Obtaining Liquid Crystal Dielectric Spectrums

In this experiment, we use two different De Vries materials with virtually no layer shrinkage. The first material is a 8422[2F3] sample. Using the Ocean Optics spectrometer, we determined the width of the sample to by 5μm. The second material is a TSiKN65 sample and the width is measured to be 5μm. We use a Hewlett-Packard Impedance Analyzer 4192A LF to apply a 0.1V oscillating current through each sample and measured the magnitude of the impedance and it's complex angle. The dielectric values are determined from these impedances using the conversion described the previous section. Using a custom National Instruments Labview program to automate the experiment, we determine these values over a range of frequencies. This technique is repeated at multiple temperatures (near the SmA-SmC transition) to obtain a 3D surface spectrum of the dielectric absorption.

Plot the 3D surface to show all data taken over all temperatures

Initial setup for IPython notebook

In [2]:
```
%pylab inline
from mpl_toolkits.mplot3d import Axes3D
import scipy.optimize as spo
```

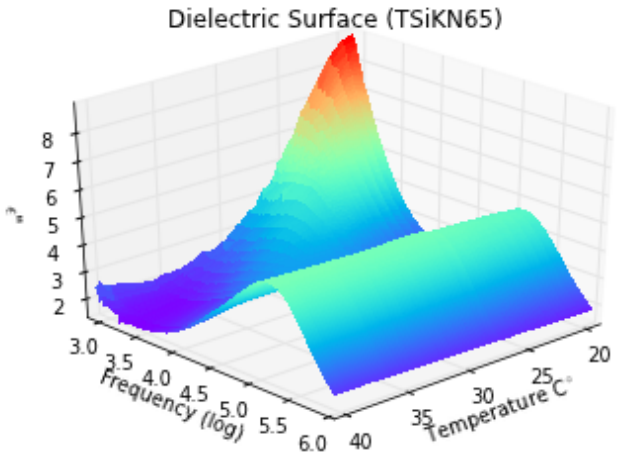
```
Welcome to pylab, a matplotlib-based Python environment [backend:
module://IPython.kernel.zmq.pylab.backend_inline].
For more information, type 'help(pylab)'.
```

TSiKN65 Surface Plot

In [18]:
```
files = ['test01','test02','test03','test04','test05','test06','test07','test08',
        'test09','test10','test11','test12','test13','test14','test15','test16',
        'test17','test18','test19','test20','test21','test22','test23','test24',
        'test25','test26','test27','test28','test29','test30','test31','test32',
        'test33','test34','test35','test36','test37','test38','test39','test40',
        'test41','test42','test43','test44','test45','test46','test47','test48',
        'test49','test50']

n = len(files)/2
A = .01**2;
eps0 = 8.854187e-12;
d = 5e-6;
e1=[]
e2=[]
Tem = [40.5, 39.5, 36.1, 35.1, 34.1, 33, 31.9, 30.9, 29.9, 29, 28, 27, 25.9, 25.5,
25.2,
               25, 24.6, 24.4, 24, 23.5, 22.9, 22, 21, 20.1, 19.2]
temp= []
fig = plt.figure()
```

```
ax = fig.gca(projection='3d')

for i in range(0,n,1):
    test = np.loadtxt('Data03082013/'+files[i*2], dtype=float,delimiter='\t')
    m = test.T[0]
    f = test.T[1]
    a = test.T[4]
    m = m[50:]
    f = f[50:]
    a = a[50:]

    test2 = np.loadtxt('Data03082013/'+files[i*2+1], dtype=float,delimiter='\t')
    m2 = test2.T[0]
    f2 = test2.T[1]
    a2 = test2.T[4]

    magn = np.concatenate((m,m2))
    freq = np.concatenate((f,f2))
    angl = np.concatenate((a,a2))
    temp = np.repeat(Tem[i],len(freq))

    omega = 2*np.pi*freq
    G = cos(angl)/(magn)
    C = -sin(angl)/(magn)

    C0 = (A*eps0)/(d)
    e2 = G/(omega*C0)
    e1 = C/omega*C0
    temp = np.repeat(Tem[i],len(freq))
    temp = temp

    if i == 0:
        dielectric = e2
        frequency = freq
        temperature = temp
    else:
        dielectric = np.vstack((dielectric,e2))
        frequency = np.vstack((freq, frequency))
        temperature = np.vstack((temperature,temp))

Z = dielectric
X = np.log10(frequency)
Y = temperature
surf = ax.plot_surface(Y,X,Z, rstride=1, cstride=1, cmap=cm.rainbow,
        linewidth=0, antialiased=False)
ax.view_init(azim=47)
title('Dielectric Surface (TSiKN65)')
ylabel('Frequency (log)')
xlabel('Temperature C$^\circ$')
ax.set_zlabel('$\epsilon$"')
```

