fraction = str2num(get(handles.L1dVar, 'String'));
    handles.L1.Disorder.NumSumPoints = str2num(get(handles.NumA, 'String'));
    handles.L2.Disorder.Fraction = str2num(get(handles.L2dVar, 'String'));
    handles.L2.Disorder.NumSumPoints = str2num(get(handles.NumB, 'String'));
    handles.L3.Disorder.Fraction = str2num(get(handles.L3dVar, 'String'));
    handles.L3.Disorder.NumSumPoints = str2num(get(handles.NumC, 'String'));

    if(size(handles.ExpDataArray,2)>=1)
set(handles.checkbox1,'String',[handles.ExpDataArray{1,1},'
',handles.ExpDataArray{3,1}]);
    else(set(handles.checkbox1,'Visible','off'));
set(handles.checkbox1b,'Visible','off'); end;
    if(size(handles.ExpDataArray,2)>=2)
set(handles.checkbox2,'String',[handles.ExpDataArray{1,2},'
',handles.ExpDataArray{3,2}]);
    else(set(handles.checkbox2,'Visible','off'));
set(handles.checkbox2b,'Visible','off'); end;
    if(size(handles.ExpDataArray,2)>=3)
set(handles.checkbox3,'String',[handles.ExpDataArray{1,3},'
',handles.ExpDataArray{3,3}]);
    else(set(handles.checkbox3,'Visible','off'));
set(handles.checkbox3b,'Visible','off'); end;
    if(size(handles.ExpDataArray,2)>=4)
set(handles.checkbox4,'String',[handles.ExpDataArray{1,4},'
',handles.ExpDataArray{3,4}]);
    else(set(handles.checkbox4,'Visible','off'));
set(handles.checkbox4b,'Visible','off'); end;
    if(size(handles.ExpDataArray,2)>=5)
set(handles.checkbox5,'String',[handles.ExpDataArray{1,5},'
',handles.ExpDataArray{3,5}]);
    else(set(handles.checkbox5,'Visible','off'));
set(handles.checkbox5b,'Visible','off'); end;
    if(size(handles.ExpDataArray,2)>=6)
set(handles.checkbox6,'String',[handles.ExpDataArray{1,6},'
',handles.ExpDataArray{3,6}]);
    else(set(handles.checkbox6,'Visible','off'));
set(handles.checkbox6b,'Visible','off'); end;

    for ii=1:3
        for jj=1:3
        out =
OscLookup(handles.wCalc,handles.L1.EpOscPar{ii,jj},handles.L1.MuOscPar{ii,jj},handles.L1.
Periodicity,handles.Dispersion,handles.L1.Disorder);
        handles.L1.Epsilon{ii,jj} = out{1};
        handles.L1.Mue{ii,jj}     = out{2};
        out =
OscLookup(handles.wCalc,handles.L2.EpOscPar{ii,jj},handles.L2.MuOscPar{ii,jj},handles.L2.
Periodicity,handles.Dispersion,handles.L2.Disorder);
        handles.L2.Epsilon{ii,jj} = out{1};
        handles.L2.Mue{ii,jj}     = out{2};
        out =
OscLookup(handles.wCalc,handles.L3.EpOscPar{ii,jj},handles.L3.MuOscPar{ii,jj},handles.L3.
Periodicity,handles.Dispersion,handles.L3.Disorder);
        handles.L3.Epsilon{ii,jj} = out{1};
        handles.L3.Mue{ii,jj}     = out{2};
        end
    end

    handles.Plot.DataShowCalc(1:6)=1;


        %note guidata does not need to be called as here handles was passed back
        %and forth using normal function arguments.
```

function Out =
OscLookup(w,EpOscPar,MuOscPar,Period,Dispersion,DisorderStruct)

```
% wvnum = w./3e10;
    if (~isstruct(Dispersion))
        clear Dispersion;
        Dispersion.Logical=0;
    end
    c=3e10;
    Disorder = DisorderStruct.Fraction;  %In fraction of center freq.
    NumSumPoints = DisorderStruct.NumSumPoints;
    EPSeff = @(w,A,G,wo,wp,inf,Period) w./w;
    MUEeff = @(w,A,G,wo,wp,inf,Period) w./w;
    EPSeffB = @(w,A,G,wo,wp,inf,Period) w./w;
    MUEeffB = @(w,A,G,wo,wp,inf,Period) w./w;
        % first do layer ELECTRIC
        switch (EpOscPar{6})
            case 1
            case 2 %Pendry  Lorentzian
               EPSeff  = @(w,A,G,wo,wp,inf,Period) (1 -(A*w.^2) ./(w.^2 - wo.^2 +
i*w*G));
               EPSeffB = @(w,A,G,wo,wp,inf,Period) (1 -(A*w.^2) ./(w.^2 - wo.^2 +
i*w*G));
            case 3 %DRS request modified Lorentzian
               EPSeff =  @(w,A,G,wo,wp,inf,Period) (inf -(A*w.^2) ./(w.^2 - wo.^2 +
i*w*G));
               EPSeffB = @(w,A,G,wo,wp,inf,Period) (inf -(A*w.^2) ./(w.^2 - wo.^2 +
i*w*G));
            case 4 %static and drude
                % e1 = 1- (wp^2)/(w^2 + tau^-2)  e2 = (1/w tau) * (wp^2)/(w^2 + tau^-2)
Added a absorption term for the al203 substrate.
               EPSadj  = @(w) i .* EpOscPar{11} .* (w ./ EpOscPar{13}).^EpOscPar{12} +
i*(EpOscPar{15}).*10.^(w./(2*EpOscPar{14}));
                %               General drude losses
Sapphire lossy phonon turn on
                %
heavyside written as (w>=EpOscPar{14})./(w)
               EPSeff  = @(w,A,G,wo,wp,inf,Period) inf.*(1*(w./w) - A.*(wp.^2)./(w.^2 +
EpOscPar{7}.^2)) + i.*(G*(w./w) + A.*(wp.^2)./(w.^2 + EpOscPar{7}.^2)) + EPSadj(w);
               EPSeffB = @(w,A,G,wo,wp,inf,Period) inf.*(1*(w./w) - A.*(wp.^2)./(w.^2 +
EpOscPar{7}.^2)) + i.*(G*(w./w) + A.*(wp.^2)./(w.^2 + EpOscPar{7}.^2)) + EPSadj(w);
            case 5 %modified static and drude
               EPSadj  = @(w) i .* EpOscPar{11} .* (w ./ EpOscPar{13}).^EpOscPar{12};
               EPSeff  = @(w,A,G,wo,wp,inf,Period) inf.*(1*(w./w) - A.*(wp)./(w+2)) +
i.*(G*(w./w) + A*(wp)./(w)) - EPSadj(w);
               EPSeffB = @(w,A,G,wo,wp,inf,Period) inf.*(1*(w./w) - A.*(wp)./(w+2)) +
i.*(G*(w./w) + A*(1./w+1).*(wp)./(w+2)) - EPSadj(w);
            case 6 % dispersive pendry lorentz for ELC from Reupong's work.
                Sb = 1; Sd = 1;
                OSCv = @(w,inf) ((2*pi*w*Period)/c) .* sqrt(inf*1)./2;
                EPSavg = @(w,A,G,wo,wp,inf) inf*(1 -(A*w.^2)./(w.^2 - wo.^2 + i*w*G));
                MUEavg = @(w,inf) sin(OSCv(w,inf)) ./ OSCv(w,inf);
                OSCa = @(w,A,G,wo,wp,inf,Period) ((i*Sd*2*pi*Period*w)/c) .*
sqrt(EPSavg(w,A,G,wo,wp,inf).*MUEavg(w,inf))/2  +  ...
                    sqrt(1-( 4 * pi^2 * Period^2 * w.^2 /c^2) .*
(MUEavg(w,inf).*EPSavg(w,A,G,wo,wp,inf))/4);
                OSCu = @(w,A,G,wo,wp,inf,Period)
(2*pi/c)*w.*Period.*sqrt(abs((MUEavg(w,inf).*EPSavg(w,A,G,wo,wp,inf))))/2;
                OSCtheta  = @(w,A,G,wo,wp,inf,Period)
2*angle(OSCa(w,A,G,wo,wp,inf,Period)) - 2*i*log(abs(OSCa(w,A,G,wo,wp,inf,Period)));
                OSCthetaI = @(w,A,G,wo,wp,inf,Period) 2*log(OSCu(w,A,G,wo,wp,inf,Period)
+ sqrt(1 + OSCu(w,A,G,wo,wp,inf,Period).^2));

                Epsadj = @(w) i.*EpOscPar{17}*(w./w) + 1 - EpOscPar{11} .* w.^2 ./ (w.^2
- EpOscPar{13}.^2 + i.*w.*EpOscPar{12}) + 1 - (EpOscPar{15} .* w.^2 ./ (w.^2 - 3.5e12^2 +
i.*w.*2e12));
%                             i*Kappa                +         A            w.^2 ./ (w.^2
- wp.^2      + i  w  Gamma      )              Higher Electric oscillator.  fixed freq.
                EPSeff = @(w,A,G,wo,wp,inf,Period) EPSavg(w,A,G,wo,wp,inf) .*
((OSCtheta(w,A,G,wo,wp,inf,Period) / 2) ./ (sin(OSCtheta(w,A,G,wo,wp,inf,Period)/2))) .*
cos(OSCtheta(w,A,G,wo,wp,inf,Period)/2).^(-Sb)   + Epsadj(w);
                MUEeff = @(w,A,G,wo,wp,inf,Period) MUEavg(w,inf) .*
((OSCtheta(w,A,G,wo,wp,inf,Period) / 2) ./ (sin(OSCtheta(w,A,G,wo,wp,inf,Period)/2))) .*
cos(OSCtheta(w,A,G,wo,wp,inf,Period)/2).^Sb;
                EPSeffB = @(w,A,G,wo,wp,inf,Period) EPSavg(w,A,G,wo,wp,inf) .*
((OSCthetaI(w,A,G,wo,wp,inf,Period)/2) ./ sinh(OSCthetaI(w,A,G,wo,wp,inf,Period)/2)) .*
cosh(OSCthetaI(w,A,G,wo,wp,inf,Period)/2).^(-Sb) + Epsadj(w);
```

```
                MUEeffB = @(w,A,G,wo,wp,inf,Period) MUEavg(w,inf)           .*
((OSCthetaI(w,A,G,wo,wp,inf,Period)/2) ./ sinh(OSCthetaI(w,A,G,wo,wp,inf,Period)/2)) .*
cosh(OSCthetaI(w,A,G,wo,wp,inf,Period)/2).^(Sb);
                EPSadjust = @(w,A2,G2,wo2,wp2,inf2) 0;
                MUEadjust = @(w,A2,G2,wo2,wp2,inf2) 0;
        end
        EpsilonA =
EPSeff(w,EpOscPar{1},EpOscPar{2},EpOscPar{3},EpOscPar{4},EpOscPar{5},Period);
        EpsilonB =
EPSeffB(w,EpOscPar{1},EpOscPar{2},EpOscPar{3},EpOscPar{4},EpOscPar{5},Period);
        MueA =
MUEeff(w,EpOscPar{1},EpOscPar{2},EpOscPar{3},EpOscPar{4},EpOscPar{5},Period);
        MueB =
MUEeffB(w,EpOscPar{1},EpOscPar{2},EpOscPar{3},EpOscPar{4},EpOscPar{5},Period);

        %Below attempts Disorder Broadening by W0resonance shifting.  No Bruggerman is
used.
           % we modify after all dispersion has happened.  dunno about this.
        EpsilonASum = 0;
        kk=(-NumSumPoints:NumSumPoints)';
        SumWo = EpOscPar{3}*(1+Disorder*kk/NumSumPoints);
        Normalize = sum( exp(-((EpOscPar{3}-SumWo)/(EpOscPar{3}*Disorder)).^2) );
        Prefactor = exp(-((EpOscPar{3}-SumWo)/(EpOscPar{3}*Disorder)).^2);
        for kki=(1:2*NumSumPoints+1)
            EpsilonAsingle(kki,:) = (Prefactor(kki)/Normalize) .*
EPSeff(w,EpOscPar{1},EpOscPar{2},SumWo(kki),EpOscPar{4},EpOscPar{5},Period);
        end
        EpsilonA = sum(EpsilonAsingle,1); %Comment this line to return to no disorder
system.  alternatively, set Disorder.Fraction to 0

        % Blending at juncture between dispersion cases.
        if(~Dispersion.Logical) % no dispersion case
            EpsilonHolder = EpsilonA;
            MueHolder     = MueA;
        else                    % dispersion case
            wBranchEps = find(w > Dispersion.EpsBreakPoint,1);   % returns index of first
case where w(index) > EpsBreakPoint
            wBranchMue = find(w > Dispersion.MueBreakPoint,1);   % kept mue blending at
eps for the moment.
            wBlend = 200;
            EpsilonHolder(1:wBranchEps-1)=EpsilonA(1:wBranchEps-1);
            MueHolder(1:wBranchMue-1)=MueA(1:wBranchMue-1);
                                 % smooth blending method based on Sin^2 + Cos^2 = 1
                                 % "factor" Varies between 1 and 0
            for ki = wBranchEps:wBranchEps+wBlend
                factor=(wBranchEps+wBlend-ki)/wBlend;
                EpsilonHolder(ki) = EpsilonA(ki) * (sin((pi/2)*factor))^2    +
EpsilonB(ki) * (cos((pi/2)*(factor)))^2;
            end
            for ki= wBranchMue:wBranchMue+wBlend
                factor=(wBranchMue+wBlend-ki)/wBlend;
                MueHolder(ki)     = MueA(ki)     * (sin((pi/2)*factor))^2    +
MueB(ki)     * (cos((pi/2)*(factor)))^2;
            end
            EpsilonHolder(wBranchEps+wBlend+1:length(w)) =
EpsilonB(wBranchEps+wBlend+1:length(w));
            MueHolder(wBranchMue+wBlend+1:length(w)) =
MueB(wBranchMue+wBlend+1:length(w));
        end

