```python
#!/usr/bin/env python
#  sjk  -  2016.07.06
import sys, re, os #, base64#, math
#from math import all
import numpy as np
import matplotlib.pyplot as plt
from scipy.optimize import curve_fit

##########################################################


kBkcal = 0.0019858775                       #  kcal/mol/K
Faraday = 23.060549                         #  mol/kcal
TempK = 298.15                              #  K
eps0 = 8.854187817e-12                      #  F/m


##########################################################

def linearFunc(x, m, b):
    return m*x + b

def vrhFunc(T, sig0, T0):
    return sig0*(np.exp(-T0*pow(T,-0.25)))

def vrhFunc2d(T, sig0, T0):
    return sig0*(np.exp(-T0*pow(T,-(1.0/3.0))))

def fitFunc(t, a, b, c):
    return a*(np.exp(-b*t)) + c

def fitFunc2(t, a, b):
    return a*(np.exp(-b*t))

def fitFunc3(t, a, b, t0):
    return a*(np.exp(-b*(t-t0)))

def fitFunc4(t, a, b, c, t0):
    return a*(np.exp(-b*(t-t0)))+c


def log10mid(x1,x2):
    """
    value halfway between two values on a log scale
    """
    return 10.0**(math.log10(x1)+(math.log10(x2)-math.log10(x1))/2.0)

def stddev(x):
    """
    Calculate mean and standard deviation of data x[]:
    mean = {\sum_i x_i \over n}
    std = sqrt(\sum_i (x_i - mean)^2 \over n-1)
    """
    from math import sqrt
```

```
    n, mean, sd = len(x), 0.0, 0.0
    for a in x:
        mean = mean + a
    mean = mean / float(n)
    for a in x:
        sd += (a - mean)**2
    sd = sqrt(sd / float(n-1))
    return sd, mean

def getData(fileName,indices,nheads):
    """
    Take a file path and return an array containing
    the columns specified in a list of indices
    """
    inObj = open(fileName,'r')
    inArray = inObj.readlines()
    dataArray = []
    i = 0
    for line in inArray:
        i += 1
        if i > nheads:
            array = line.split()
            if len(array) >= len(indices):
                tmp = []
                for j in indices:
                    tmp.append(array[j])
                dataArray.append(tmp)
    inObj.close()
    return dataArray

def getDropboxRoot():
    # find the path for Dropbox's root watch folder from its sqlite host.db dat
    # Dropbox stores its databases under the currently logged in user's %APPDAT
    # If you have installed multiple instances of dropbox under the same login
    # Dropbox stores its databases under the currently logged in user's %APPDAT
    # usually "C:\Documents and Settings\<login_account>\Application Data"


    try:
        #sConfigFile = os.path.join(os.environ['APPDATA'],'Dropbox','host.db')
        sConfigFile = os.path.join(os.environ['LOCALAPPDATA'],'Dropbox','host.db
    except KeyError:
        sConfigFile = os.path.join(os.environ['HOME'],'.dropbox','host.db')

##    sConfigFile = os.path.join(os.environ['APPDATA'],'Dropbox','host.db')
    #sConfigFile = os.path.join(os.environ['LOCALAPPDATA'],'Google','Drive','sy

    # return null string if can't find or work database file.
    if not os.path.exists(sConfigFile):
        return None

    # Dropbox Watch Folder Location is base64 encoded as the last line of the h
    with open(sConfigFile) as dbxfile:
        for sLine in dbxfile:
```

```
            pass