Out[18]: <matplotlib.text.Text at 0xe7c1470>



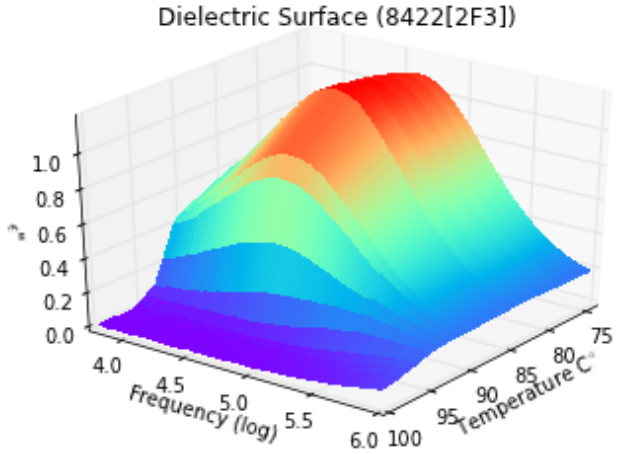
8422[2F3] Surface

In [17]:

```
files = ['test01','test02','test03','test04','test05','test06','test07','test08',
        'test09','test10','test11','test12','test13','test14','test15','test16',
        'test17','test18','test19','test20','test21','test22','test23','test24',
        'test25','test26','test27','test28','test29','test30','test31','test32']

n = len(files)/2
A = .01**2;
eps0 = 8.854187e-12;
d = 5e-6;
e1=[]
e2=[]
Tem = [100.1, 98.9, 97.9, 96.9, 96, 94.6, 93.5, 92.4, 91, 90.1, 84, 82, 80, 77.9,
               75.9, 74, 71.9, 70, 68, 67, 66, 65]
temp= []
fig = plt.figure()
ax = fig.gca(projection='3d')

for i in range(0,n,1):
    test = np.loadtxt('Data04262013/'+files[i], dtype=float,delimiter='\t')
    m = test.T[0]
    f = test.T[1]
    a = test.T[4]
    m = m[5:]
    f = f[5:]
    a = a[5:]

    magn = m
    freq = f
    angl = a
    temp = np.repeat(Tem[i],len(freq))

    omega = 2*np.pi*freq
    G = cos(angl)/(magn)
    C = -sin(angl)/(magn)
```

```
    C0 = (A*eps0)/(d)
    e2 = G/(omega*C0)
    e1 = C/omega*C0
    temp = np.repeat(Tem[i],len(freq))
    temp = temp

    if i == 0:
        dielectric = e2
        frequency = freq
        temperature = temp
    else:
        dielectric = np.vstack((dielectric,e2))
        frequency = np.vstack((freq, frequency))
        temperature = np.vstack((temperature,temp))

Z = dielectric
X = np.log10(frequency)
Y = temperature
surf = ax.plot_surface(Y,X,Z, rstride=1, cstride=1, cmap=cm.rainbow,
        linewidth=0, antialiased=False)
ax.view_init(azim=37)
title('Dielectric Surface (8422[2F3])')
ylabel('Frequency (log)')
xlabel('Temperature C$^\circ$')
ax.set_zlabel('$\epsilon$"')
```

Out[17]: <matplotlib.text.Text at 0x19414f30>



Soft Mode Absorption

From these spectra, one can easily see the Goldstone mode that exists throughout the SmC phase, described earlier. This is heavily influenced by effects from surface anchoring and cannot be easily quantified \cite{blinc}. For this experiment, we restrict ourselves to the analysis of the mode that appears right at the transition temperature. This is known as the soft mode and results from the fluctuations of the tilt angle. A typical De Vries material exhibit large soft-mode absorptions because of the reduced layer shrinkage. In a non-De Vries material, the restoring force of tilt fluctuations should be close to the elastic energy required for layer shrinkage (induced by the tilt) at the transition temperature. This means the switching of the tilt

director will require more energy than a De Vries material, and dielectric absorption will be small in this region. In the De Vries materials, the molecules are already tilted in the SmA phase and can easily align and tilt in the oscillating frequency. In general, the weaker the coupling between the layer spacing and tilt fluctuations, the higher the soft-mode response.

We quantify the soft-mode data by using the Debye Equation for each spectrum. We use IPython notebook and Matplotlib to plot the data and fit our model to each mode in a spectrum. The modes are characterized by their dielectric susceptibility χ, distribution parameter β, and resonant frequency. A least-squares-fit function is used to determine these values.