        % now do layer MAGNETIC OSCILLATORS
        switch (MuOscPar{6})
            case 1
            case 2 %Pendry  Lorentzian
               Osc = @(w,A,G,wo,wp,inf,Period) (1 -(A*w.^2) ./(w.^2 - wo.^2 + i*w*G));
            case 3 %DRS request modified Lorentzian
               Osc = @(w,A,G,wo,wp,inf,Period) (inf -(A*w.^2) ./(w.^2 - wo.^2 + i*w*G));
            case 4 %static
               Osc = @(w,A,G,wo,wp,inf,Period) (w./w)*inf+i*G;
            case 5 %defined polynomial from data
                PepsR=[9.0921e-120,-1.3447e-106,8.6821e-094,-3.2111e-081,7.5005e-069,-
1.1498e-056,1.1628e-044,-7.5734e-033,2.9814e-021,-6.1566e-010,57.463;];
                PepsI=[-1.2189e-120,1.8719e-107,-1.2664e-094,4.963e-082,-1.246e-
069,2.0911e-057,-2.3735e-045,1.7976e-033,-8.6903e-022,2.4196e-010,-29.374;];
                Osc = @(wvnum,A,G,wo,wp,inf) polyval(PepsR,wvnum) + i.*
polyval(PepsI,wvnum);
            case 6 % dispersive pendry lorentz for ELC from Reupong's work.
                % LINKED TO ELECTRIC OSCILLATOR ABOVE!
                Out
        end

% Dispersion = Osc(w, OscPars{1},OscPars{2},OscPars{3},OscPars{4},OscPars{5});
Out = {EpsilonHolder,MueHolder};
```

function Out = TELookup(layers, polarization);

```
% this function takes data on the setup of the sample, and returns an
% annonymous function which contains the expression form for the desired
% quantity.  keeps the switch and case statements out of the main loop.

l = int2str(layers);
if (polarization == 'spol') p = 's';
else p='p'; end
val = strcat(l,p);
c = 3e10;
Pi=3.14159;
Tr = 1;
Re = 1;

switch val
    case {'1s'}
Tr =
@(w,L1Epxx,L1Epyy,L1Epzz,L1Muxx,L1Muyy,L1Muzz,L2Epxx,L2Epyy,L2Epzz,L2Muxx,L2Muyy,L2Muzz,L
3Epxx,L3Epyy,L3Epzz,L3Muxx,L3Muyy,L3Muzz,theta,d1,d2,d3) ...
 -(4.*exp(-2*Pi*i.*d1.*(((w./c).*cos(theta))-(sqrt(((w./c).^2).*L1Epxx -
(((w./c).*sin(theta)).^2 ./ L1Muzz))))) .* ((w./c).*cos(theta)) .*
(sqrt(((w./c).^2).*L1Epxx - (((w./c).*sin(theta)).^2 ./ L1Muzz))) ) ...
 ./ ( exp(2.*2*Pi*i.*d1.*(sqrt(((w./c).^2).*L1Epxx - (((w./c).*sin(theta)).^2 ./
L1Muzz)))) ...
 .*(((w./c).*cos(theta)) - (sqrt(((w./c).^2).*L1Epxx - (((w./c).*sin(theta)).^2 ./
L1Muzz)))).^2 ...
 - (((w./c).*cos(theta)) + (sqrt(((w./c).^2).*L1Epxx - (((w./c).*sin(theta)).^2 ./
L1Muzz)))).^2 );

Re =
@(w,L1Epxx,L1Epyy,L1Epzz,L1Muxx,L1Muyy,L1Muzz,L2Epxx,L2Epyy,L2Epzz,L2Muxx,L2Muyy,L2Muzz,L
3Epxx,L3Epyy,L3Epzz,L3Muxx,L3Muyy,L3Muzz,theta,d1,d2,d3) ...
 (exp(2*2*Pi*i*d1*(sqrt(((w./c).^2).*L1Epxx - (((w./c).*sin(theta)).^2 ./ L1Muzz)))) - 1)
...
 .* (((w./c).*cos(theta)).^2 - (sqrt(((w./c).^2).*L1Epxx - (((w./c).*sin(theta)).^2 ./
L1Muzz))).^2)      ...
 ./ (exp(2*2*Pi*i*d1*(sqrt(((w./c).^2).*L1Epxx - (((w./c).*sin(theta)).^2 ./
L1Muzz)))).*(((w./c).*cos(theta)) - (sqrt(((w./c).^2).*L1Epxx - (((w./c).*sin(theta)).^2
./ L1Muzz)))).^2  ...
 - (((w./c).*cos(theta)) + (sqrt(((w./c).^2).*L1Epxx - (((w./c).*sin(theta)).^2 ./
L1Muzz)))).^2);

    case {'1p'}
Tr =
@(w,L1Epxx,L1Epyy,L1Epzz,L1Muxx,L1Muyy,L1Muzz,L2Epxx,L2Epyy,L2Epzz,L2Muxx,L2Muyy,L2Muzz,L
3Epxx,L3Epyy,L3Epzz,L3Muxx,L3Muyy,L3Muzz,theta,d1,d2,d3) ...
 -(4.*exp(-2*Pi*i.*d1.*(((w./c).*cos(theta))-(sqrt( (((w./c).^2).*L1Muxx -
(((w./c).*sin(theta)).^2 ./ L1Epzz)).*L1Epyy )))) .* ((w./c).*cos(theta)) .* (sqrt(
(((w./c).^2).*L1Muxx - (((w./c).*sin(theta)).^2 ./ L1Epzz)).*L1Epyy )) ) ./ (
exp(2.*2*Pi*i.*d1.*(sqrt( (((w./c).^2).*L1Muxx - (((w./c).*sin(theta)).^2 ./
L1Epzz)).*L1Epyy ))).*(((w./c).*cos(theta)) - (sqrt( (((w./c).^2).*L1Muxx -
(((w./c).*sin(theta)).^2 ./ L1Epzz)).*L1Epyy ))).^2 - (((w./c).*cos(theta)) + (sqrt(
(((w./c).^2).*L1Muxx - (((w./c).*sin(theta)).^2 ./ L1Epzz)).*L1Epyy ))).^2 );
```

```
Re =
@(w,L1Epxx,L1Epyy,L1Epzz,L1Muxx,L1Muyy,L1Muzz,L2Epxx,L2Epyy,L2Epzz,L2Muxx,L2Muyy,L2Muzz,L
3Epxx,L3Epyy,L3Epzz,L3Muxx,L3Muyy,L3Muzz,theta,d1,d2,d3) ...
 (exp(2*2*Pi*i*d1*(sqrt( (((w./c).^2).*L1Muxx - (((w./c).*sin(theta)).^2 ./
L1Epzz)).*L1Epyy ))) - 1).*(((w./c).*cos(theta)).^2 - (sqrt( (((w./c).^2).*L1Muxx -
(((w./c).*sin(theta)).^2 ./ L1Epzz)).*L1Epyy )).^2)  ./ (exp(2*2*Pi*i*d1*(sqrt(
(((w./c).^2).*L1Muxx - (((w./c).*sin(theta)).^2 ./ L1Epzz)).*L1Epyy
))).*(((w./c).*cos(theta)) - (sqrt( (((w./c).^2).*L1Muxx - (((w./c).*sin(theta)).^2 ./
L1Epzz)).*L1Epyy ))).^2 - (((w./c).*cos(theta)) + (sqrt( (((w./c).^2).*L1Muxx -
(((w./c).*sin(theta)).^2 ./ L1Epzz)).*L1Epyy ))).^2);

    case {'2s'}
Tr =
@(w,L1Epxx,L1Epyy,L1Epzz,L1Muxx,L1Muyy,L1Muzz,L2Epxx,L2Epyy,L2Epzz,L2Muxx,L2Muyy,L2Muzz,L
3Epxx,L3Epyy,L3Epzz,L3Muxx,L3Muyy,L3Muzz,theta,d1,d2,d3) ...
- (4* exp( -2*i*(d1+d2).*((w./c).*cos(theta)).*Pi) .* ((w./c).*cos(theta)) .* (sqrt(
(((w./c).^2).*L1Epxx - (((w./c).*sin(theta)).^2 ./ L1Muzz)).*L1Muyy )) .* (sqrt(
(((w./c).^2).*L2Epxx - (((w./c).*sin(theta)).^2 ./ L2Muzz)).*L2Muyy ))) ./
(((w./c).*cos(theta)) .* ((sqrt( (((w./c).^2).*L1Epxx - (((w./c).*sin(theta)).^2 ./
L1Muzz)).*L1Muyy )) - (sqrt( (((w./c).^2).*L2Epxx - (((w./c).*sin(theta)).^2 ./
L2Muzz)).*L2Muyy ))).^2 .* cos(2*Pi.*(d1.*(sqrt( (((w./c).^2).*L1Epxx -
(((w./c).*sin(theta)).^2 ./ L1Muzz)).*L1Muyy )) - d2.*(sqrt( (((w./c).^2).*L2Epxx -
(((w./c).*sin(theta)).^2 ./ L2Muzz)).*L2Muyy )))) - ((w./c).*cos(theta)) .* ((sqrt(
(((w./c).^2).*L1Epxx - (((w./c).*sin(theta)).^2 ./ L1Muzz)).*L1Muyy )) + (sqrt(
(((w./c).^2).*L2Epxx - (((w./c).*sin(theta)).^2 ./ L2Muzz)).*L2Muyy ))).^2 .*
cos(2*Pi.*(d1.*(sqrt( (((w./c).^2).*L1Epxx - (((w./c).*sin(theta)).^2 ./ L1Muzz)).*L1Muyy
)) + d2.*(sqrt( (((w./c).^2).*L2Epxx - (((w./c).*sin(theta)).^2 ./ L2Muzz)).*L2Muyy ))))
...
+ i .* ( ((sqrt( (((w./c).^2).*L1Epxx - (((w./c).*sin(theta)).^2 ./ L1Muzz)).*L1Muyy )) -
(sqrt( (((w./c).^2).*L2Epxx - (((w./c).*sin(theta)).^2 ./ L2Muzz)).*L2Muyy ))) .* (-
((w./c).*cos(theta)).^2 + (sqrt( (((w./c).^2).*L1Epxx - (((w./c).*sin(theta)).^2 ./
L1Muzz)).*L1Muyy )).*(sqrt( (((w./c).^2).*L2Epxx - (((w./c).*sin(theta)).^2 ./
L2Muzz)).*L2Muyy ))) .* sin( 2*Pi .* (d1.*(sqrt( (((w./c).^2).*L1Epxx -
(((w./c).*sin(theta)).^2 ./ L1Muzz)).*L1Muyy )) - d2.*(sqrt( (((w./c).^2).*L2Epxx -
(((w./c).*sin(theta)).^2 ./ L2Muzz)).*L2Muyy )))) + ((sqrt( (((w./c).^2).*L1Epxx -
(((w./c).*sin(theta)).^2 ./ L1Muzz)).*L1Muyy )) + (sqrt( (((w./c).^2).*L2Epxx -
(((w./c).*sin(theta)).^2 ./ L2Muzz)).*L2Muyy ))) .* (((w./c).*cos(theta)).^2 + (sqrt(
(((w./c).^2).*L1Epxx - (((w./c).*sin(theta)).^2 ./ L1Muzz)).*L1Muyy )).*(sqrt(
(((w./c).^2).*L2Epxx - (((w./c).*sin(theta)).^2 ./ L2Muzz)).*L2Muyy ))) .* sin(2*Pi .*
(d1.*(sqrt( (((w./c).^2).*L1Epxx - (((w./c).*sin(theta)).^2 ./ L1Muzz)).*L1Muyy )) +
d2.*(sqrt( (((w./c).^2).*L2Epxx - (((w./c).*sin(theta)).^2 ./ L2Muzz)).*L2Muyy )))) ));

Re =
@(w,L1Epxx,L1Epyy,L1Epzz,L1Muxx,L1Muyy,L1Muzz,L2Epxx,L2Epyy,L2Epzz,L2Muxx,L2Muyy,L2Muzz,L
3Epxx,L3Epyy,L3Epzz,L3Muxx,L3Muyy,L3Muzz,theta,d1,d2,d3) ...
    1;

    case {'2p'}
Tr =
@(w,L1Epxx,L1Epyy,L1Epzz,L1Muxx,L1Muyy,L1Muzz,L2Epxx,L2Epyy,L2Epzz,L2Muxx,L2Muyy,L2Muzz,L
3Epxx,L3Epyy,L3Epzz,L3Muxx,L3Muyy,L3Muzz,theta,d1,d2,d3) ...
1;
Re =
@(w,L1Epxx,L1Epyy,L1Epzz,L1Muxx,L1Muyy,L1Muzz,L2Epxx,L2Epyy,L2Epzz,L2Muxx,L2Muyy,L2Muzz,L
3Epxx,L3Epyy,L3Epzz,L3Muxx,L3Muyy,L3Muzz,theta,d1,d2,d3) ...
    1;