    # decode last line, path to dropbox watch folder with no trailing slash.
    return base64.b64decode(sLine)

def getLine(arr,wid):
    """ Make a line ready for writing
    arr is an array of column data
    wid is the column width"""
    line = ''
    for dat in arr:
        if isinstance(dat, float):
            st = '%.8g' % dat
        elif isinstance(dat, int):
            st = '%d' % dat
        else:
            st = dat
        line += st.rjust(wid)
    line += '\n'
    return line

def flatten(lst):
    return sum( ([x] if not isinstance(x, list) else flatten(x) for x in lst)

def minmax(lst):
    listmin = lst[0]
    listmax = lst[0]
    for num in lst:
        listmin = min(num,listmin)
        listmax = max(num,listmax)
    return listmin, listmax

def getAxisRange(lst,buff):
    lmin, lmax = minmax(lst)
    axmin = lmin-(lmax-lmin)*buff
    axmax = lmax+(lmax-lmin)*buff
    return axmin, axmax

def seq(start, stop, step=1):
    n = int(round((stop - start)/float(step)))
    if n > 1:
        return([start + step*i for i in range(n+1)])
    else:
        return([])

# flattens a list eg. flatten(1, 2, ['b','a','c']) = [1, 2, 'a', 'b', 'c']
def flatten(*args):
    for x in args:
        if hasattr(x, '__iter__'):
            for y in flatten(*x):
                yield y
        else:
            yield x
```

```
def isIVFile(df):
    if len(df.split('-')) == 3 and df.split('-')[-1] != 'TPt.dat' and df.split
        return True
    else:
        return False

def isIVtFile(df):
    if len(df.split('-')) == 7 and df.split('-')[-1] == 'IVt.dat':
        return True
    else:
        return False

def getIVtFileInfo(df):
    if isIVtFile(df):
        samp = df.split('-')[0]
        meas = int(df.split('-')[1][1:])
        temp = df.split('-')[2]
        dev = df.split('-')[3]
        it = df.split('-')[4]
        volt = int(df.split('-')[5][1:])
        dfType = df.split('-')[-1].split('.')[0]
        return samp, meas, temp, dev, it, volt, dfType
    else:
        return None

def getIVFileInfo(df):
    if isIVFile(df):
        samp = df.split('-')[0]
        temp = df.split('-')[1]
        it = df.split('-')[-1].split('.')[0]
        return samp, temp, it
    else:
        return None

def complex_impedance(omega,Cp,G):
    D = G/(omega*Cp)
    Rp = 1/G
    Z1 = G/(np.power(G,2) + np.power(omega*Cp,2)) # real part of impedance
    Z2 = -1.0*(omega*Cp)/(np.power(G,2) + np.power(omega*Cp,2))
    return Z1, Z2

##########################################################

ColWidth = 20

#print
print ('---- Box ----')
boxpath = os.path.join('C:\\','Users','admin','box')

print ('Dropbox path:    ' + boxpath)
print (os.path.exists(boxpath))

rootPath = os.path.join(boxpath,'sigma-data-share')
#print
```

```
print (os.path.exists(rootPath))

dataPath = os.path.join(rootPath,'test','2019-11-01-10MOhm-test-02')
#dataPath = os.path.join(rootPath,'Cu_L9','2019-11-01-Cu_L9-RT-003')
#dataPath = os.path.join(rootPath,'H_02_Spiro_Ref','2019-11-05-H_02_Spiro_Ref-a
#dataPath = os.path.join(rootPath,'H_02_Spiro_Ref','2019-11-05-H_02_Spiro_Ref-a
#dataPath = os.path.join(rootPath,'H_02_Spiro_Ref','2019-11-05-H_02_Spiro_Ref-R
#dataPath = os.path.join(rootPath,'H_02_Spiro_Ref','2019-11-05-H_02_Spiro_Ref-T
##dataPath = os.path.join(rootPath,'H_02_Spiro_Ref','2019-11-05-H_02_Spiro_Ref-
#dataPath = os.path.join(rootPath,'H_02_Spiro_Ref','2019-11-05-H_02_Spiro_Ref-T
#dataPath = os.path.join(rootPath,'H_03_Spiro_undoped_Ref','2019-11-08-H_03_Spi
##dataPath = os.path.join(rootPath,'H_01_ZnO','2019-11-09-H_01_ZnO-ambient-001'