For the experiment, we are looking to compare this behavior to the mean-field theory for the susceptibility of these materials. One conclusion of the theory is that the susceptibility of order-parameter fluctuations (like the soft mode fluctuations) for a second-order transition point T_C diverges according to:

$$\Delta\epsilon = \chi = \frac{1}{m(T - T_C)}$$

From this equation, we plot the reciprocal of our χ against the $T - T_C$ and extract the constant slope m. This plot is known as a Curie-Weiss Plot. In our experiment, we expect the crystals to show linear behavior, indicating a second-order phase transition. This can be modeled by the Landau expansion of the free energy:

$$g = g_0 + \frac{1}{2}\alpha(T - T_C)\Theta^2 + \frac{1}{2\chi_\infty \epsilon_0} P^2 - C\Theta P - PE,$$

where Θ is the tilt angle, P is the polarization, E is the electric field applied, and g_0 is the nonsingular term corresponding the SmA phase \cite{devries}. A smaller leading Landau coefficient α broadens the temperature region where the mean-field theory is valid \cite{dielectric}. This coefficient measures how easily the tilt angle can fluctuate in these materials from external perturbations (like an oscillating electric field). From our conclusions earlier in this section, a larger soft-mode from small layer shrinkage leads to a smaller restoring force and smaller values of α. In the Landau model, this means the tilt-angle is easily effected by external electric fields.

If we minimize the free energy equation with respect to the polarization, this gives the spontaneous polarization \cite{blinc}
$P_s = \chi_\infty \epsilon_0 C \theta_s$

where χ_∞ is $\epsilon_\infty - 1$ and C is a constant that couples the tilt and the polarization. We obtain this value from optical experiments done by M. Krueger and F. Giesselmann in 2005 \cite{dielectric}. Therefore, we expect the slope of our Curie-Weiss plot to be

$$m_\chi = \frac{\alpha}{\epsilon_0 C^2 \chi_\infty^2}.$$

We solve this equation in terms of α to determine the soft-mode susceptibility Landau coefficient of the liquid crystal

$$\alpha = \frac{m_\chi m_{P\theta}^2}{\epsilon_0}$$

where $C^2 \chi^{2}{\infty} = m^{2}{P\Theta}$. This coefficient quantifies the dielectric susceptibility of the system in the presence of external electric perturbation.

The Code to Analyze the Data

Run the cell to load in two programs which process the HP 4192A LF impedance analyzer data recorded by the Labview Program, "Impendance Analyzer." The files are exported as ".txt" files, and the ".py" programs import the data into the IPython notebook. Using matplotlib and numpy modules, the data is parsed and filtered to create spectras for each temperature run. The second program plots all the data as a 3D surface

Each spectra is a record of the dielectric constant, ϵ'', against the frequency (in Hertz). The surface plot is this value at all temperatures and frequencies.

The impedance analyzer measures the magnitude of the complex impedance and the

Import the single temperature data to fit the curve and initialize

In [3]:
```
files = ['test35','test36']

test = np.loadtxt('Data03082013/'+files[0], dtype=float,delimiter='\t')
#magnitude
m = test.T[0]
#frequency
f = test.T[1]
#angle projected into imaginery/real plane
a = test.T[4]

test2 = np.loadtxt('Data03082013/'+files[1], dtype=float,delimiter='\t')
m2 = test2.T[0]
f2 = test2.T[1]
a2 = test2.T[4]
```

Combine the data into a 3D array

In [4]:
```
magn = np.concatenate((m,m2))
freq = np.concatenate((f,f2))
angl = np.concatenate((a,a2))
```

Conversion to Dielectric

Put the data in terms of the dielectric constant.