    case {'3s'}
Tr =
@(w,L1Epxx,L1Epyy,L1Epzz,L1Muxx,L1Muyy,L1Muzz,L2Epxx,L2Epyy,L2Epzz,L2Muxx,L2Muyy,L2Muzz,L
3Epxx,L3Epyy,L3Epzz,L3Muxx,L3Muyy,L3Muzz,theta,d1,d2,d3) ...
(16 .* exp(2 .* i .* Pi .* (d3 .* (-((w./c).*cos(theta))+(sqrt( (((w./c).^2).*L3Epxx -
(((w./c).*sin(theta)).^2 ./ L3Muzz)).*L3Muyy )))+d2 .* (-((w./c).*cos(theta))+(sqrt(
(((w./c).^2).*L2Epxx - (((w./c).*sin(theta)).^2 ./ L2Muzz)).*L2Muyy ))+2 .* (sqrt(
(((w./c).^2).*L3Epxx - (((w./c).*sin(theta)).^2 ./ L3Muzz)).*L3Muyy )))+d1 .* (-
((w./c).*cos(theta))+(sqrt( (((w./c).^2).*L1Epxx - (((w./c).*sin(theta)).^2 ./
L1Muzz)).*L1Muyy ))+2 .*((sqrt( (((w./c).^2).*L2Epxx - (((w./c).*sin(theta)).^2 ./
L2Muzz)).*L2Muyy ))+(sqrt( (((w./c).^2).*L3Epxx - (((w./c).*sin(theta)).^2 ./
L3Muzz)).*L3Muyy )))))) .* ((w./c).*cos(theta)) .* (sqrt( (((w./c).^2).*L1Epxx -
(((w./c).*sin(theta)).^2 ./ L1Muzz)).*L1Muyy )) .* (sqrt( (((w./c).^2).*L2Epxx -
(((w./c).*sin(theta)).^2 ./ L2Muzz)).*L2Muyy )) .* (sqrt( (((w./c).^2).*L3Epxx -
(((w./c).*sin(theta)).^2 ./ L3Muzz)).*L3Muyy ))) ./ ...
    ((((w./c).*cos(theta))+(sqrt( (((w./c).^2).*L1Epxx - (((w./c).*sin(theta)).^2 ./
L1Muzz)).*L1Muyy ))+ exp(4 .* i .* d1 .* Pi .* (sqrt( (((w./c).^2).*L1Epxx -
(((w./c).*sin(theta)).^2 ./ L1Muzz)).*L1Muyy ))) .*(-((w./c).*cos(theta))+(sqrt(
(((w./c).^2).*L1Epxx - (((w./c).*sin(theta)).^2 ./ L1Muzz)).*L1Muyy )))) .* (sqrt(
(((w./c).^2).*L2Epxx - (((w./c).*sin(theta)).^2 ./ L2Muzz)).*L2Muyy )) .* ...
    (exp(4 .* i .* Pi .* (d1 .* (sqrt( (((w./c).^2).*L2Epxx - (((w./c).*sin(theta)).^2 ./
L2Muzz)).*L2Muyy ))+(d1+d2+d3) .* (sqrt( (((w./c).^2).*L3Epxx - (((w./c).*sin(theta)).^2
./ L3Muzz)).*L3Muyy )))) .* ((sqrt( (((w./c).^2).*L2Epxx - (((w./c).*sin(theta)).^2 ./
L2Muzz)).*L2Muyy ))-(sqrt( (((w./c).^2).*L3Epxx - (((w./c).*sin(theta)).^2 ./
L3Muzz)).*L3Muyy ))) .* (-((w./c).*cos(theta))+(sqrt( (((w./c).^2).*L3Epxx -
(((w./c).*sin(theta)).^2 ./ L3Muzz)).*L3Muyy )))- exp(4 .* i .* Pi.* (d1+d2) .*  ((sqrt(
(((w./c).^2).*L2Epxx - (((w./c).*sin(theta)).^2 ./ L2Muzz)).*L2Muyy ))+(sqrt(
(((w./c).^2).*L3Epxx - (((w./c).*sin(theta)).^2 ./ L3Muzz)).*L3Muyy )))) .* ...
    ((sqrt( (((w./c).^2).*L2Epxx - (((w./c).*sin(theta)).^2 ./ L2Muzz)).*L2Muyy ))-(sqrt(
(((w./c).^2).*L3Epxx - (((w./c).*sin(theta)).^2 ./ L3Muzz)).*L3Muyy ))) .*
(((w./c).*cos(theta))+(sqrt( (((w./c).^2).*L3Epxx - (((w./c).*sin(theta)).^2 ./
L3Muzz)).*L3Muyy )))- exp(4 .* i .* Pi .* ((d1+d2) .* (sqrt( (((w./c).^2).*L2Epxx -
(((w./c).*sin(theta)).^2 ./ L2Muzz)).*L2Muyy ))+(d1+d2+d3) .* (sqrt( (((w./c).^2).*L3Epxx
- (((w./c).*sin(theta)).^2 ./ L3Muzz)).*L3Muyy )))) .* (-((w./c).*cos(theta))+(sqrt(
(((w./c).^2).*L3Epxx - (((w./c).*sin(theta)).^2 ./ L3Muzz)).*L3Muyy ))) .* ((sqrt(
(((w./c).^2).*L2Epxx - (((w./c).*sin(theta)).^2 ./ L2Muzz)).*L2Muyy ))+(sqrt(
(((w./c).^2).*L3Epxx - (((w./c).*sin(theta)).^2 ./ L3Muzz)).*L3Muyy )))+ ...
    exp(4 .* i .* Pi .* (d2 .* (sqrt( (((w./c).^2).*L3Epxx - (((w./c).*sin(theta)).^2 ./
L3Muzz)).*L3Muyy ))+d1 .* ((sqrt( (((w./c).^2).*L2Epxx - (((w./c).*sin(theta)).^2 ./
L2Muzz)).*L2Muyy ))+(sqrt( (((w./c).^2).*L3Epxx - (((w./c).*sin(theta)).^2 ./
L3Muzz)).*L3Muyy ))))) .* (((w./c).*cos(theta))+(sqrt( (((w./c).^2).*L3Epxx -
(((w./c).*sin(theta)).^2 ./ L3Muzz)).*L3Muyy ))) .* ((sqrt( (((w./c).^2).*L2Epxx -
(((w./c).*sin(theta)).^2 ./ L2Muzz)).*L2Muyy ))+(sqrt( (((w./c).^2).*L3Epxx -
(((w./c).*sin(theta)).^2 ./ L3Muzz)).*L3Muyy ))) ) + (sqrt( (((w./c).^2).*L1Epxx -
(((w./c).*sin(theta)).^2 ./ L1Muzz)).*L1Muyy )) .* (((w./c).*cos(theta))+exp(4 .* i .* Pi
.* d1 .*  (sqrt( (((w./c).^2).*L1Epxx - (((w./c).*sin(theta)).^2 ./ L1Muzz)).*L1Muyy )))
.* (((w./c).*cos(theta))-(sqrt( (((w./c).^2).*L1Epxx - (((w./c).*sin(theta)).^2 ./
L1Muzz)).*L1Muyy ))) + (sqrt( (((w./c).^2).*L1Epxx - (((w./c).*sin(theta)).^2 ./
L1Muzz)).*L1Muyy ))) .* ...
    (exp(4 .* i .* Pi .* (d1 .* (sqrt( (((w./c).^2).*L2Epxx - (((w./c).*sin(theta)).^2 ./
L2Muzz)).*L2Muyy ))+(d1+d2+d3) .* (sqrt( (((w./c).^2).*L3Epxx - (((w./c).*sin(theta)).^2
./ L3Muzz)).*L3Muyy )))) .* ((sqrt( (((w./c).^2).*L2Epxx - (((w./c).*sin(theta)).^2 ./
L2Muzz)).*L2Muyy ))-(sqrt( (((w./c).^2).*L3Epxx - (((w./c).*sin(theta)).^2 ./
L3Muzz)).*L3Muyy ))) .* (-((w./c).*cos(theta))+(sqrt( (((w./c).^2).*L3Epxx -
(((w./c).*sin(theta)).^2 ./ L3Muzz)).*L3Muyy ))) + exp(4 .* i .* Pi .* (d1+d2).* ((sqrt(
(((w./c).^2).*L2Epxx - (((w./c).*sin(theta)).^2 ./ L2Muzz)).*L2Muyy ))+(sqrt(
(((w./c).^2).*L3Epxx - (((w./c).*sin(theta)).^2 ./ L3Muzz)).*L3Muyy )))) .* ...
    ((sqrt( (((w./c).^2).*L2Epxx - (((w./c).*sin(theta)).^2 ./ L2Muzz)).*L2Muyy ))-(sqrt(
(((w./c).^2).*L3Epxx - (((w./c).*sin(theta)).^2 ./ L3Muzz)).*L3Muyy ))) .*
(((w./c).*cos(theta))+(sqrt( (((w./c).^2).*L3Epxx - (((w./c).*sin(theta)).^2 ./
L3Muzz)).*L3Muyy ))) + exp(4 .* i .* Pi .* ((d1+d2) .* (sqrt( (((w./c).^2).*L2Epxx -
(((w./c).*sin(theta)).^2 ./ L2Muzz)).*L2Muyy ))+(d1+d2+d3) .* (sqrt( (((w./c).^2).*L3Epxx
- (((w./c).*sin(theta)).^2 ./ L3Muzz)).*L3Muyy )))) .* (-((w./c).*cos(theta))+(sqrt(
(((w./c).^2).*L3Epxx - (((w./c).*sin(theta)).^2 ./ L3Muzz)).*L3Muyy ))) .* ((sqrt(
(((w./c).^2).*L2Epxx - (((w./c).*sin(theta)).^2 ./ L2Muzz)).*L2Muyy ))+(sqrt(
(((w./c).^2).*L3Epxx - (((w./c).*sin(theta)).^2 ./ L3Muzz)).*L3Muyy )))+ ...
    exp(4 .* i .* Pi .* (d2 .* (sqrt( (((w./c).^2).*L3Epxx - (((w./c).*sin(theta)).^2 ./
L3Muzz)).*L3Muyy ))+d1 .* ((sqrt( (((w./c).^2).*L2Epxx - (((w./c).*sin(theta)).^2 ./
L2Muzz)).*L2Muyy ))+(sqrt( (((w./c).^2).*L3Epxx - (((w./c).*sin(theta)).^2 ./
L3Muzz)).*L3Muyy ))))) .* (((w./c).*cos(theta))+(sqrt( (((w./c).^2).*L3Epxx -
(((w./c).*sin(theta)).^2 ./ L3Muzz)).*L3Muyy ))) .* ((sqrt( (((w./c).^2).*L2Epxx -
(((w./c).*sin(theta)).^2 ./ L2Muzz)).*L2Muyy ))+(sqrt( (((w./c).^2).*L3Epxx -
(((w./c).*sin(theta)).^2 ./ L3Muzz)).*L3Muyy ))))) ;


Re =
@(w,L1Epxx,L1Epyy,L1Epzz,L1Muxx,L1Muyy,L1Muzz,L2Epxx,L2Epyy,L2Epzz,L2Muxx,L2Muyy,L2Muzz,L
3Epxx,L3Epyy,L3Epzz,L3Muxx,L3Muyy,L3Muzz,theta,d1,d2,d3) ...
    1;

    case {'3p'}
```

```
Tr = 
@(w,L1Epxx,L1Epyy,L1Epzz,L1Muxx,L1Muyy,L1Muzz,L2Epxx,L2Epyy,L2Epzz,L2Muxx,L2Muyy,L2Muzz,L
3Epxx,L3Epyy,L3Epzz,L3Muxx,L3Muyy,L3Muzz,theta,d1,d2,d3) ...
    1;
Re = 
@(w,L1Epxx,L1Epyy,L1Epzz,L1Muxx,L1Muyy,L1Muzz,L2Epxx,L2Epyy,L2Epzz,L2Muxx,L2Muyy,L2Muzz,L
3Epxx,L3Epyy,L3Epzz,L3Muxx,L3Muyy,L3Muzz,theta,d1,d2,d3) ...
    1;

end

Out = {Tr,Re};
```

function varargout = InteractiveEpMu(varargin);

```
global handles;
% vargsout = [AepSTART,GepSTART,woepSTART,wpepSTART]
% INTERACTIVEEPMU M-file for InteractiveEpMu.fig


% Begin initialization code - DO NOT EDIT
    gui_Singleton = 1;
    gui_State = struct('gui_Name',       mfilename, ...
                       'gui_Singleton',  gui_Singleton, ...
                       'gui_OpeningFcn', @InteractiveEpMu_OpeningFcn, ...
                       'gui_OutputFcn',  @InteractiveEpMu_OutputFcn, ...
                       'gui_LayoutFcn',  [] , ...
                       'gui_Callback',   []);


    if nargin && ischar(varargin{1})  gui_State.gui_Callback = str2func(varargin{1});
end

    if nargout
        varargout{1} = gui_mainfcn(gui_State, varargin{:});
%        uiwait(handles.figure1);
    else
        gui_mainfcn(gui_State,varargin{:});
    end

% --- Executes just before InteractiveEpMu is made visible.
function InteractiveEpMu_OpeningFcn(hObject, eventdata, handles, varargin)
%   Trans00degDEFAULT = load('.DAT');
    wAHzDEFAULT = load('Freq.DAT');
    handles.wExp = wAHzDEFAULT;
%   handles.Trans00deg = Trans00degDEFAULT;

    %to find and assign data variables
    if(nargin > 3)
     for index = 1:1:(nargin-3),
            if nargin-3==index, break, end
            if ischar(varargin{index})
                switch (lower(varargin{index}))
                 case 'wahz'
                  handles.wExp = varargin{index+1};
                 case 'exparray'
                  handles.ExpDataArray = varargin{index+1};
                end
            end
            end
    end

    handles = Initialize_Material_Tensors(hObject, eventdata, handles);  % this manages
most all initial assignments.