###dataPath = os.path.join(rootPath,'H_01_ZnO','2019-11-09-H_01_ZnO-ambient-dar
##dataPath = os.path.join(rootPath,'H_01_ZnO','2019-11-09-H_01_ZnO-TempScan-003
#dataPath = os.path.join(rootPath,'H_01_ZnO','2019-11-11-H_01_ZnO-TempScan-004'
#dataPath = os.path.join(rootPath,'H_15_Cu_Tetra_O','2019-11-09-H_15_Cu_Tetra_O
##dataPath = os.path.join(rootPath,'H_14_Cu_Poly_PhI','2019-11-14-H_14_Cu_Poly_
##dataPath = os.path.join(rootPath,'H_26_Cu_Poly_DII','2019-11-14-H_26_Cu_Poly_
#dataPath = os.path.join(rootPath,'H_24_Cu_Poly_DIBr','2019-11-16-H_24_Cu_Poly_
#dataPath = os.path.join(rootPath,'H_24_Cu_Poly_DIBr','2019-11-16-H_24_Cu_Poly_
#dataPath = os.path.join(rootPath,'H_28_Cu_Poly_PhBr','2019-11-17-H_28_Cu_Poly_
#dataPath = os.path.join(rootPath,'H_38_Cu_tmby_Cu12LitBP','2019-11-22-H_38_Cu_
#dataPath = os.path.join(rootPath,'H_38_Cu_tmby_Cu12LitBP','2019-11-22-H_38_Cu_
#dataPath = os.path.join(rootPath,'H_38b_Cu_tmby_Cu12LitBP','2019-11-25-H_38b_C
#dataPath = os.path.join(rootPath,'H_37_Cu_Tetra_C_Cu12LitBP','2019-11-30-H_37_
##dataPath = os.path.join(rootPath,'H_37_Cu_Tetra_C_Cu12LitBP','2019-11-30-H_37

################################################################################

dataPath = os.path.join(rootPath,'H_36_Cu_Tetra_O_Cu12LitBP','2019-11-26-H_36_Cu
dataPath = os.path.join(rootPath,'H_35_Cu_Tetra_O_Cu12','2019-12-04-H_35_Cu_Tetr
dataPath = os.path.join(rootPath,'H_40b_Cu_BTT','2019-12-06-H_40b_Cu_BTT-TempSca
dataPath = os.path.join(rootPath,'H_37_Cu_Tetra_C_Cu12LitBP','2019-11-25-H_37_Cu
dataPath = os.path.join(rootPath,'H_37_Cu_Tetra_C_Cu12LitBP','2019-11-30-H_37_Cu

################################################################################

#dataPath = os.path.join(rootPath,'H_40b_Cu_BTT','2019-12-12-H_40b_Cu_BTT-TempS
#dataPath = os.path.join(rootPath,'H_40b_Cu_BTT','2019-12-12-H_40b_Cu_BTT-295K-
#dataPath = os.path.join(rootPath,'H_41b_Cu_BHT','2019-11-25-H_41b_Cu_BHT-ambie
#dataPath = os.path.join(rootPath,'H_41b_Cu_BHT','2019-12-13-H_41b_Cu_BHT-ambie
#dataPath = os.path.join(rootPath,'H_41b_Cu_BHT','2019-12-13-H_41b_Cu_BHT-TempS
#dataPath = os.path.join(rootPath,'H_38_Cu_tmby_Cu12LitBP','2019-12-13-H_38_Cu_
#dataPath = os.path.join(rootPath,'H_38b_Cu_tmby_Cu12LitBP','2019-12-16-H_38b_C
#dataPath = os.path.join(rootPath,'H_38_Cu_tmby_Cu12LitBP','2019-12-18-H_38_Cu_
#dataPath = os.path.join(rootPath,'H_38_Cu_tmby_Cu12LitBP','2019-12-18-H_38_Cu_
#dataPath = os.path.join(rootPath,'H_38_Cu_tmby_Cu12LitBP','2019-12-19-H_38_Cu_

#dataPath = os.path.join(rootPath,'JF_ZnOink', '2020-03-05-JF_ZnOink_B2-anneale

################################################################################
#
#dataPath = os.path.join(rootPath,'test', 'Eric-1K')
#dataPath = os.path.join(rootPath,'test', 'Eric-1K100pF')
```

```
#dataPath = os.path.join(rootPath,'test', 'Eric-10M100pF')
#dataPath = os.path.join(rootPath,'test', 'Eric-10M')
#
#dataPath = os.path.join(rootPath,'JS_04_CuI_run1', '2021-07-02_JS04_CuI_ambien
#dataPath = os.path.join(rootPath,'JS_04_CuI_run1', '2021-07-02_JS04_CuI_ambien
#dataPath = os.path.join(rootPath,'JS_04_CuI_run1', '2021-07-02_JS04_CuI_pumpin
#dataPath = os.path.join(rootPath,'JS_04_CuI_run1', '2021-07-02_JS04_CuI_pumpin
#dataPath = os.path.join(rootPath,'JS_04_CuI_run1', '2021-07-02_JS04_CuI_pumpin
#dataPath = os.path.join(rootPath,'JS_04_CuI_run1', '2021-07-02_JS04_CuI_pumpin
#dataPath = os.path.join(rootPath,'JS_04_CuI_run1', '2021-07-02_JS04_CuI_tempsc
#
#dataPath = os.path.join(rootPath,'JS_07_CuI_run1', '2021-07-16_JS07_CuI_ambien