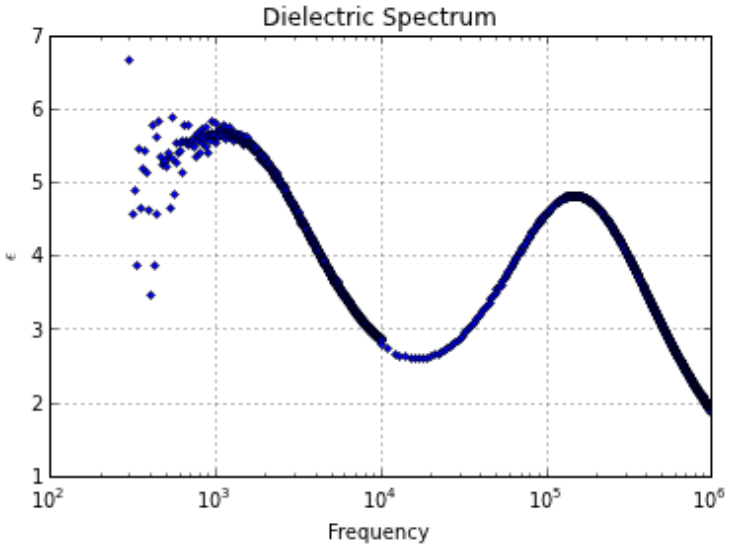
In [4]:
```
A = .01**2;
eps0 = 8.854187e-12;
d = 5e-6;
omega = 2*np.pi*freq;

G = cos(angl)/(magn);
C = -sin(angl)/(magn);

C0 = (A*eps0)/(d);

""" e2 is the epsilon double primed or the imaginery part of the dielectric data
    e1 is the epsilong primed or real part of the dielectric data """

e2 = G/(omega*C0);
e1 = C/(omega*C0);
```

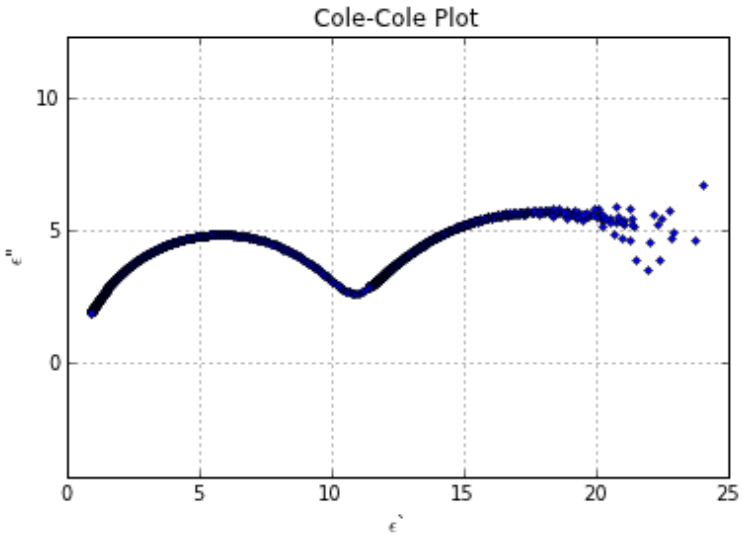
Plot the data using matplotlib

The first plot is the imaginery dielectric spectrum. The second plot is the cole-cole plot of the liquid crystal.

In [5]:
```
plt.figure(1)
semilogx(freq,e2,'.')
xlabel('Frequency')
ylabel('$\epsilon$')
title('Dielectric Spectrum')
grid()

plt.figure(3)
plot(e1,e2,'.')
xlabel('$\epsilon$`')
ylabel('$\epsilon$"')
title('Cole-Cole Plot')
grid()
axis('equal')
```

Out[5]: (0.0, 25.0, 1.0, 7.0)





Use fitted model to determine the dielectric parameters and susceptibility.

Example of one analysis process:

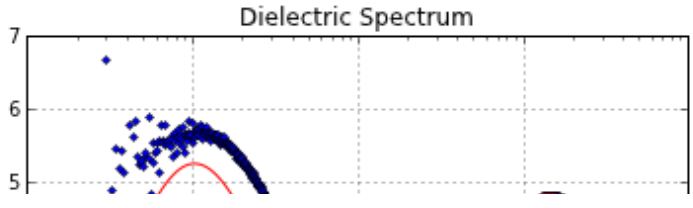
In [6]:
```
""" GUESSED VALUES for the fit of the primary peak in the dielectric data. """
E_infinity = 0;
Delta_E = 12;
Relaxation_frequency = 1000;
Beta = .9;
G_low_frequency = 0;

""" GUESSSED VALUES for the fit of the secondary peak in dielectric data. """
#E_infinity2 = 0;
Delta_E2 = 11;
Relaxation_frequency2 = 150000;
Beta2 = .9;
#G_low_frequency2 = 0;

""" Defining the data as x,y """
x = freq;
y = e2;
```