    % This sets up the initial plot - only do when we are invisible
    % so window can get raised using InteractiveEpMu.
    if strcmp(get(hObject,'Visible'),'off')
        axis(handles.TransPlot);
        axis([min(handles.wExp)/2,max(handles.wExp)*2,0,1]);
    end

    handles.Plot.DataShow(1:(length(handles.ExpDataArray)))=1;         %initially select
'on' for plot of all Transmission Data
    %display the starter schematic picture:

    guidata(hObject,handles);

    Update_Setup(hObject, eventdata, handles);

    % UIWAIT % makes InteractiveEpMu wait for user response (see UIRESUME)
    uiwait(handles.figure1);


function varargout = InteractiveEpMu_OutputFcn(hObject, eventdata, handles)

    % Get default command line output from handles structure
%    handles.output.Lorentz = handles.EpxLorentz;
%    handles.output.epx = handles.epx;
    varargout{1} = handles;
    delete(handles.figure1);


% --------------------------------------------------------------------------------
% --------EXECUTES ON UPDATE BUTTON PRESS, ALSO CALLED AFTER ANY ACTION -------
% --------------------------------------------------------------------------------
function Update_Callback(hObject, eventdata, handles)
    % hObject    handle to Update (see GCBO)
    % eventdata  reserved - to be defined in omega0Mu future version of MATLAB
    % handles    structure with handles and user data (see GUIDATA)

    %Setup parameters for Transmission calculation.  These just reduce the length of the
trans equation
    % so we don't have to reference the handles structure at each variable.
    i=sqrt(-1);

    c=3e10;
    d1=handles.L1.d;
    d2=handles.L2.d;
    d3=handles.L3.d;
    handles.L1.Periodicity = d3;
    d1Var=handles.L1.dVar * 0.01;
    d2Var=handles.L2.dVar * 0.01;
    d3Var=handles.L3.dVar * 0.01;

    wAHzSTEP = (max(handles.wExp)-min(handles.wExp))/length(handles.wExp);  % this ensure
matching steps to data
    wCalcMin = handles.wCalcMin - mod(handles.wCalcMin - handles.wExp(1),wAHzSTEP);
    handles.wCalc = [wCalcMin : wAHzSTEP/2 : handles.wCalcMax ];
                % this For Loop would be better written as a find(handles.wCalc =
handles.wExp(1)) or something similar.

    wCalcExpMatch = find(handles.wCalc >= handles.wExp(1),1);

%   TensorSetup(handles)                    %% LAYER 1\
%   Use eval commands to only change the active oscillator in active layer.
%    eval(['OSCEpar(1:19) =', handles.EActive(1:19)])
%    eval(['OSCMpar(1:19) =', handles.MActive(1:19)])
    OSCEii = str2num(handles.EActive(21));    OSCEjj=str2num(handles.EActive(23));
OSCMii=str2num(handles.MActive(21));    OSCMjj=str2num(handles.MActive(23));

%    This contains the layer and the {ii,jj} tensor component.  Need to make sure all
layers and {ii,jj} are already setup.
    EpOscPar = eval(handles.EActive);
    MuOscPar= eval(handles.MActive);
    Disorder = eval([handles.EActive(1:11),'Disorder']);
    Periodicity = eval([handles.EActive(1:11),'Periodicity']);

    out =
OscLookup(handles.wCalc,EpOscPar,MuOscPar,Periodicity,handles.Dispersion,Disorder);
%   handles.L1.Epsilon{OSCEii,OSCEjj} = out{1};    handles.L1.Mue{OSCMii,OSCMjj}     =
out{2};
    eval([handles.EActive(1:11),'Epsilon{OSCEii,OSCEjj} = out{1};']);
    eval([handles.MActive(1:11),'Mue{OSCMii,OSCMjj} = out{2};']);


         if(get(handles.EL1isotropic,'Value'))
             handles.L1.Epsilon{2,2} = handles.L1.Epsilon{1,1};
handles.L1.Epsilon{3,3} = handles.L1.Epsilon{1,1};
```

```
             handles.L1.EpOscPar{2,2} = handles.L1.EpOscPar{1,1};
handles.L1.EpOscPar{3,3} = handles.L1.EpOscPar{1,1};
         end
         if(get(handles.ML1isotropic,'Value'))
             handles.L1.Mue{2,2} = handles.L1.Mue{1,1};
handles.L1.Mue{3,3} = handles.L1.Mue{1,1};
             handles.L1.MuOscPar{2,2} = handles.L1.MuOscPar{1,1};
handles.L1.MuOscPar{3,3} = handles.L1.MuOscPar{1,1};
         end

        if(get(handles.EL2isotropic,'Value'))
            handles.L2.Epsilon{2,2} = handles.L2.Epsilon{1,1};
handles.L2.Epsilon{3,3} = handles.L2.Epsilon{1,1};
            handles.L2.EpOscPar{2,2} = handles.L2.EpOscPar{1,1};
handles.L2.EpOscPar{3,3} = handles.L2.EpOscPar{1,1};
        end
        if(get(handles.ML2isotropic,'Value'))
            handles.L2.Mue{2,2} = handles.L2.Mue{1,1};
handles.L2.Mue{3,3} = handles.L2.Mue{1,1};
            handles.L2.MuOscPar{2,2} = handles.L2.MuOscPar{1,1};
handles.L2.MuOscPar{3,3} = handles.L2.MuOscPar{1,1};
        end

        if(get(handles.EL3isotropic,'Value'))
            handles.L3.Epsilon{2,2} = handles.L3.Epsilon{1,1};
handles.L3.Epsilon{3,3} = handles.L3.Epsilon{1,1};
            handles.L3.EpOscPar{2,2} = handles.L3.EpOscPar{1,1};
handles.L3.EpOscPar{3,3} = handles.L3.EpOscPar{1,1};
        end
        if(get(handles.ML3isotropic,'Value'))
            handles.L3.Mue{2,2} = handles.L3.Mue{1,1};
handles.L3.Mue{3,3} = handles.L3.Mue{1,1};
            handles.L3.MuOscPar{2,2} = handles.L3.MuOscPar{1,1};
handles.L3.MuOscPar{3,3} = handles.L3.MuOscPar{1,1};
        end

% Transmission Equation Selector.  This function contains a database of anonymous
function forms for diferent layers, polarization, Refl/Trans, and other factors.
    Out = TELookup(handles.Setup.Layers, handles.Setup.Pol);   % TELookup outputs two
anon funcs, a trans and refl equation.
    Tr = Out{1};   %Tr = @(w,Epxx,Epyy,Epzz,Muxx,Muyy,Muzz,theta,d)
    Re = Out{2};

            % ad-hoc modifying all these parts for single-angle calculation but multi-
data-set DISPLAY ONLY.
    for jj=1:1           % cut   (size(handles.ExpDataArray,2))
        handles.DataCalc{1,jj}=handles.ExpDataArray{1,jj};    %writes angle info to
DataCalc
        theta=str2double(handles.ExpDataArray{1,jj})*pi/180;  %top row in ExpDataArray is
theta lables
        if(handles.ExpDataArray{3,jj}) == 'T'  %bottom row is 'T' or 'R' lables
            Transmit = 0;
            Tran =
(Tr(handles.wCalc,handles.L1.Epsilon{1,1},handles.L1.Epsilon{2,2},handles.L1.Epsilon{3,3}
,handles.L1.Mue{1,1},handles.L1.Mue{2,2},handles.L1.Mue{3,3},handles.L2.Epsilon{1,1},hand
les.L2.Epsilon{2,2},handles.L2.Epsilon{3,3},handles.L2.Mue{1,1},handles.L2.Mue{2,2},handl
es.L2.Mue{3,3},handles.L3.Epsilon{1,1},handles.L3.Epsilon{2,2},handles.L3.Epsilon{3,3},ha
ndles.L3.Mue{1,1},handles.L3.Mue{2,2},handles.L3.Mue{3,3},theta,d1,d2,d3));
            Trans = Tran .* conj(Tran);
            Transmit = Transmit + Trans;

% Resolution Limit filter.
                handles.DataCalc{2,jj} = Transmit;  %writes average of thickness varience
data to 2nd row
                handles.DataCalc{3,jj} = 'T';       %wrties T/R to 3rd row
                [b,a] = butter(5,handles.ResLimit,'low');
                TransmitFlip(1:length(handles.wCalc)) = Transmit(end:-1:1);
                TransmitBuffered(1:length(handles.wCalc)) = TransmitFlip(:);
TransmitBuffered(end+1:end+length(handles.wCalc))=Transmit(:);
```

```
                TransmitFiltered = filtfilt(b,a,TransmitBuffered);
                handles.DataCalc{4,jj} = 0;
                handles.DataCalc{4,jj} =
TransmitFiltered(length(handles.wCalc)+1:2*length(handles.wCalc));
                        % cell col 4 holds filtered data.

%               FitDiff{jj} = abs(Trans(wCalcExpMatch : wCalcExpMatch +
length(handles.wExp)-1) - transpose(handles.ExpDataArray{2,jj}));  %array diff between
data and theory in overlap region only.  used for fit quality below in graph update
routines.
        elseif(handles.ExpDataArray{3,jj}) == 'R'
            Refl =
(Re(handles.wCalc,handles.L1.Epsilon{1,1},handles.L1.Epsilon{2,2},handles.L1.Epsilon{3,3}
,handles.L1.Mue{1,1},handles.L1.Mue{2,2},handles.L1.Mue{3,3},handles.L2.Epsilon{1,1},hand
les.L2.Epsilon{2,2},handles.L2.Epsilon{3,3},handles.L2.Mue{1,1},handles.L2.Mue{2,2},handl
es.L2.Mue{3,3},handles.L3.Epsilon{1,1},handles.L3.Epsilon{2,2},handles.L3.Epsilon{3,3},ha
ndles.L3.Mue{1,1},handles.L3.Mue{2,2},handles.L3.Mue{3,3},theta,d1,d2,d3));
            Reflect = Refl .* conj(Refl);
            handles.DataCalc{2,jj} = Reflect;
            handles.DataCalc{3,jj} = 'R';
%            FitDiff{jj} = abs(Reflect(wCalcExpMatch : wCalcExpMatch +
length(handles.wExp)-1) - transpose(handles.ExpDataArray{2,jj}));
        end
    end

    %update graphs
    axes(handles.EpxPlot); cla;     %Ep Plot Update
    switch(get(handles.EpList,'value'))     % this can be optimized using eval
        case 1
            plot(handles.wCalc,real(handles.L1.Epsilon{1,1}),'Color','black'); hold on;
            plot(handles.wCalc,imag(handles.L1.Epsilon{1,1}),'Color','Red');
        case 2
            plot(handles.wCalc,real(handles.L1.Epsilon{2,2}),'Color','black'); hold on;
            plot(handles.wCalc,imag(handles.L1.Epsilon{2,2}),'Color','Red');
        case 3
            plot(handles.wCalc,real(handles.L1.Epsilon{3,3}),'Color','black'); hold on;
            plot(handles.wCalc,imag(handles.L1.Epsilon{3,3}),'Color','Red');
        case 4
            plot(handles.wCalc,real(handles.L2.Epsilon{1,1}),'Color','black'); hold on;
            plot(handles.wCalc,imag(handles.L2.Epsilon{1,1}),'Color','Red');
        case 5
            plot(handles.wCalc,real(handles.L2.Epsilon{2,2}),'Color','black'); hold on;
            plot(handles.wCalc,imag(handles.L2.Epsilon{2,2}),'Color','Red');
        case 6
            plot(handles.wCalc,real(handles.L2.Epsilon{3,3}),'Color','black'); hold on;
            plot(handles.wCalc,imag(handles.L2.Epsilon{3,3}),'Color','Red');
        case 7
            plot(handles.wCalc,real(handles.L3.Epsilon{1,1}),'Color','black'); hold on;
            plot(handles.wCalc,imag(handles.L3.Epsilon{1,1}),'Color','Red');
        case 8
            plot(handles.wCalc,real(handles.L3.Epsilon{2,2}),'Color','black'); hold on;
            plot(handles.wCalc,imag(handles.L3.Epsilon{2,2}),'Color','Red');
        case 9
            plot(handles.wCalc,real(handles.L3.Epsilon{3,3}),'Color','black'); hold on;
            plot(handles.wCalc,imag(handles.L3.Epsilon{3,3}),'Color','Red');
    end
    axis([handles.wCalcMin, handles.wCalcMax, handles.Plot.EpxMin ,
handles.Plot.EpxMax]);

    axes(handles.MuzPlot);   cla;     %Mu Plot Update
    switch(get(handles.MuList,'value'))      % this can be optimized using eval
        case 1
            plot(handles.wCalc,real(handles.L1.Mue{1,1}),'Color','black'); hold on;
            plot(handles.wCalc,imag(handles.L1.Mue{1,1}),'Color','Red');
        case 2
            plot(handles.wCalc,real(handles.L1.Mue{2,2}),'Color','black'); hold on;
            plot(handles.wCalc,imag(handles.L1.Mue{2,2}),'Color','Red');
        case 3
            plot(handles.wCalc,real(handles.L1.Mue{3,3}),'Color','black'); hold on;
            plot(handles.wCalc,imag(handles.L1.Mue{3,3}),'Color','Red');
        case 4
            plot(handles.wCalc,real(handles.L2.Mue{1,1}),'Color','black'); hold on;
            plot(handles.wCalc,imag(handles.L2.Mue{1,1}),'Color','Red');
        case 5
            plot(handles.wCalc,real(handles.L2.Mue{2,2}),'Color','black'); hold on;
            plot(handles.wCalc,imag(handles.L2.Mue{2,2}),'Color','Red');
        case 6
            plot(handles.wCalc,real(handles.L2.Mue{3,3}),'Color','black'); hold on;
            plot(handles.wCalc,imag(handles.L2.Mue{3,3}),'Color','Red');
        case 7
            plot(handles.wCalc,real(handles.L3.Mue{1,1}),'Color','black'); hold on;
            plot(handles.wCalc,imag(handles.L3.Mue{1,1}),'Color','Red');
        case 8
            plot(handles.wCalc,real(handles.L3.Mue{2,2}),'Color','black'); hold on;
            plot(handles.wCalc,imag(handles.L3.Mue{2,2}),'Color','Red');
        case 9
            plot(handles.wCalc,real(handles.L3.Mue{3,3}),'Color','black'); hold on;
            plot(handles.wCalc,imag(handles.L3.Mue{3,3}),'Color','Red');
    end
    axis([handles.wCalcMin, handles.wCalcMax, handles.Plot.MuzMin, handles.Plot.MuzMax]);

    TFitRes = 0; RFitRes = 0; TFitii=0; RFitii=0;
    axes(handles.TransPlot); cla;   %Calculated Reflect/Trans Plot update   % ad-hoc
modified for single angle multi- dataset.