#dataPath = os.path.join(rootPath,'JS_07_CuI_run1', '2021-07-16_JS07_CuI_pumpin
#dataPath = os.path.join(rootPath,'JS_07_CuI_run1', '2021-07-18_JS07_CuI_vacuum
#dataPath = os.path.join(rootPath,'JS_07_CuI_run1', '2021-07-18_JS07_CuI_tempsc
#dataPath = os.path.join(rootPath,'JS_07_CuI_run1', '2021-07-19_JS07_CuI_tempsc
#dataPath = os.path.join(rootPath,'JS_07_CuI_run1', '2021-07-21_JS07_CuI_tempsc
#dataPath = os.path.join(rootPath,'JS_07_CuI_run1', '2021-07-26_JS07_CuI_tempsc
#dataPath = os.path.join(rootPath,'JS_07_CuI_run1', '2021-07-30_JS07_CuI_tempsc
dataPath = os.path.join(rootPath,'JS_07_CuI_run1', '2021-08-02_JS07_CuI_tempscan
dataPath = os.path.join(rootPath,'JS_07_CuI_run1', '2021-08-04_JS07_CuI_tempscan
dataPath = os.path.join(rootPath,'JS_07_CuI_run1', '2021-08-06_JS07_CuI_tempscan
dataPath = os.path.join(rootPath,'JS_07_CuI_run1', '2021-08-17_JS07_CuI_tempscan


dataPath = os.path.join(rootPath,'JS_X6_AuNP_CuI', '2021-08-19_JSX6_AuNP_CuI_amb
dataPath = os.path.join(rootPath,'JS_X6_AuNP_CuI', '2021-08-19_JSX6_AuNP_CuI_amb
dataPath = os.path.join(rootPath,'JS_X6_AuNP_CuI', '2021-08-19_JSX6_AuNP_CuI_pum
dataPath = os.path.join(rootPath,'JS_X6_AuNP_CuI','2021-08-19_JSX6_AuNP_CuI_temp
dataPath = os.path.join(rootPath,'JS_X6_AuNP_CuI','2021-08-20_JSX6_AuNP_CuI_temp
dataPath = os.path.join(rootPath,'JS_X6_AuNP_CuI','2021-08-23_JSX6_AuNP_CuI_temp


dataPath = os.path.join(rootPath,'JS_X2_AuNP_CuI','2021-08-25_JSX2_AuNP_CuI_amb
dataPath = os.path.join(rootPath,'JS_X2_AuNP_CuI','2021-08-25_JSX2_AuNP_CuI_amb
dataPath = os.path.join(rootPath,'JS_X2_AuNP_CuI','2021-08-25_JSX2_AuNP_CuI_pump
dataPath = os.path.join(rootPath,'JS_X2_AuNP_CuI','2021-08-25_JSX2_AuNP_CuI_temp
dataPath = os.path.join(rootPath,'JS_X2_AuNP_CuI','2021-08-27_JSX2_AuNP_CuI_temp
dataPath = os.path.join(rootPath,'JS_X2_AuNP_CuI','2021-08-28_JSX2_AuNP_CuI_temp
dataPath = os.path.join(rootPath,'JS_X2_AuNP_CuI','2021-09-08_JSX2_AuNP_CuI_temp




dataPath = os.path.join(rootPath,'EC3_PTAA','2023-03-06_EC3_PTAA_Au_ambient_001
dataPath = os.path.join(rootPath,'EC3_PTAA','2023-03-06_EC3_PTAA_Au_ambientdark_
dataPath = os.path.join(rootPath,'EC3_PTAA','2023-03-06_EC3_PTAA_Au_ambientdark_
dataPath = os.path.join(rootPath,'EC3_PTAA','2023-03-06_EC3_PTAA_Au_pumpdown_004
dataPath = os.path.join(rootPath,'EC3_PTAA','2023-03-06_EC3_PTAA_Au_pumpdown_005
dataPath = os.path.join(rootPath,'EC3_PTAA','2023-03-06_EC3_PTAA_Au_pumpdown_006
dataPath = os.path.join(rootPath,'EC3_PTAA','2023-03-06_EC3_PTAA_Au_tempscan_007
dataPath = os.path.join(rootPath,'EC3_PTAA','2023-03-16_EC3_PTAA_Au_tempscan_008
dataPath = os.path.join(rootPath,'EC3_PTAA','2023-03-31_EC3_PTAA_Au_tempscan_009
dataPath = os.path.join(rootPath,'EC3_PTAA','2023-04-17_EC3_PTAA_Au_tempscan_010
```

```
dataPath = os.path.join(rootPath,'EC2_PTAA','2023-04-24_EC2_PTAA_Au_ambient_001
dataPath = os.path.join(rootPath,'EC2_PTAA','2023-04-24_EC2_PTAA_Au_pumpdown_00
dataPath = os.path.join(rootPath,'EC2_PTAA','2023-04-24_EC2_PTAA_Au_pumpdown_00
dataPath = os.path.join(rootPath,'EC2_PTAA','2023-04-24_EC2_PTAA_Au_tempscan_00
dataPath = os.path.join(rootPath,'EC2_PTAA','2023-04-26_EC2_PTAA_Au_tempscan_00
dataPath = os.path.join(rootPath,'EC2_PTAA','2023-04-29_EC2_PTAA_Au_tempscan_00
dataPath = os.path.join(rootPath,'EC2_PTAA','2023-05-01_EC2_PTAA_Au_tempscan_00
#dataPath = os.path.join(rootPath,'EC2_PTAA','2023-05-03_EC2_PTAA_Au_tempscan_0
dataPath = os.path.join(rootPath,'EC2_PTAA','2023-05-09_EC2_PTAA_Au_tempscan_00


dataPath = os.path.join(rootPath,'DS6_CQD_Perov','2023-05-18_DS6_CQD_Perov_pump