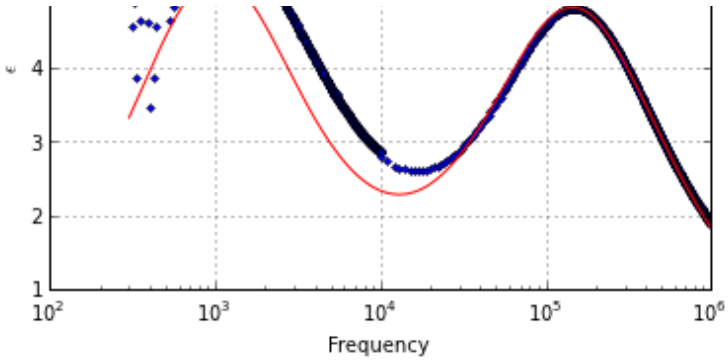
Model function to fit our data

In [7]:
```
def func(x, Einf,delE,relaxFreq,beta,g,delE2,relaxFreq2,beta2):
    z = (Einf + (delE/(1 + (1j*(x/relaxFreq))**(beta))) + g + (delE2/(1 + (1j*
(x/relaxFreq2))**(beta2))))
    return -(z.imag)
```

Display our 'guessed' model to avoid timeout in curve_fit function

In [8]:
```
guess = func(x,E_infinity, Delta_E, Relaxation_frequency, Beta, G_low_frequency,
Delta_E2, Relaxation_frequency2, Beta2)
plt.figure(1)
semilogx(freq,e2,'.')
semilogx(freq,guess,'r')
xlabel('Frequency')
ylabel('$\epsilon$')
title('Dielectric Spectrum')
grid()
```



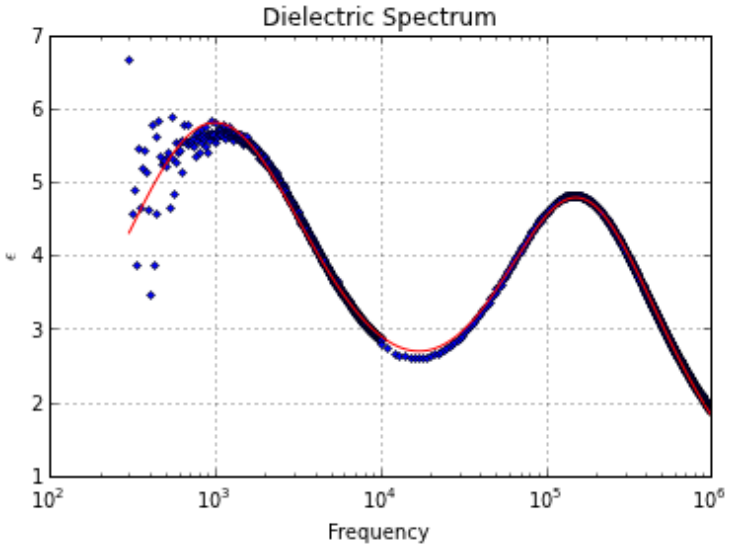


Use scipy's 'curve_fit' function to fit our model to the data.

Print out the values of the fittedmodel and plot the model over the data.

In [9]:
```
popt, pcov = spo.curve_fit(func,x,y,[E_infinity, Delta_E, Relaxation_frequency,
Beta, G_low_frequency, Delta_E2, Relaxation_frequency2, Beta2])
fittedmodel =
func(x,popt[0],popt[1],popt[2],popt[3],popt[4],popt[5],popt[6],popt[7])

def fit_results(param,values):
    for i in range(0,8,1):
        print param[i] + str(values[i])

params = ["\nRESULTS FOR PRIMARY PEAK: \n\nE_infinity: ", "Delta_E: ",
"Relaxation_frequency: ", "Beta: ", "G_low_frequency: ",
          "\nRESULTS FOR SECONDARY PEAK: \n\nE_infinity: ", "Delta_E: ",
"Relaxation_frequency: ", "Beta: ", "G_low_frequency: "];

fit_results(params,popt)

plt.figure(2)
semilogx(freq,e2,'.')
semilogx(freq,fittedmodel,'r')
xlabel('Frequency')
ylabel('$\epsilon$')
title('Dielectric Spectrum')
grid()
```

```
RESULTS FOR PRIMARY PEAK:

E_infinity: 0.0
Delta_E: 16.1055269206
Relaxation_frequency: 961.090887303
Beta: 0.785402730347
G_low_frequency: 0.0

RESULTS FOR SECONDARY PEAK:

E_infinity: 10.2428329901
Delta_E: 162243.20224

Relaxation_frequency: 0.919287227143
```



In [ ]:

Back to top

More info on IPython website (http://ipython.org). The code for this site (https://github.com/ipython/nbviewer) is licensed under BSD (https://github.com/ipython/nbviewer/blob/master/LICENSE.txt). Some icons from Glyphicons Free (http://glyphicons.com), built thanks to Twitter Bootstrap (http://twitter.github.com/bootstrap/)

This web site does not host notebooks, it only renders notebooks available on other websites. Thanks to all our contributors (https://github.com/ipython/nbviewer/contributors).