%    plot(handles.wCalc,handles.DataCalc{2,1}-
handles.Plot.Toffset,'Color','Red','LineWidth',.1); hold on;
    plot(handles.wCalc,handles.DataCalc{4,1}(1:length(handles.wCalc))-
handles.Plot.Toffset,'Color',handles.Plot.ColorArray{1},'LineWidth',2); hold on;
    for jj=1:(size(handles.ExpDataArray,2))
        if handles.Plot.DataShow(jj)
            if handles.ExpDataArray{3,jj} == 'T'

plot(handles.wExp,handles.ExpDataArray{2,jj},'Color',handles.Plot.ColorArray{jj});hold
on;
%                 if(handles.Plot.DataShowCalc(jj))
%                     plot(handles.wCalc,handles.DataCalc{2,jj}-
handles.Plot.Toffset,'Color','Red','LineWidth',.1); hold on;
%                     plot(handles.wCalc,handles.DataCalc{4,jj}(1:length(handles.wCalc))-
handles.Plot.Toffset,'Color',handles.Plot.ColorArray{jj},'LineWidth',2); hold on;
%                     %                   TFitii = TFitii + 1;
% %                    TFitRes = TFitRes + sum(FitDiff{jj})/length(handles.wExp);
%FitDiff matches ExpDataArray and DataCalc so the T or R order is the same in it.
% %                    if(TFitii) set(handles.TFitRes,'String',num2str( TFitRes /
TFitii));  end; %The quality of fit here is the average difference between data and
theory, extrinsic of #datapoints and #angles considered.
%                end
            elseif handles.ExpDataArray{3,jj} == 'R'

plot(handles.wExp,handles.ExpDataArray{2,jj},'Color',handles.Plot.ColorArray{jj});hold
on;
                if(handles.Plot.DataShowCalc(jj))

plot(handles.wCalc,handles.DataCalc{2,jj}+handles.Plot.Toffset,'Color',handles.Plot.Color
Array{jj},'LineWidth',2); hold on;
%                   RFitii = RFitii + 1;
%                   RFitRes = RFitRes + sum(FitDiff{jj})/length(handles.wExp);
%                   if(RFitii) set(handles.RFitRes,'String',num2str( RFitRes / RFitii));
end;
                end
            end
        end
    end
%    if(TFitii | RFitii) set(handles.NWFitRes,'String',num2str( (TFitRes+RFitRes) /
(TFitii+RFitii) ) );  end; % Sum Fit is average of R and T weighted by how many scans are
in each.
```

```
%    if(TFitii & RFitii) set(handles.NUFitRes,'String',num2str( (TFitRes/TFitii +
RFitRes/RFitii) / 2 )); end; % Sum Fit is average of R and T weighted by how many scans
are in each.
    axis([handles.wCalcMin , handles.wCalcMax, handles.Plot.TransMin,
handles.Plot.TransMax]);

    guidata(hObject,handles);

% --------------- FINISH BUTTON ----------------------------------------
function Finish_Callback(hObject, eventdata, handles)
    % hObject    handle to Finish (see GCBO)
    selection = questdlg(['Close ' get(handles.figure1,'Name') '?'],...
                         ['Close ' get(handles.figure1,'Name') '...'],...
                         'Yes','No','Yes');
    if strcmp(selection,'No') return; end
    handles.NotDone = 1;
    guidata(hObject,handles);
    uiresume(handles.figure1);

% --------------- UPDATE SETUP ---------------------------------------------
% updates the Setup parameters such as number of layers, and polarization:
function Update_Setup(hObject, eventdata, handles)
    axes(handles.Diagram);
    SetupFile = strcat(handles.Setup.Pol,int2str(handles.Setup.Layers),'lay.tif');
    I=imread(SetupFile, 'tif');
    imagesc(I);
    set(gcf, 'Units', 'pixels');
    set(gca, 'Visible','off');
    handles.L1.d = str2double(get(handles.L1d, 'String'));
    handles.L2.d = str2double(get(handles.L2d, 'String'));
    handles.L3.d = str2double(get(handles.L3d, 'String'));

% -----------------------------------------------------------------
% ---------------------Electric Parameter--------------------------  [A, G, wo, wp, inf,
'function']

function OscAmpEp_Callback(hObject, eventdata, handles)
    input = str2double(get(hObject, 'String'));
    if isnan(input) set(hObject, 'String', 0); errordlg('Input must be a
number','Error');
    else eval([handles.EActive,'{1} = input']); end;
    guidata(hObject,handles);
    Update_Callback(hObject, eventdata, handles)
function GEp_Callback(hObject, eventdata, handles)
    input = str2double(get(hObject,'String'));
    if isnan(input) set(hObject, 'String', 0); errordlg('Input must be a
number','Error');
    else eval([handles.EActive,'{2} = input']); end;
    guidata(hObject,handles)
    Update_Callback(hObject, eventdata, handles)
function omega0Ep_Callback(hObject, eventdata, handles)
    input = str2double(get(hObject,'String'));
    if isnan(input) set(hObject, 'String', 0); errordlg('Input must be a
number','Error');
    else eval([handles.EActive,'{3} = input']); end;
    guidata(hObject,handles)
    Update_Callback(hObject, eventdata, handles)
function omegapEp_Callback(hObject, eventdata, handles)
    input = str2double(get(hObject,'String'));
    if isnan(input) set(hObject, 'String', 0); errordlg('Input must be a
number','Error');
    else eval([handles.EActive,'{4} = input']); end;
    guidata(hObject,handles)
    Update_Callback(hObject, eventdata, handles)
function epinf_Callback(hObject, eventdata, handles)
    input = str2double(get(hObject,'String'));
    if isnan(input) set(hObject, 'String', 0); errordlg('Input must be a
number','Error');
    else eval([handles.EActive,'{5} = input']); end;
    guidata(hObject,handles)
    Update_Callback(hObject, eventdata, handles)
function epKappa_Callback(hObject, eventdata, handles)
    input = str2double(get(hObject,'String'));
    if isnan(input) set(hObject, 'String', 0); errordlg('Input must be a
number','Error');
    else eval([handles.EActive,'{7} = input']); end;
    guidata(hObject,handles)
    Update_Callback(hObject, eventdata, handles)
%%%%%%%%% Second Set of OscPar for a second oscillator in Epsilon
function OscAmpEp2_Callback(hObject, eventdata, handles)
    input = str2double(get(hObject,'String'));
    if isnan(input) set(hObject, 'String', 0); errordlg('Input must be a
number','Error');
    else eval([handles.EActive,'{11} = input']); end;
    guidata(hObject,handles)
    Update_Callback(hObject, eventdata, handles)
function GEp2_Callback(hObject, eventdata, handles)
    input = str2double(get(hObject,'String'));
    if isnan(input) set(hObject, 'String', 0); errordlg('Input must be a
number','Error');
    else eval([handles.EActive,'{12} = input']); end;
    guidata(hObject,handles)
    Update_Callback(hObject, eventdata, handles)
function omega0Ep2_Callback(hObject, eventdata, handles)
    input = str2double(get(hObject,'String'));
    if isnan(input) set(hObject, 'String', 0); errordlg('Input must be a
number','Error');
    else eval([handles.EActive,'{13} = input']); end;
    guidata(hObject,handles)
    Update_Callback(hObject, eventdata, handles)
function omegapEp2_Callback(hObject, eventdata, handles)
    input = str2double(get(hObject,'String'));
    if isnan(input) set(hObject, 'String', 0); errordlg('Input must be a
number','Error');
    else eval([handles.EActive,'{14} = input']); end;
    guidata(hObject,handles)
    Update_Callback(hObject, eventdata, handles)
function epinf2_Callback(hObject, eventdata, handles)
    input = str2double(get(hObject,'String'));
    if isnan(input) set(hObject, 'String', 0); errordlg('Input must be a
number','Error');
    else eval([handles.EActive,'{15} = input']); end;
    guidata(hObject,handles)
    Update_Callback(hObject, eventdata, handles)
function epKappa2_Callback(hObject, eventdata, handles)
    input = str2double(get(hObject,'String'));
    if isnan(input) set(hObject, 'String', 0); errordlg('Input must be a
number','Error');
    else eval([handles.EActive,'{17} = input']); end;
    guidata(hObject,handles)
    Update_Callback(hObject, eventdata, handles)

function EpsBreakSlider_Callback(hObject, eventdata, handles)
    input = get(hObject,'Value');
    handles.Dispersion.EpsBreakPoint = input*handles.wCalcMax;
    guidata(hObject,handles)
    Update_Callback(hObject, eventdata, handles)

%---------------------------------------------------------------------
% -------------- Magnetic Parameter  ----------------------------------- [A, G, wo, wp,
inf, 'function']
function OscAmpMu_Callback(hObject, eventdata, handles)
    input = str2double(get(hObject,'String'));
    if isnan(input) set(hObject, 'String', 0); errordlg('Input must be a
number','Error');
    else eval([handles.MActive,'{1} = input']); end;
    guidata(hObject,handles)
    Update_Callback(hObject, eventdata, handles)
```

```
function GMu_Callback(hObject, eventdata, handles)
    input = str2double(get(hObject,'String'));
    if isnan(input) set(hObject, 'String', 0); errordlg('Input must be a
number','Error');
    else eval([handles.MActive,'{2} = input']); end;
    guidata(hObject,handles)
    Update_Callback(hObject, eventdata, handles)
function omega0Mu_Callback(hObject, eventdata, handles)
    input = str2double(get(hObject,'String'));
    if isnan(input) set(hObject, 'String', 0); errordlg('Input must be a
number','Error');
    else eval([handles.MActive,'{3} = input']); end;
    guidata(hObject,handles)
    Update_Callback(hObject, eventdata, handles)
function omegapMu_Callback(hObject, eventdata, handles)
    input = str2double(get(hObject,'String'));
    if isnan(input) set(hObject, 'String', 0); errordlg('Input must be a
number','Error');
    else eval([handles.MActive,'{4} = input']); end;
    guidata(hObject,handles)
    Update_Callback(hObject, eventdata, handles)
function muinf_Callback(hObject, eventdata, handles)
    input = str2double(get(hObject,'String'));
    if isnan(input) set(hObject, 'String', 0); errordlg('Input must be a
number','Error');
    else eval([handles.MActive,'{5} = input']); end;
    guidata(hObject,handles)
    Update_Callback(hObject, eventdata, handles)

function MueBreakSlider_Callback(hObject, eventdata, handles)
    input = get(hObject,'Value');
    handles.Dispersion.MueBreakPoint = input*handles.wCalcMax;
    guidata(hObject,handles)
    Update_Callback(hObject, eventdata, handles)


%---------------plotting function callbacks -------------------------
function EpxPlotMax_Callback(hObject, eventdata, handles)
    handles.Plot.EpxMax = str2double(get(handles.EpxPlotMax, 'String'));
    if isnan(handles.Plot.EpxMax) set(hObject, 'String', 0); errordlg('Input must be a
number','Error'); end
    guidata(hObject,handles)
    Update_Callback(hObject, eventdata, handles)
function EpxPlotMin_Callback(hObject, eventdata, handles)
    handles.Plot.EpxMin = str2double(get(handles.EpxPlotMin, 'String'));
    if isnan(handles.Plot.EpxMin) set(hObject, 'String', 0); errordlg('Input must be a
number','Error'); end
    guidata(hObject,handles)
    Update_Callback(hObject, eventdata, handles)
function MuzPlotMax_Callback(hObject, eventdata, handles)
    handles.Plot.MuzMax = str2double(get(handles.MuzPlotMax, 'String'));
    if isnan(handles.Plot.MuzMax) set(hObject, 'String', 0); errordlg('Input must be a
number','Error'); end
    guidata(hObject,handles)
    Update_Callback(hObject, eventdata, handles)
function MuzPlotMin_Callback(hObject, eventdata, handles)
    handles.Plot.MuzMin = str2double(get(handles.MuzPlotMin, 'String'));
    if isnan(handles.Plot.MuzMin) set(hObject, 'String', 0); errordlg('Input must be a
number','Error'); end
    guidata(hObject,handles)
    Update_Callback(hObject, eventdata, handles)
function TransPlotMax_Callback(hObject, eventdata, handles)
    handles.Plot.TransMax = str2double(get(handles.TransPlotMax, 'String'));
    if isnan(handles.Plot.TransMax) set(hObject, 'String', 0); errordlg('Input must be a
number','Error'); end
    guidata(hObject,handles)
    Update_Callback(hObject, eventdata, handles)
function TransPlotMin_Callback(hObject, eventdata, handles)
    handles.Plot.TransMin = str2double(get(handles.TransPlotMin, 'String'));
    if isnan(handles.Plot.TransMin) set(hObject, 'String', 0); errordlg('Input must be a
number','Error'); end
    guidata(hObject,handles)
    Update_Callback(hObject, eventdata, handles)
function Toffset_Callback(hObject, eventdata, handles)
    handles.Plot.Toffset = str2double(get(handles.Toffset, 'String'));
    if isnan(handles.Plot.Toffset) set(hObject, 'String', 0); errordlg('Input must be a
number','Error'); end
    guidata(hObject,handles)
    Update_Callback(hObject, eventdata, handles)

function checkbox1_Callback(hObject, eventdata, handles)
    handles.Plot.DataShow(1) = get(hObject,'Value');    guidata(hObject,handles);
Update_Callback(hObject, eventdata, handles)
function checkbox2_Callback(hObject, eventdata, handles)
    handles.Plot.DataShow(2) = get(hObject,'Value');    guidata(hObject,handles);
Update_Callback(hObject, eventdata, handles)
function checkbox3_Callback(hObject, eventdata, handles)
    handles.Plot.DataShow(3) = get(hObject,'Value');    guidata(hObject,handles);
Update_Callback(hObject, eventdata, handles)
function checkbox4_Callback(hObject, eventdata, handles)
    handles.Plot.DataShow(4) = get(hObject,'Value');    guidata(hObject,handles);
Update_Callback(hObject, eventdata, handles)
function checkbox5_Callback(hObject, eventdata, handles)
    handles.Plot.DataShow(5) = get(hObject,'Value');    guidata(hObject,handles);
Update_Callback(hObject, eventdata, handles)
function checkbox6_Callback(hObject, eventdata, handles)
    handles.Plot.DataShow(6) = get(hObject,'Value');    guidata(hObject,handles);
Update_Callback(hObject, eventdata, handles)

function checkbox1b_Callback(hObject, eventdata, handles)
    handles.Plot.DataShowCalc(1) = get(hObject,'Value');    guidata(hObject,handles);
Update_Callback(hObject, eventdata, handles)
function checkbox2b_Callback(hObject, eventdata, handles)
    handles.Plot.DataShowCalc(2) = get(hObject,'Value');    guidata(hObject,handles);
Update_Callback(hObject, eventdata, handles)
function checkbox3b_Callback(hObject, eventdata, handles)
    handles.Plot.DataShowCalc(3) = get(hObject,'Value');    guidata(hObject,handles);
Update_Callback(hObject, eventdata, handles)
function checkbox4b_Callback(hObject, eventdata, handles)
    handles.Plot.DataShowCalc(4) = get(hObject,'Value');    guidata(hObject,handles);
Update_Callback(hObject, eventdata, handles)
function checkbox5b_Callback(hObject, eventdata, handles)
    handles.Plot.DataShowCalc(5) = get(hObject,'Value');    guidata(hObject,handles);
Update_Callback(hObject, eventdata, handles)
function checkbox6b_Callback(hObject, eventdata, handles)
    handles.Plot.DataShowCalc(6) = get(hObject,'Value');    guidata(hObject,handles);
Update_Callback(hObject, eventdata, handles)