dataPath = os.path.join(rootPath,'DS6_CQD_Perov','2023-05-18_DS6_CQD_Perov_pump
dataPath = os.path.join(rootPath,'DS6_CQD_Perov','2023-05-19_DS6_CQD_Perov_ambi
dataPath = os.path.join(rootPath,'DS6_CQD_Perov','fake')

dataPath = os.path.join(rootPath,'DS7_CQD_Perov','2023-05-19_DS7_CQD_Perov_pump
dataPath = os.path.join(rootPath,'DS7_CQD_Perov','2023-05-19_DS7_CQD_Perov_pump
dataPath = os.path.join(rootPath,'DS7_CQD_Perov','2023-05-19_DS7_CQD_Perov_temp


dataPath = os.path.join(rootPath,'DS3_CuSCN','2023-05-25_DS3_CuSCN_30_ambient_0
dataPath = os.path.join(rootPath,'DS3_CuSCN','2023-05-25_DS3_CuSCN_30_pumpingdo
dataPath = os.path.join(rootPath,'DS3_CuSCN','2023-05-25_DS3_CuSCN_30_pumpingdo
dataPath = os.path.join(rootPath,'DS3_CuSCN','2023-05-25_DS3_CuSCN_30_pumpingdo
dataPath = os.path.join(rootPath,'DS3_CuSCN','2023-05-25_DS3_CuSCN_30_tempscan_
dataPath = os.path.join(rootPath,'DS3_CuSCN','2023-05-26_DS3_CuSCN_30_tempscan_
dataPath = os.path.join(rootPath,'DS1_CuSCN','2023-05-30_DS1_CuSCN_30mix_ambien
dataPath = os.path.join(rootPath,'DS1_CuSCN','2023-05-30_DS1_CuSCN_30mix_pumpin
dataPath = os.path.join(rootPath,'DS1_CuSCN','2023-05-30_DS1_CuSCN_30mix_tempsc

print (dataPath)

imgPath = os.path.join(dataPath,'plots')
dataSets = []

if not os.path.isdir(imgPath):
    os.makedirs(imgPath)

dataFiles = [ f for f in os.listdir(dataPath) if os.path.isfile(os.path.join(da

## fingers generic
width = 37.5869e-3  # (m)  ?
length = 1.0e-3     # (m)
#thick = 500.0e-9     # (m)  ?
thick = 10.0e-9     # (m)  ?

#### Interdigitated B2 generic
#width = 1.785040 *0.9*0.25*0.1# (m)  ? #fingers broken #coverage #actually sol
#length =  20e-6    # (m)
#thick = 500.0e-9     *0.03# (m)  ?


###
```

```
###  set up master list of data
###
samples = set()
temperatures = set()
iterations = set()
for df in dataFiles:
    if isIVFile(df):
        samp, temp, it = getIVFileInfo(df)
        samples.add(samp)
        temperatures.add(temp)
        iterations.add(it)

#print dataFiles


samps = sorted(list(samples))
temps = sorted(list(temperatures))
iters = sorted(list(iterations))
nits = len(iters)
ntemps = len(temps)
sample = samps[0]
if len(samps) > 1:
    print ("  More than one sample name.  Only looking at sample:  "+sample)
else:
    print ("  Sample name:  "+sample)

line = "  Iterations:"
for it in iters:
    line += "  "+it
print (line)

print ("  Temperatures:  %d" %ntemps)
#print


#print nothing

###
###  create master list and add I, V, T, P, and t data
###

#fnadd = ''
#masterList = []
#maxit = 1
#for temp in temps:
#    for it in iters:
#        milist = [sample,it,temp,[]]
#        masterList.append(milist)
#        maxit=max(int(it),maxit)

#absVoltMin = 0.0
#absVoltMax = 10.19
#minCurrent = -1.0e20
```

```
#plt.clf()
plt.close('all')
fig, ax1 = plt.subplots()
fsize = 14
ptsize = 9
lwidth = 1.5

DataSet1 = []
DataSet2 = []
DataSet3 = []
for temp in temps:
    TKData = []
    PTorrData = []
    ROhmsData = []
    OffsetData = []
    sigmaData = []
    #plt.cla()
    #ax1.set_xlabel(xlabl, fontsize = fsize)
    #ax1.set_ylabel(ylabl, fontsize = fsize)

    writeSigmaTimeFiles = True
    if writeSigmaTimeFiles:
        df = sample+'-'+temp+'-cond-time.dat'
        dFile = os.path.join(imgPath,df)
        outFile = open(dFile,'w')
        head = ['time(s)', 'R(Ohms)', 'cond(S/cm)', 'T(K)', 'P(Torr)']
        line = getLine(head,15)
        outFile.write(line)

    sigtimeArray = []

    for it in iters:
        VVoltsData = []
        IAmpsData = []

        starr = []

        df = sample+'-'+temp+'-'+it+'.dat'

        print (df)
        dFile = os.path.join(dataPath,df)

        if os.path.isfile(dFile):
            dataSet = getData(dFile,[0,1,2,3,4,5],0)

            TKData.append(float(dataSet[0][3]))
            PTorrData.append(float(dataSet[0][4]))
            starr.append(float(dataSet[0][5]))
            starr.append(float(dataSet[0][3]))
            starr.append(float(dataSet[0][4]))

            for dat in dataSet:
                vvv = float(dat[0])
```

```
    if vvv>=-1000000000.0:
        VVoltsData.append(vvv)
        IAmpsData.append(float(dat[1]))

xxx = np.array(VVoltsData)
yyy = np.array(IAmpsData)
#print '--'
#print dFile
fitParams, fitCovariances = curve_fit(linearFunc, xxx, yyy)

Res = 1.0/fitParams[0]
sigma = fitParams[0]*length/(thick*width*100.0)  # (S/cm)
#sigmaSD = perr[0]*length/(thick*width*100.0)  # (S/cm)

if it==iters[0]:
    maxxx = max(abs(xxx))
    mayyy = max(abs(yyy))

    prefixx = ''
    xfac = 1.0
    prefixy = ''
    yfac = 1.0

    if maxxx < 1.0e-3:
        prefixx = '$\mu$'
        xfac = 1.0e6
    elif maxxx < 1.0:
        prefixx = 'm'
        xfac = 1.0e3

    if mayyy < 1.0e-12:
        prefixy = 'f'
        yfac = 1.0e15
    elif mayyy < 1.0e-9:
        prefixy = 'p'
        yfac = 1.0e12
    elif mayyy < 1.0e-6:
        prefixy = 'n'
        yfac = 1.0e9
    elif mayyy < 1.0e-3:
        prefixy = '$\mu$'
        yfac = 1.0e6
    elif mayyy < 1.0:
        prefixy = 'm'
        yfac = 1.0e3


    plt.cla()
    xlabl = 'voltage ('+prefixx+'V)'
    ylabl = 'current ('+prefixy+'A)'
    ax1.set_xlabel(xlabl, fontsize = fsize)
    ax1.set_ylabel(ylabl, fontsize = fsize)
    #ax1.set_xscale('log')
    #ax1.set_yscale('log')
```

```
        ax1.plot(xxx*xfac, yyy*yfac, 'o')
        for j in range(0,len(xxx)):
            ax1.text(xxx[j]*xfac, yyy[j]*yfac, str(int(it)), color="black",
        #perr = np.sqrt(np.diag(fitCovariances))
        #print perr
        #ResSD = perr[0]/fitParams[0]/fitParams[0]

        ROhmsData.append(Res)
        OffsetData.append(fitParams[1])
        sigmaData.append(sigma)

        starr.insert(1,Res)
        starr.insert(2,sigma)
        sigtimeArray.append(starr)



if writeSigmaTimeFiles:
    #print sigtimeArray
    for dat in sigtimeArray:
        line = getLine(dat,15)
        outFile.write(line)
    outFile.close()

if len(TKData) > 1:
    TKSD, TKAve = stddev(TKData)
    TKSD_inv, TKAve_inv = stddev([1000 / T for T in TKData])
    PTorrSD, PTorrAve = stddev(PTorrData)
    ROhmsSD, ROhmsAve = stddev(ROhmsData)
    OffsetSD, OffsetAve = stddev(OffsetData)
    sigmaSD, sigmaAve = stddev(sigmaData)
else:
    TKSD = 0.0
    TKAve = TKData[0]
    PTorrSD = 0.0
    PTorrAve = PTorrData[0]
    ROhmsSD = 0.0
    ROhmsAve = ROhmsData[0]
    OffsetSD = 0.0
    OffsetAve = OffsetData[0]
    sigmaSD = 0.0
    sigmaAve = sigmaData[0]

title = 'T = '
title += '%.4f K' % TKAve
title += '      b = %.4g A\n' % OffsetAve
title += 'R = %.6g' % ROhmsAve
title += '      Rsd = %.6g' % ROhmsSD
ax1.set_title(title)

ax1.plot(xxx*xfac, yfac*linearFunc(xxx, 1/ROhmsAve, OffsetAve), 'g-')  ####
```

```
    makePlots = True
    if makePlots:

        pngName = sample+'-'+temp+'.png'
        pngFile = os.path.join(imgPath,pngName)
        plt.savefig(pngFile, bbox_inches=0, dpi=300)



    DataSet1.append([TKAve, TKSD, ROhmsAve, ROhmsSD, PTorrAve, PTorrSD]) #fix S
    DataSet2.append([TKAve_inv, TKSD_inv, sigmaAve, sigmaSD]) #fix SDs!!!!
    DataSet3.append([TKAve, TKSD, sigmaAve, sigmaSD]) #fix SDs!!!!