    % -------------OTHER CALLBACKS ------------------------
function edit15_Callback(hObject, eventdata, handles)
function FileMenu_Callback(hObject, eventdata, handles)
function OpenMenuItem_Callback(hObject, eventdata, handles)
    file = uigetfile('*.fig');
    if ~isequal(file, 0)
        open(file);
    end
    guidata(hObject,handles)
function PrintMenuItem_Callback(hObject, eventdata, handles)
    printdlg(handles.figure1)
function CloseMenuItem_Callback(hObject, eventdata, handles)
    selection = questdlg(['Close ' get(handles.figure1,'Name') '?'],...
                         ['Close ' get(handles.figure1,'Name') '...'],...
                         'Yes','No','Yes');
    if strcmp(selection,'No')
        return;
    end
    delete(handles.figure1)
```

```
function DispBreak_Callback(hObject, eventdata, handles)
    handles.Dispersion.Logical = get(hObject, 'Value')
    guidata(hObject,handles);
    Update_Callback(hObject, eventdata,handles);

function EpList_Callback(hObject, eventdata, handles)
    val = get(hObject, 'Value');   % here we update what oscillator parametres are
displayed based on which tensor element is selected.

    str = get(hObject, 'String');
    str = str(val);
    handles.EActive = ['handles.',char(str)];
    String='String';
    Value='Value';

    eval(['set(handles.OscAmpEp, String, ',  handles.EActive,'{1})']);
    eval(['set(handles.GEp,      String, ',  handles.EActive,'{2})']);
    eval(['set(handles.omega0Ep, String, ',  handles.EActive,'{3})']);
    eval(['set(handles.omegapEp, String, ',  handles.EActive,'{4})']);
    eval(['set(handles.epinf,    String, ',  handles.EActive,'{5})']);
    eval(['set(handles.EpForm,   Value,  ',  handles.EActive,'{6})']);
    eval(['set(handles.epKappa,  String, ',  handles.EActive,'{7})']);

    eval(['set(handles.OscAmpEp2, String, ', handles.EActive,'{11})']);
    eval(['set(handles.GEp2,      String, ', handles.EActive,'{12})']);
    eval(['set(handles.omega0Ep2, String, ', handles.EActive,'{13})']);
    eval(['set(handles.omegapEp2, String, ', handles.EActive,'{14})']);
    eval(['set(handles.epinf2,    String, ', handles.EActive,'{15})']);
    eval(['set(handles.epKappa2,  String, ', handles.EActive,'{17})']);

    guidata(hObject,handles)
    Update_Callback(hObject, eventdata, handles);


function AddEp_Callback(hObject, eventdata, handles)
    input = char(inputdlg('New Oscillator Parameter String', 'Input', 1));
    handles.EpListCell{length(handles.EpListCell)+1}=input;
    set(handles.EpList,'String',char(handles.EpListCell));
    guidata(hObject,handles);    Update_Setup(hObject, eventdata, handles)
function DelEp_Callback(hObject, eventdata, handles)


function MuList_Callback(hObject, eventdata, handles)
    val = get(hObject, 'Value');   % here we update what oscillator parametres are
displayed based on which tensor element is selected.
    str = get(hObject, 'String');
    str = str(val);
    handles.MActive = ['handles.',char(str)];
    String='String';
    Value='Value';
%     switch val
%         case 1
%             set(handles.OscAmpMu, 'String', handles.L1.MuOscPar{1,1}{1});
set(handles.GMu, 'String', handles.L1.MuOscPar{1,1}{2}); set(handles.omega0Mu, 'String',
handles.L1.MuOscPar{1,1}{3}); set(handles.omegapMu, 'String',
handles.L1.MuOscPar{1,1}{4}); set(handles.MuForm, 'value', handles.L1.MuOscPar{1,1}{6});

    eval(['set(handles.OscAmpMu, String, ',  handles.MActive,'{1})']);
    eval(['set(handles.GMu,      String, ',  handles.MActive,'{2})']);
    eval(['set(handles.omega0Mu, String, ',  handles.MActive,'{3})']);
    eval(['set(handles.omegapMu, String, ',  handles.MActive,'{4})']);
    eval(['set(handles.MuForm,   Value,  ',  handles.MActive,'{6})']);

    guidata(hObject,handles)
    Update_Callback(hObject, eventdata, handles);
```

```
function AddMu_Callback(hObject, eventdata, handles)
    input = char(inputdlg('New Oscillator Parameter String', 'Input', 1));
    handles.MuListCell{length(handles.MuListCell)+1}=input;
    set(handles.MuList,'String',char(handles.MuListCell));
    guidata(hObject,handles);    Update_Setup(hObject, eventdata, handles)
function DelMu_Callback(hObject, eventdata, handles)


function EL1isotropic_Callback(hObject, eventdata, handles)
    if(get(hObject,'Value'))
        handles.L1.EpOscPar{2,2} = handles.L1.EpOscPar{1,1};
        handles.L1.EpOscPar{3,3} = handles.L1.EpOscPar{1,1};
    end


function PolPopup_Callback(hObject, eventdata, handles)
    val = get(hObject,'Value');
    switch val
    case 1
        handles.Setup.Pol = 'spol';
    case 2
        handles.Setup.Pol = 'ppol';
    end
    guidata(hObject,handles)
    Update_Setup(hObject, eventdata, handles);
    Update_Callback(hObject, eventdata, handles);


function LayersPopup_Callback(hObject, eventdata, handles)
    val = get(hObject,'Value');
    switch val
    case 1
        handles.Setup.Layers = 1;
        set(handles.L2d, 'visible', 'off');
        set(handles.text22, 'visible', 'off');
        set(handles.L3d, 'visible', 'off');
        set(handles.text23, 'visible', 'off');
    case 2
        handles.Setup.Layers = 2;
        set(handles.L2d, 'visible', 'on');
        set(handles.text22, 'visible', 'on');
        set(handles.L3d, 'visible', 'off');
        set(handles.text23, 'visible', 'off');
    case 3
        handles.Setup.Layers = 3;
        set(handles.L2d, 'visible', 'on');
        set(handles.text22, 'visible', 'on');
        set(handles.L3d, 'visible', 'on');
        set(handles.text23, 'visible', 'on');
    end
    guidata(hObject,handles);    Update_Setup(hObject, eventdata, handles)
function LayersPopup_CreateFcn(hObject, eventdata, handles)
    if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor')) set(hObject,'BackgroundColor','white'); end



% Frequency set Max and Min.  wCalcMax wCalcMin
function wSetMin_Callback(hObject, eventdata, handles)
    handles.wCalcMin=str2double(get(hObject,'String'));
    guidata(hObject,handles)
    Update_Callback(hObject, eventdata, handles)
function wSetMax_Callback(hObject, eventdata, handles)
    handles.wCalcMax=str2double(get(hObject,'String'));
    guidata(hObject,handles)
    Update_Callback(hObject, eventdata, handles)
```

```
  % --------- THICKNESSES FOR 3 LAYERS ---------------
function L1d_Callback(hObject, eventdata, handles)
    handles.L1.d = str2double(get(hObject, 'String'));    guidata(hObject,handles);
Update_Callback(hObject, eventdata, handles)
function L2d_Callback(hObject, eventdata, handles)
    handles.L2.d = str2double(get(hObject, 'String'));    guidata(hObject,handles);
Update_Callback(hObject, eventdata, handles)
function L3d_Callback(hObject, eventdata, handles)
    handles.L3.d = str2double(get(hObject, 'String'));    guidata(hObject,handles);
Update_Callback(hObject, eventdata, handles)


% ------------ FITTING FUNCTION CALLBACKS ------------
function TFitRes_Callback(hObject, eventdata, handles)
function RFitRes_Callback(hObject, eventdata, handles)
function edit31_Callback(hObject, eventdata, handles)
function NUFitRes_Callback(hObject, eventdata, handles)
function FitTo_Callback(hObject, eventdata, handles)
    val = get(hObject, 'value');
    switch val
        case 1 %transmission
            handles.Fit.FitType = 'T';
        case 2 %reflection
            handles.Fit.FitType = 'R';
        case 3 %W Sum
            handles.Fit.FitType = 'WS';
        case 4 %UW  Sum
            handles.Fit.FitType = 'UWS';
    end
    guidata(hObject,handles)

function FitIter_Callback(hObject, eventdata, handles)
    handles.Fit.MaxIter = str2double(get(hObject,'String'));
    guidata(hObject,handles)

function Bound_Callback(hObject, eventdata, handles)
    handles.Fit.Bound = str2double(get(hObject,'String'));
    guidata(hObject,handles)

function RunFit_Callback(hObject, eventdata, handles)
    handles = Fit_NLBounded(hObject, eventdata, handles);  % Run The Fitting routine.
Takes current ocillator parameters as starting.
    % re-set the display values on the gui to match values modified in RunFit
    set(handles.OscAmpEp, 'String', handles.L1.EpOscPar{1,1}{1});
    set(handles.GEp,      'String', handles.L1.EpOscPar{1,1}{2});
    set(handles.omega0Ep, 'String', handles.L1.EpOscPar{1,1}{3});
    set(handles.omegapEp, 'String', handles.L1.EpOscPar{1,1}{4});
    set(handles.epinf,    'String', handles.L1.EpOscPar{1,1}{5});
    set(handles.OscAmpMu, 'String', handles.L1.MuOscPar{3,3}{1});
    set(handles.GMu,      'String', handles.L1.MuOscPar{3,3}{2});
    set(handles.omega0Mu, 'String', handles.L1.MuOscPar{3,3}{3});
    set(handles.omegapMu, 'String', handles.L1.MuOscPar{3,3}{4});
    set(handles.ResidueOut, 'String', handles.Fit.ResNorm);
    guidata(hObject,handles)
    Update_Callback(hObject, eventdata, handles)


function ResidueOut_Callback(hObject, eventdata, handles)

function reset_Callback(hObject, eventdata, handles)
    handles = Initialize_Material_Tensors(hObject, eventdata, handles);
    guidata(hObject,handles)
    Update_Callback(hObject, eventdata, handles)

function EpForm_Callback(hObject, eventdata, handles)
    Num = get(handles.EpList, 'value');
    val = get(hObject, 'value');
    if(Num <= 3)
        handles.L1.EpOscPar{Num,Num}{6}= val;
    elseif (Num <= 6)
        handles.L2.EpOscPar{Num-3,Num-3}{6} = val;
    elseif (Num <= 9)
        handles.L3.EpOscPar{Num-6,Num-6}{6} = val;
    end
    guidata(hObject,handles)
    Update_Callback(hObject, eventdata, handles);

function MuForm_Callback(hObject, eventdata, handles)
    Num = get(handles.MuList, 'value');
    val = get(hObject, 'value');
    if(Num <= 3)
        handles.L1.MuOscPar{Num,Num}{6}= val;
    elseif (Num <= 6)
        handles.L2.MuOscPar{Num-3,Num-3}{6} = val;
    elseif (Num <= 9)
        handles.L3.MuOscPar{Num-6,Num-6}{6} = val;
    end;
    guidata(hObject,handles)
    Update_Callback(hObject, eventdata, handles);

function ResLim_Callback(hObject, eventdata, handles)
    handles.ResLimit = str2double(get(hObject,'string'));
    guidata(hObject,handles);    Update_Callback(hObject, eventdata, handles);

function EL2isotropic_Callback(hObject, eventdata, handles)
    guidata(hObject,handles);    Update_Callback(hObject, eventdata, handles);
function EL3isotropic_Callback(hObject, eventdata, handles)
    guidata(hObject,handles);    Update_Callback(hObject, eventdata, handles);
function ML1isotropic_Callback(hObject, eventdata, handles)
    guidata(hObject,handles);    Update_Callback(hObject, eventdata, handles);
function ML2isotropic_Callback(hObject, eventdata, handles)
    guidata(hObject,handles);    Update_Callback(hObject, eventdata, handles);
function ML3isotropic_Callback(hObject, eventdata, handles)
    guidata(hObject,handles);    Update_Callback(hObject, eventdata, handles);