for temp in temps:
    plt.clf()
    fig,ax1 = plt.subplots()
    left, bottom = 0.1, 0.98


    for it in iters:
        writeImpedanceFiles = True
        if writeImpedanceFiles:
            df = sample+'-'+temp+'-'+it+'-IS-Zdata.dat'
            dFile = os.path.join(imgPath,df)
            outFile = open(dFile,'w')
            head = ['freq(Hz)', 'ReZ(Ohms)', 'ImZ(Ohms)']
            line = getLine(head,15)
            outFile.write(line)

        df = sample+'-'+temp+'-'+it+'-IS.dat'

        print (df)
        dFile = os.path.join(dataPath,df)

        if os.path.isfile(dFile):
            #print 'yes'
            dataSet = getData(dFile,[0,1,2,3,4],0)

            vs = set()
            fs = set()
            for d in dataSet:
                voltage=float(d[1])
                frequency=float(d[2])
                vs.add(voltage)
                fs.add(frequency)

            voltages = sorted(list(vs))
            frequencies = sorted(list(fs))

            freqs=[]
            rezs=[]
```

```
            imzs=[]
            for dat in dataSet:
                vv = float(dat[1])
                if vv==0.0:
                    freq = float(dat[2])
                    omega = 2.0*np.pi*freq
                    cap = float(dat[4])
                    cond = float(dat[3])
                    rez, imz = complex_impedance(omega,cap,cond)
                    #impData.append([])
                    freqs.append(freq)
                    rezs.append(rez)
                    imzs.append(imz)
                    if writeImpedanceFiles:
                        line = getLine([freq,rez,imz],15)
                        outFile.write(line)

            z1arr = np.array(rezs)
            z2arr = np.array(imzs)

            ax1.plot(z1arr,-1*z2arr,'.')

    makePlots = True

    if makePlots:
        ax1.set_xlabel("Z'")
        ax1.set_ylabel("-Z''")

        plt.gca().xaxis.get_major_formatter().set_powerlimits((0,0))
        plt.gca().yaxis.get_major_formatter().set_powerlimits((0,0))

        limmin = max(ax1.get_xlim()[0],ax1.get_ylim()[0])
        limmax = max(ax1.get_xlim()[1],ax1.get_ylim()[1])
        ax1.set_xlim([limmin,limmax])
        ax1.set_ylim([limmin,limmax])
        ax1.set_aspect('equal')

        pngName = sample+'-'+temp+'-IS-Nyquist.png'
        pngFile = os.path.join(imgPath,pngName)

        plt.tight_layout()
        plt.savefig(pngFile, bbox_inches=0, dpi=300)




##############################
#  Output data files:
##############################

df = sample+'-TRP.dat'
dFile = os.path.join(imgPath,df)
```

```
outFile = open(dFile,'w')
head = ['aveTemp(K)', 'sdTemp(K)', 'aveR(Ohms)', 'sdR(Ohms)', 'avePress(Torr)',
line = getLine(head,15)
outFile.write(line)
for dat in DataSet1:
    line = getLine(dat,15)
    outFile.write(line)
outFile.close()

df = sample+'-cond-vs-invT.dat'
dFile = os.path.join(imgPath,df)
outFile = open(dFile,'w')
head = ['ave1000/T(K-1)', 'sd1000/T(K-1)', 'aveSigma(S/cm)', 'sdSigma(S/cm)']
line = getLine(head,15)
outFile.write(line)
for dat in DataSet2:
    line = getLine(dat,15)
    outFile.write(line)
outFile.close()

df = sample+'-cond-vs-T.dat'
dFile = os.path.join(imgPath,df)
outFile = open(dFile,'w')
head = ['aveTemp(K)', 'sdTemp(K)', 'aveSigma(S/cm)', 'sdSigma(S/cm)']
head.append('width(m)')
head.append('length(m)')
head.append('thick(m)')
line = getLine(head,15)
outFile.write(line)
DataSet3[0].append(width)
DataSet3[0].append(length)
DataSet3[0].append(thick)
for dat in DataSet3:
    line = getLine(dat,15)
    outFile.write(line)
outFile.close()


##############################
#  Plot cond vs. 1000/T:
##############################

pngName = samp+'-cond-vs-invT.png'
pngFile = os.path.join(imgPath,pngName)
##plt.clf()
##fig, ax1 = plt.subplots()
##fsize = 18
##ptsize = 8
##lwidth = 1.5
plt.cla()
for item in ([ax1.title, ax1.xaxis.label, ax1.yaxis.label] +
             ax1.get_xticklabels() + ax1.get_yticklabels()):
    item.set_fontsize(fsize)
ax1.set_xlabel('$\mathregular{1000/T (K^{-1})}$')
```

```