  %!!!!!!!!!!!!! CREATE FUNCTIONS!!!!!!!!!!!!!!!
function OscAmpEp_CreateFcn(hObject, eventdata, handles)
    if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor')) set(hObject,'BackgroundColor','white');    end
function GEp_CreateFcn(hObject, eventdata, handles)
    if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor')) set(hObject,'BackgroundColor','white');    end
function omega0Ep_CreateFcn(hObject, eventdata, handles)
    if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor')) set(hObject,'BackgroundColor','white');    end
function omegapEp_CreateFcn(hObject, eventdata, handles)
    if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor')) set(hObject,'BackgroundColor','white');    end
function epinf_CreateFcn(hObject, eventdata, handles)
    if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor')) set(hObject,'BackgroundColor','white');    end
function OscAmpMu_CreateFcn(hObject, eventdata, handles)
    if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))set(hObject,'BackgroundColor','white');    end
function GMu_CreateFcn(hObject, eventdata, handles)
    if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor')) set(hObject,'BackgroundColor','white');    end
function omega0Mu_CreateFcn(hObject, eventdata, handles)
    if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor')) set(hObject,'BackgroundColor','white');    end
function omegapMu_CreateFcn(hObject, eventdata, handles)
    if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor')) set(hObject,'BackgroundColor','white'); end
function EpxPlotMax_CreateFcn(hObject, eventdata, handles)
    if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor')) set(hObject,'BackgroundColor','white');    end
function EpxPlotMin_CreateFcn(hObject, eventdata, handles)
```

```
    if ispc && isequal(get(hObject,'BackgroundColor'), 
get(0,'defaultUicontrolBackgroundColor')) set(hObject,'BackgroundColor','white');    end
function MuzPlotMax_CreateFcn(hObject, eventdata, handles)
    if ispc && isequal(get(hObject,'BackgroundColor'), 
get(0,'defaultUicontrolBackgroundColor')) set(hObject,'BackgroundColor','white');    end
function TransPlotMax_CreateFcn(hObject, eventdata, handles)
    if ispc && isequal(get(hObject,'BackgroundColor'), 
get(0,'defaultUicontrolBackgroundColor')) set(hObject,'BackgroundColor','white');    end
function TransPlotMin_CreateFcn(hObject, eventdata, handles)
    if ispc && isequal(get(hObject,'BackgroundColor'), 
get(0,'defaultUicontrolBackgroundColor')) set(hObject,'BackgroundColor','white');    end
function Toffset_CreateFcn(hObject, eventdata, handles)
    if ispc && isequal(get(hObject,'BackgroundColor'), 
get(0,'defaultUicontrolBackgroundColor')) set(hObject,'BackgroundColor','white');    end
function edit15_CreateFcn(hObject, eventdata, handles)
    if ispc && isequal(get(hObject,'BackgroundColor'), 
get(0,'defaultUicontrolBackgroundColor')) set(hObject,'BackgroundColor','white');    end
function popupmenu2_CreateFcn(hObject, eventdata, handles)
    if ispc && isequal(get(hObject,'BackgroundColor'), 
get(0,'defaultUicontrolBackgroundColor')) set(hObject,'BackgroundColor','white'); end
function popupmenu3_CreateFcn(hObject, eventdata, handles)
    if ispc && isequal(get(hObject,'BackgroundColor'), 
get(0,'defaultUicontrolBackgroundColor'))        set(hObject,'BackgroundColor','white'); 
end
function EpList_CreateFcn(hObject, eventdata, handles)
    if ispc && isequal(get(hObject,'BackgroundColor'), 
get(0,'defaultUicontrolBackgroundColor'))    set(hObject,'BackgroundColor','white'); 
end
function ResidueOut_CreateFcn(hObject, eventdata, handles)
    if ispc && isequal(get(hObject,'BackgroundColor'), 
get(0,'defaultUicontrolBackgroundColor'))    set(hObject,'BackgroundColor','white'); end
function MuForm_CreateFcn(hObject, eventdata, handles)
    if ispc && isequal(get(hObject,'BackgroundColor'), 
get(0,'defaultUicontrolBackgroundColor'));    set(hObject,'BackgroundColor','white'); end
function EpForm_CreateFcn(hObject, eventdata, handles)
    if ispc && isequal(get(hObject,'BackgroundColor'), 
get(0,'defaultUicontrolBackgroundColor')); set(hObject,'BackgroundColor','white'); end
function edit36_CreateFcn(hObject, eventdata, handles)
    if ispc && isequal(get(hObject,'BackgroundColor'), 
get(0,'defaultUicontrolBackgroundColor'));  set(hObject,'BackgroundColor','white'); end
function edit37_CreateFcn(hObject, eventdata, handles)
    if ispc && isequal(get(hObject,'BackgroundColor'), 
get(0,'defaultUicontrolBackgroundColor')); set(hObject,'BackgroundColor','white'); end
function Bound_CreateFcn(hObject, eventdata, handles)
    if ispc && isequal(get(hObject,'BackgroundColor'), 
get(0,'defaultUicontrolBackgroundColor'))  set(hObject,'BackgroundColor','white'); end
function L1d_CreateFcn(hObject, eventdata, handles)
    if ispc && isequal(get(hObject,'BackgroundColor'), 
get(0,'defaultUicontrolBackgroundColor')); set(hObject,'BackgroundColor','white'); end
function L2d_CreateFcn(hObject, eventdata, handles)
    if ispc && isequal(get(hObject,'BackgroundColor'), 
get(0,'defaultUicontrolBackgroundColor')); set(hObject,'BackgroundColor','white'); end
function L3d_CreateFcn(hObject, eventdata, handles)
    if ispc && isequal(get(hObject,'BackgroundColor'), 
get(0,'defaultUicontrolBackgroundColor')); set(hObject,'BackgroundColor','white'); end
function TFitRes_CreateFcn(hObject, eventdata, handles)
    if ispc && isequal(get(hObject,'BackgroundColor'), 
get(0,'defaultUicontrolBackgroundColor'))        set(hObject,'BackgroundColor','white'); 
end;
function FitIter_CreateFcn(hObject, eventdata, handles)
    if ispc && isequal(get(hObject,'BackgroundColor'), 
get(0,'defaultUicontrolBackgroundColor'))    set(hObject,'BackgroundColor','white'); end
function RFitRes_CreateFcn(hObject, eventdata, handles)
    if ispc && isequal(get(hObject,'BackgroundColor'), 
get(0,'defaultUicontrolBackgroundColor'))        set(hObject,'BackgroundColor','white'); 
end
function edit31_CreateFcn(hObject, eventdata, handles)
```

```
    if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))        set(hObject,'BackgroundColor','white');
end
function NUFitRes_CreateFcn(hObject, eventdata, handles)
    if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))        set(hObject,'BackgroundColor','white');
end
function FitTo_CreateFcn(hObject, eventdata, handles)
    if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))    set(hObject,'BackgroundColor','white');end
function PolPopup_CreateFcn(hObject, eventdata, handles)
    if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor')) set(hObject,'BackgroundColor','white'); end
function MuList_CreateFcn(hObject, eventdata, handles)
    if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))    set(hObject,'BackgroundColor','white'); end
function wSetMax_CreateFcn(hObject, eventdata, handles)
    if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))    set(hObject,'BackgroundColor','white');end
function wSetMin_CreateFcn(hObject, eventdata, handles)
    if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))    set(hObject,'BackgroundColor','white');end
function MuzPlotMin_CreateFcn(hObject, eventdata, handles)
    if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor')) set(hObject,'BackgroundColor','white');    end
function L1dVar_CreateFcn(hObject, eventdata, handles)
    if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))   set(hObject,'BackgroundColor','white');
end
function L3dVar_CreateFcn(hObject, eventdata, handles)
    if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))   set(hObject,'BackgroundColor','white');
end
function L2dVar_CreateFcn(hObject, eventdata, handles)
    if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))   set(hObject,'BackgroundColor','white'); end
function edit38_CreateFcn(hObject, eventdata, handles)
    if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor')) set(hObject,'BackgroundColor','white');  end
function ResLim_CreateFcn(hObject, eventdata, handles)
    if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))    set(hObject,'BackgroundColor','white'); end
function GEp2_CreateFcn(hObject, eventdata, handles)
    if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))    set(hObject,'BackgroundColor','white'); end
function OscAmpEp2_CreateFcn(hObject, eventdata, handles)
    if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))    set(hObject,'BackgroundColor','white');end
function omega0Ep2_CreateFcn(hObject, eventdata, handles)
    if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))    set(hObject,'BackgroundColor','white');end
function omegapEp2_CreateFcn(hObject, eventdata, handles)
    if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))    set(hObject,'BackgroundColor','white'); end
function epinf2_CreateFcn(hObject, eventdata, handles)
    if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))     set(hObject,'BackgroundColor','white'); end
function muinf_CreateFcn(hObject, eventdata, handles)
    if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))    set(hObject,'BackgroundColor','white'); end
function epKappa2_CreateFcn(hObject, eventdata, handles)
    if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))     set(hObject,'BackgroundColor','white'); end
function epKappa_CreateFcn(hObject, eventdata, handles)
    if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))     set(hObject,'BackgroundColor','white');
end
function EpsBreakSlider_CreateFcn(hObject, eventdata, handles)
```

```
    if isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
set(hObject,'BackgroundColor',[.9 .9 .9]);     end
function MueBreakSlider_CreateFcn(hObject, eventdata, handles)
    if isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
set(hObject,'BackgroundColor',[.9 .9 .9]);    end
function NumA_CreateFcn(hObject, eventdata, handles)
    if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))set(hObject,'BackgroundColor','white'); end
function NumB_CreateFcn(hObject, eventdata, handles)
    if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))     set(hObject,'BackgroundColor','white'); end
function NumC_CreateFcn(hObject, eventdata, handles)
    if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor')) set(hObject,'BackgroundColor','white'); end

function loadF_Callback(hObject, eventdata, handles)    % Note that currently, if loaded
variable has layers other than L1{1,1} changed, this will not be
    str = '''SavedState1.mat'''                         % updated - as calculation is
only called by Initialize_Material_Tensors.  This needs to be implimented here.
    LoadFile(hObject, eventdata, handles, str)
function loadF2_Callback(hObject, eventdata, handles)
    LoadFile(hObject, eventdata, handles, '''SavedState2.mat''')
function loadF3_Callback(hObject, eventdata, handles)
    LoadFile(hObject, eventdata, handles, '''SavedState3.mat''')
function loadF4_Callback(hObject, eventdata, handles)
    LoadFile(hObject, eventdata, handles, '''SavedState4.mat''')
function loadF5_Callback(hObject, eventdata, handles)
    LoadFile(hObject, eventdata, handles, '''SavedState5.mat''')

function LoadFile(hObject, eventdata, handles, SaveName);
    str = ['handlesTEMP = load(',SaveName,')'];
    eval(['handlesTEMP = load(',SaveName,')']);
%     handlesTEMP.L1.Disorder = handles.L1.Disorder;
%     handlesTEMP.L2.Disorder = handles.L2.Disorder;
%     handlesTEMP.L3.Disorder = handles.L3.Disorder;
    handles.L1 = handlesTEMP.L1;
    handles.L2 = handlesTEMP.L2;
    handles.L3 = handlesTEMP.L3;
    handles.Fit = handlesTEMP.Fit;
    handles.Plot = handlesTEMP.Plot;
    handles.Setup = handlesTEMP.Setup;
    handles.DataCalc = handlesTEMP.DataCalc;
    handles.wCalcMin = handlesTEMP.wCalcMin;
    handles.wCalcMax = handlesTEMP.wCalcMax;
    handles.Dispersion = handlesTEMP.Dispersion;
    set(handles.L1d, 'String',handles.L1.d);
    set(handles.L2d, 'String',handles.L2.d);
    set(handles.L3d, 'String',handles.L3.d);
    guidata(hObject,handles)
    Update_Callback(hObject, eventdata, handles);

function saveF_Callback(hObject, eventdata, handles)
    selection = questdlg(['Save ?'],['Save ?'],'Yes','No','Yes');
    if strcmp(selection,'Yes')
        save('SavedState1.mat', '-struct', 'handles'); end
function saveF2_Callback(hObject, eventdata, handles)
    selection = questdlg(['Save ?'],['Save ?'],'Yes','No','Yes');
    if strcmp(selection,'Yes')
        save('SavedState2.mat', '-struct', 'handles'); end
function saveF3_Callback(hObject, eventdata, handles)
    selection = questdlg(['Save ?'],['Save ?'],'Yes','No','Yes');
    if strcmp(selection,'Yes')
        save('SavedState3.mat', '-struct', 'handles'); end
function saveF4_Callback(hObject, eventdata, handles)
    selection = questdlg(['Save ?'],['Save ?'],'Yes','No','Yes');
    if strcmp(selection,'Yes')
        save('SavedState4.mat', '-struct', 'handles'); end
function saveF5_Callback(hObject, eventdata, handles)
    selection = questdlg(['Save ?'],['Save ?'],'Yes','No','Yes');
    if strcmp(selection,'Yes')
        save('SavedState5.mat', '-struct', 'handles'); end

 function WriteSol_Callback(hObject, eventdata, handles)
     for ii=1:5
        SaveName = ['''SavedState',int2str(ii),'.mat'''];
        WriteName = ['WriteOut',int2str(ii),'.mat'];
        eval(['WriteBuffer = load(',SaveName,')']);
        EpsR(:,ii) = real(WriteBuffer.L1.Epsilon{1,1})';
        EpsI(:,ii) = imag(WriteBuffer.L1.Epsilon{1,1})';
        MueR(:,ii) = real(WriteBuffer.L1.Mue{1,1})';
        MueI(:,ii) = imag(WriteBuffer.L1.Mue{1,1})';
    end
        save('EpsROut.mat', 'EpsR', '-ascii', '-tabs');
        save('EpsIOut.mat', 'EpsI', '-ascii', '-tabs');
        save('MueROut.mat', 'MueR', '-ascii', '-tabs');
        save('MueIOut.mat', 'MueI', '-ascii', '-tabs');