#ax1.set_xlim([3,10])
#ax1.set_ylim([-1e-14,1e-14])
ax1.set_ylabel('conductivity ($\Omega\mathregular{^{-1}cm^{-1}}$)')
ax1.set_xscale('linear')
ax1.set_yscale('log')

invTAve = []
invTSD = []
sigmaAve = []
sigmaSD = []
for i in range(0,len(DataSet2)):
    print (DataSet2[i])
    invTAve.append(DataSet2[i][0])
    invTSD.append(DataSet2[i][1])
    sigmaAve.append(DataSet2[i][2])
    sigmaSD.append(DataSet2[i][3])

#print invTAve
#print nothing

#xxx = np.array(invTAve[2:-2])
#xxxerr = np.array(invTSD[2:-2])
#yyy = np.array(sigmaAve[2:-2])
#yyyerr = np.array(sigmaSD[2:-2])

xxx = np.array(invTAve)
xxxerr = np.array(invTSD)
yyy = np.array(sigmaAve)
yyyerr = np.array(sigmaSD)
        #ax1.plot(xxx, yyy1, '-', linewidth=lwidth)
ax1.plot(xxx, yyy, 'o', markersize=ptsize, fillstyle='none', color='#cb4154')

iii=0
for x, y in zip(xxx, yyy):
    iii+=1
    ax1.text(x, y, str(iii), color="black", fontsize=5, horizontalalignment='cer

#ax1.errorbar(xxx, yyy, xerr=xxxerr, yerr=yyyerr, fmt='o')
#plt.tight_layout()

#####################################
ax2 = ax1.twiny()
ax2.plot(xxx, yyy, 'o', markersize=ptsize, alpha=0)

xmin, xmax = ax1.get_xlim()

print (xmin, xmax)
tixlabel = []
tixloc = []
for tk in [300, 250, 200, 150, 100, 50]:
    ntk = 1000.0/tk
    if ntk>xmin and ntk<xmax:
        tixlabel.append(tk)
        tixloc.append(ntk)
```

```
new_tick_labels = np.array(tixlabel)
new_tick_locations = np.array(tixloc)

print (new_tick_labels)
print (new_tick_locations)

ax2.set_xlim(ax1.get_xlim())
ax2.set_xticks(new_tick_locations)
ax2.set_xticklabels(new_tick_labels)
ax2.set_xlabel(r"$T$ (K)")

for item in ([ax2.title, ax2.xaxis.label, ax2.yaxis.label] +
             ax2.get_xticklabels() + ax2.get_yticklabels()):
    item.set_fontsize(fsize)
####################################

plt.tight_layout()
plt.savefig(pngFile, bbox_inches=0, dpi=600)

##############################
##  Plot cond vs. T-1/4:
##############################
#
#pngName = samp+'-cond-vrh.png'
#pngFile = os.path.join(imgPath,pngName)
#
#plt.cla()
#for item in ([ax1.title, ax1.xaxis.label, ax1.yaxis.label] +
#             ax1.get_xticklabels() + ax1.get_yticklabels()):
#    item.set_fontsize(fsize)
#
#
##tempmax = 276.0
##tempmin = 200.0
##tempmax = 2700.0
#
#temps = []
#conds = []
#vrhx = []
#for dat in DataSet3:
#    temp = float(dat[0])
##    if temp < tempmax and temp > tempmin:
#    temps.append(temp)
#    vrhtemp = pow(temp,-0.25)
#    vrhx.append(vrhtemp)
#    conds.append(float(dat[2]))
#
#xxx = np.array(temps)
#yyy = np.array(conds)
#vrh = np.array(vrhx)
#
##fitParams, fitCovariances = curve_fit(vrhFunc, xxx, yyy)
##print fitParams
##fitParams2d, fitCovariances2d = curve_fit(vrhFunc2d, xxx, yyy)
```

```
##print fitParams2d
#
#plt.cla()
#ax1.set_xlabel('$T\mathregular{^{-1/4} (K^{-1/4})}$')
##ax1.set_ylim([-1e-14,1e-14])
#ax1.set_ylabel('$\sigma$ ($\Omega\mathregular{^{-1}cm^{-1}}$)')
##ax1.set_xscale('log')
#ax1.set_yscale('log')
#
#ax1.plot(vrh, yyy, 'o')
#
#title = '     $\sigma_{0}$  = '
##title += '%.4g ' % fitParams[0]
##title += '      $T_{0}$ = %.6g' % fitParams[1]
##ax1.set_title(title)
#
##ax1.plot(vrh, vrhFunc(xxx, fitParams[0], fitParams[1]), 'g--')
##ax1.plot(vrh, vrhFunc2d(xxx, fitParams2d[0], fitParams2d[1]), 'r-')
#
#sig0 = 2.80432002e+12
#T0 = -1.65440244e+02
##ax1.plot(vrh, vrhFunc(xxx, sig0, T0), 'g--')
#sig0 = 3.e+14
#T0 = 1.84e+02
##ax1.plot(vrh, vrhFunc(xxx, sig0, T0), 'r-')
#
#ax2 = ax1.twiny()
#ax2.plot(vrh, yyy, alpha=0)
#
#xmin, xmax = ax1.get_xlim()
#
#print xmin, xmax
#tixlabel = []
#tixloc = []
#for tk in [300, 250, 200, 150, 100, 50]:
#    ntk = pow(tk,-.25)
#    if ntk>xmin and ntk<xmax:
#        tixlabel.append(tk)
#        tixloc.append(ntk)
#new_tick_labels = np.array(tixlabel)
#new_tick_locations = np.array(tixloc)
#
#print new_tick_labels
#print new_tick_locations
#
#ax2.set_xlim(ax1.get_xlim())
#ax2.set_xticks(new_tick_locations)
#ax2.set_xticklabels(new_tick_labels)
#ax2.set_xlabel(r"$T$ (K)")
#
#
#plt.savefig(pngFile, bbox_inches=0, dpi=600)
#
```