% Disorder Broadening variables
function L1dVar_Callback(hObject, eventdata, handles)
    handles.L1.Disorder.Fraction = str2double(get(hObject, 'String'));
guidata(hObject,handles);    Update_Callback(hObject, eventdata, handles)
function L2dVar_Callback(hObject, eventdata, handles)
function L3dVar_Callback(hObject, eventdata, handles)

function NumA_Callback(hObject, eventdata, handles)
    handles.L1.Disorder.NumSumPoints = str2double(get(hObject, 'String'));
guidata(hObject,handles);    Update_Callback(hObject, eventdata, handles)
function NumB_Callback(hObject, eventdata, handles)
function NumC_Callback(hObject, eventdata, handles)
```

Bibliography.

1. Smith, D.R., et al., *Composite Medium with Simultaneously Negative Permeability and Permittivity.* Physical Review Letters, 2000. **84**(18): p. 4184-4187.

2. Veselago, V.G., Sov. Phys. Usp, 1968. **10**(509).

3. Andrew, A.H., B.B. Jeffrey, and L.C. Isaac, *Experimental Observations of a Left-Handed Material That Obeys Snell's Law.* Physical Review Letters, 2003. **90**(13): p. 137401.

4. Vodo, P., et al., *Microwave photonic crystal with tailor-made negative refractive index.* Applied Physics Letters, 2004. **85**(10): p. 1858-1860.

5. Hsu, A.-C., et al., *Far-Infrared Resonance in Split Ring Resonators.* Japanese Journal of Applied Physics, 2004. **43**: p. L176-L179.

6. Linden, S., et al., *Magnetic response of metamaterials at 100 terahertz.* Science (New York, N.Y.), 2004. **306**(5700): p. 1351-1353.

7. Zhou, J., et al., *Saturation of the Magnetic Response of Split-Ring Resonators at Optical Frequencies.* Physical Review Letters, 2005. **95**(22): p. 223902.

8. Shalaev, V.M., et al., *Negative index of refraction in optical metamaterials.* Opt. Lett., 2005. **30**(24): p. 3356-3358.

9. Pendry, J.B., D. Schurig, and D.R. Smith, *Controlling electromagnetic fields.* Science, 2006. **312**(5781): p. 1780-2.

10. Cummer, S.A., et al., *Full-wave simulations of electromagnetic cloaking structures.* Phys Rev E Stat Nonlin Soft Matter Phys, 2006. **74**(3 Pt 2): p. 036621.

11. Schurig, D., et al., *Metamaterial electromagnetic cloak at microwave frequencies.* Science, 2006. **314**(5801): p. 977-80.

12. Shelby, R.A., D.R. Smith, and S. Schultz, *Experimental verification of a negative index of refraction.* Science, 2001. **292**(5514): p. 77-9.

13. Kin, L., et al., *Free-space focused-beam characterization of left-handed materials.* Applied Physics Letters, 2003. **82**(15): p. 2535-2537.

14. Parazzoli, C.G., et al., *Experimental verification and simulation of negative index of refraction using Snell's law.* Phys Rev Lett, 2003. **90**(10): p. 107401.

15. Parazzoli, C.G., et al., *Performance of a negative index of refraction lens.* Applied Physics Letters, 2004. **84**(17): p. 3232-3234.

16. Schurig, D. and D.R. Smith, *Negative index lens aberrations.* Phys Rev E Stat Nonlin Soft Matter Phys, 2004. **70**(6 Pt 2): p. 065601.

17. Koschny, T., et al., *Effective Medium Theory of Left-Handed Materials.* Physical Review Letters, 2004. **93**(10): p. 107402.

18. Starr, A.F., et al., *Fabrication and characterization of a negative-refractive-index composite metamaterial.* Physical Review B (Condensed Matter and Materials Physics), 2004. **70**(11): p. 113102.

19. Smith, D.R., et al., *Gradient index metamaterials.* Physical Review E (Statistical, Nonlinear, and Soft Matter Physics), 2005. **71**(3): p. 036609.

20. Smith, D.R., et al., *Electromagnetic parameter retrieval from inhomogeneous metamaterials.* Physical Review E (Statistical, Nonlinear, and Soft Matter Physics), 2005. **71**(3): p. 036617.

21. Kong, J.A., *Electromagnetic wave theory*. 2nd ed. 1990, New York: Wiley.

22. Driscoll, T., et al., *Electromagnetic characterization of planar metamaterials by oblique angle spectroscopic measurements.* Phys. Rev. B, 2007. **75**: p. 115114.

23. Smith, D.R., D. Schurig, and J.J. Mock, *Characterization of a planar artificial magnetic metamaterial surface.* Physical Review E (Statistical, Nonlinear, and Soft Matter Physics), 2006. **74**(3): p. 036604.

24. Derov, J.S., et al., IEEE Microwaves and wireless communications letters, 2005. **15**(9).

25. Schurig, D., International Journal of Numerical Modeling, 2006. **19**(2): p. 215.

26. Jackson, J.D., *Classical electrodynamics*. 3rd ed. 1999, New York: Wiley.

27. Driscoll, T., et al., *Free-space microwave focusing by a negative-index gradient lens.* Applied Physics Letters, 2006. **88**(8): p. 081101.

28. Fang, N., et al., *Sub-diffraction-limited optical imaging with a silver superlens.* Science, 2005. **308**(5721): p. 534-7.

29. Shuang, Z., et al., *Experimental Demonstration of Near-Infrared Negative-Index Metamaterials.* Physical Review Letters, 2005. **95**(13): p. 137404.

30. Th, K., et al., *Impact of inherent periodic structure on effective medium description of left-handed and related metamaterials.* Physical Review B (Condensed Matter and Materials Physics), 2005. **71**(24): p. 245105.

31. Smith, D.R., et al., *Direct calculation of permeability and permittivity for a left-handed metamaterial.* Applied Physics Letters, 2000. **77**(14): p. 2246-2248.

32. Andrey, K.S., S. Gennady, and M.S. Vladimir, *Magnetic plasmon resonance.* Physical Review E (Statistical, Nonlinear, and Soft Matter Physics), 2006. **73**(3): p. 036609.

33. Woolum, J.A., *WVASE published by J.A. Woolum co.*

34. references, m.b.W.

35. Huade, Y., J. Blaine, and B.J. Ralph, *Optical anisotropic-dielectric response of mercuric iodide.* Physical Review B (Condensed Matter), 1997. **56**(15): p. 9414-9421.

36. Shamonin, M., et al., Microwave Optical Technology Letters, 2005. **44**: p. 133-136.

37. Gorkunov, M., et al., *Effective magnetic properties of a composite material with circular conductive elements.* european journal of physics B, 2002. **28**(3): p. p263.

38. *FR4 is a Flame Retardant Laminate, a fiberglass based common circuit board material.*

39. Rachford, F.J., et al., *Calculations and measurements of wire and/or split-ring negative index media.* Physical Review E (Statistical, Nonlinear, and Soft Matter Physics), 2002. **66**(3): p. 036613.

40. Shelby, R.A., et al., *Microwave transmission through a two-dimensional, isotropic, left-handed metamaterial.* Applied Physics Letters, 2001. **78**(4): p. 489-491.

41. Yen, T.J., et al., *Terahertz magnetic response from artificial materials.* Science, 2004. **303**(5663): p. 1494-6.

42. Born, M. and E. Wolf, *Principles of optics : electromagnetic theory of propagation, interference and diffraction of light*. 7th (expanded) ed. 1999, Cambridge ; New York: Cambridge University Press.

43. Pendry, J.B., et al., *Extremely low frequency plasmons in metallic mesostructures.* Physical Review Letters, 1996. **76**(25): p. 4773-4776.

44. Pendry, J.B., et al., *Magnetism from conductors and enhanced nonlinear phenomena.* Microwave Theory and Techniques, IEEE Transactions on, 1999. **47**(11): p. 2075-2084.

45. Berreman, D.W., *Infrared Absorption at Longitudinal Optic Frequency in Cubic Crystal Films.* Physical Review, 1963. **130**(6): p. 2193.

46. Starr, A.F., et al., *Angle resolved microwave spectrometer for metamaterial studies.* Review of Scientific Instruments, 2004. **75**(4): p. 820-825.

47. Padilla, W.J., et al., *Dynamical Electric and Magnetic Metamaterial Response at Terahertz Frequencies.* Physical Review Letters, 2006. **96**(10): p. 107401.

48. Wang, B., et al., *Surface waves in photonic crystal slabs.* Physical Review B (Condensed Matter and Materials Physics), 2006. **74**(19): p. 195104.

49. *Wavenumbers (cm-1) are the common units of infrared spectroscopy, 33 cm-1 = 1 THz.*

50. *The frequency resolution in FTIR is directly related to the displacement of the modulating mirror. Setting the spectrometer for lower resolution allows for faster scans with less noise, at the expense of omitting sharp features (such as our substrate Fabrey-Perot fringes) in the spectra.*

51. Bogdan-Ioan, P. and A.C. Steven, *Determining the effective electromagnetic properties of negative-refractive-index metamaterials from internal fields.* Physical Review B (Condensed Matter and Materials Physics), 2005. **72**(16): p. 165102.

52. Azzam, R.M.A. and N.M. Bashara, *Ellipsometry and polarized light*. 1977, Amsterdam ; New York New York: North-Holland Pub. Co. ;
sole distributors for the U.S.A. and Canada Elsevier North-Holland.

53. Loewenstein, E.V., D.R. Smith, and R.L. Morgan, *Optical constants of far infrared materials. 2: Crystalline solids.* Appl. Opt., 1973. **12**(2): p. 398.

54. *Dow Chemicals Product datasheet:.* http://www.dow.com/cyclotene/solution/highfreq.htm.

55. Dolling, G., et al., *Simultaneous negative phase and group velocity of light in a metamaterial.* Science, 2006. **312**(5775): p. 892-4.

56. Maxim, V.G., et al., *Effect of microscopic disorder on magnetic properties of metamaterials.* Physical Review E (Statistical, Nonlinear, and Soft Matter Physics), 2006. **73**(5): p. 056605.

57. Pendry, J.B., *Negative Refraction Makes a Perfect Lens.* Physical Review Letters, 2000. **85**(18): p. 3966-3969.

58. Chen, H.T., et al., *Active terahertz metamaterial devices.* Nature, 2006. **444**(7119): p. 597-600.

59. Liu, Z., et al., *Far-field optical hyperlens magnifying sub-diffraction-limited objects.* Science, 2007. **315**(5819): p. 1686.

60. Shadrivov, I.V., S.K. Morrison, and Y.S. Kivshar, *Tunable split-ring resonators for nonlinear negative-index metamaterials.* Opt. Express, 2006. **14**(20): p. 9344-9349.

61. Chen, H.-T., et al., *Experimental demonstration of frequency-agile terahertz metamaterials.* Nat Photon, 2008. **2**(5): p. 295-298.

62. Chang, Y.J., et al., *Surface versus bulk characterizations of electronic inhomogeneity in a VO[sub 2] thin film.* Physical Review B (Condensed Matter and Materials Physics), 2007. **76**(7): p. 075118.

63. Qazilbash, M.M., et al., *Mott transition in VO2 revealed by infrared spectroscopy and nano-imaging.* Science, 2007. **318**(5857): p. 1750-3.

64. Zylbersztejn, A. and N.F. Mott, *Metal-insulator transition in vanadium dioxide.* Physical Review B (Solid State), 1975. **11**(11): p. 4383-4395.

65. Willie, J.P., N.B. Dimitri, and R.S. David, *Negative refractive index materials.* Materials Today, 2006. **9**(7-8): p. p28.

66. Driscoll, T., et al., *Tuned permeability in terahertz split-ring resonators for devices and sensors.* Applied Physics Letters, 2007. **91**(6): p. 062511.

67. O'Hara, J.F., et al., *Thin-film sensing with planar terahertz metamaterials: sensitivity and limitations.* Opt. Express, 2008. **16**(3): p. 1786-1795.

68. Katsarakis, N., et al., *Electric coupling to the magnetic resonance of split ring resonators.* Applied Physics Letters, 2004. **84**(15): p. 2943-2945.

69. Billard, D., F. Gervais, and B. Piriou, Journal of Infrared and Millimeter Waves, 1980. **1**(4).

70. Liu, R., et al., *Description and explanation of electromagnetic behaviors in artificial metamaterials based on effective medium theory.* Physical Review E (Statistical, Nonlinear, and Soft Matter Physics), 2007. **76**(2): p. 026606-8.

71. Driscoll, T., et al., *Quantitative investigation of a terahertz artificial magnetic resonance using oblique angle spectroscopy.* Applied Physics Letters, 2007. **90**(9): p. 092508.

72. Koschny, T., et al., *Resonant and antiresonant frequency dependence of the effective parameters of metamaterials.* Physical Review E (Statistical, Nonlinear, and Soft Matter Physics), 2003. **68**(6): p. 065602.

73. Rini, M., et al., *Photoinduced phase transition in VO2 nanocrystals: ultrafast control of surface-plasmon resonance.* Opt. Lett., 2005. **30**(5): p. 558-560.

74. Kim, H.T., et al., *Mechanism and observation of Mott transition in VO2-based two- and three-terminal devices.* New Journal of Physics, 2004. **6**(52).

75. Smith, D.R., *Applied physics. How to build a superlens.* Science, 2005. **308**(5721): p. 502-3.

76. Cai, W., et al., *Optical Cloaking with Non-Magnetic Metamaterials.* arXiv, 2007. **0611242**.

77. Liu, H., et al., *Design of antenna radome composed of metamaterials for high gain.* IEEE Antennas and propagation symposion proceedings, 2006: p. 19-22.

78. Shamonin, M., et al., *Properties of a metamaterial element: Analytical solutions and numerical simulations for a singly split double ring.* Journal of Applied Physics, 2004. **95**(7).

79. Aydin, K., et al., *Investigation of magnetic resonances for different split-ring resonator parameters and designs.* New Journal of Physics, 2005. **7**(168).

80. *The amplitude change (as modeled in WVASE) from adding a 500nm layer of Si spheres is less than 0.05 percent, below the noise level for this experiment.*

81. Brucherseifer, M., et al., *Label-free probing of the binding state of DNA by time-domain terahertz sensing.* Applied Physics Letters, 2000. **77**(24): p. 4049-4051.