Readme for portfolio code

T. Zack Crawford

12 Feb 2021

Purpose

To showcase some of my computer science skills, here is some code as a “portfolio” of projects I have completed in my past. The projects below illustrate such skills in various areas. For each project, I will explain a bit on the relevance, how to use the code, and think about any results.

Admittance spectroscopy project

Skills showcased:

- Extracting meaningful insights from raw data
- Visualization of such data to emphasize an idea
- Knowledge of some statistics, particularly error analysis
- General python knowledge as well as some shell scripting
- Knowledge of plotting software, in this case gnuplot

About:

This code is a direct copy of from what I actually used during my physical chemistry research at the University of Oregon! Generally, the project studies the physics of electrons passing through contacts in a photovoltaic (solar) cell. Admittance spectroscopy is one sort of measurement that gives insight about the physics of a particular material included in the solar cell. The experiment involves measuring the sample’s capacitance over a range of temperatures with varying alternating-current (AC) probe frequencies. Sometimes you can observe a step in the capacitance as temperature increases, and this can be related to an energetic defect within the bandgap of the solar cell’s semiconductor absorber material. The primary numerical results of this particular experiment are a defect energy (E_d), attempt-to-escape frequency (ν_0 or `nu0`), and their associated uncertainties. To elaborate, I will try to provide a minimal physics background while still emphasizing the relevance.

The usable current which comes out of the solar cell is due to solar light exciting electrons in the absorber material and extracting them through contacts. For an electron to dissipate this energy absorbed from light, it must do so with a physical process translating electrons between discrete quantum states of energy. In solar power technology, we want this physical process to occur in some sort of electrical circuit load as opposed to within the solar cell. The natural property of semiconductors is that they have a “bandgap” which is a large energy difference between occupied and unoccupied quantum states (when there is no outside thermal or photo-excitation). The result of this that when solar light excites electrons across the bandgap to “usually-unoccupied” states, it takes a relatively long time for such an electron to relax. And thus some asymmetry in the solar cell can drive the electron through a contact before the energy dissipates as heat or something else.

In the case of some crystallographic defect or impurity in the absorber material, sometimes quantum states can appear within the bandgap. These defect states can "trap" electrons and force them to dissipate energy with transitions smaller than the bandgap energy. This decreases the "lifetime" of mobile electrons carrying solar energy, and thus reducing the efficiency of the solar cell device. Quantum mechanical calculations can predict this (E_d) energy difference between this mid-gap state and the band edges dependent upon crystal structure, atom arrangement, etc. So if we can measure a characteristic defect energy (E_d) experimentally, this can give insight into the microscopic physics going on within the material.

The theory behind the experiment is much more complex, but I will try to quickly summarize. In a solar cell with a single rate-limiting contact, its capacitance can be representative of a charge depletion region with a particular depth from the contact interface (like a parallel plate capacitor). If you were to apply an outside voltage bias to the device, it would grow or shrink this depletion length, which translates to quantum states in the material charging & discharging. In addition, the environmental temperature (T) helps determine the depletion length; it gives rise to a population of excited electrons which are accessible to being moved. If we apply an AC "probe" voltage to the device, electrons in the depletion region are gaining and losing a characteristic energy (E_a) over time which we can predict by writing a rate equation:

$$r = T^2\, \nu_0\, exp\left(\frac{-E_a}{k_b T}\right) \tag{1}$$

$$E_a = k_b T\, ln\left(\frac{\nu_0\, T^2}{f}\right) \tag{2}$$

The $T^2 \nu_0$ prefactor can be physically thought of as how quickly an electron attempts to escape a trap, ν_0 being the fastest rate at high temperature. The ν_0 limitation comes from a theoretical cross-sectional area by which the flux of electrons must flow through (on the atomic scale). At a given frequency, as we increase temperature, E_a will increase. Once $E_a = E_d$, electrons from the defect state will begin to escape with the AC probe, and so the capacitance $C = \frac{dQ}{dV}$ will increase. Realistically, the concentration defect states will be distributed normally (with a Gaussian form) within the bandgap, thus giving a curvature to the step on the $C(T)$ plot. So the inflection point of the step gives the characteristic defect energy, $E_d = E_a$.

Now if we rewrite the equation as:

$$ln\left(\frac{f}{T^2}\right) = \frac{-E_d}{k_b T} + ln(\nu_0) \tag{3}$$

This allows us to form an "Arrhenius plot" of $ln\left(\frac{f}{T^2}\right)$ vs $\frac{1}{T}$, and get defect energy from the slope and ν_0 from the intercept. In the included plots, note that the Arrhenius plot's $\frac{1}{T}$ axis continues all the way to 0, making the value of ν_0 more apparent.

How to use code:

I would suggest executing code within some unix shell, such as bash. The only dependencies you need installed are gnuplot and python with the scipy, numpy, and base packages. I have already executed the scripts for you, though. Navigating to the directory with the project files, the data taken during experiment is in a tab-separated plain-text file `data/ct.dat`. The `ct.plt` file is a here-document script for gnuplot to make a basic plot for the data, saved in the corresponding `.png` file. It is written in such a way to presume you execute it from a shell in the directory it is located in. Otherwise, you would have to edit the script to directly point to the path of the data file.

The primary python script to do the analysis is `CTworkup.py`. You will have to go into the file and change the paths to the input file to get it to run. For each frequency, it fits the data over a range of temperatures to the form of a sigmoidal function, which will give the temperatures for the Arrhenius plot. This sigmoid was of the form:

$$C = \frac{L}{1 + exp(-k(T - T_0))} + C_{min} \tag{4}$$

where T_0 is the temperature of the inflection point, k is a growth factor, and L is the change in capacitance over the step ($C_{max}-C_{min}).

Now the script outputs the important results to `data/CT_out.tsv`, as well as the information needed to make the Arrhenius plot. The script will print a bunch of important information to the terminal, which I have piped into `data/CTworkup.log`.

Next, `ct2.plt` pulls the sigmoidal fits from the log file and overlays them with the data. And `arrhenius.plt` takes data from `data/CT_out.tsv` to plot the linear fit. If we open `data/CT_out.tsv`, we can see we calculated $E_d = 0.33 \pm 0.05$ eV and $\nu_0 = (1.2 \pm 0.1)e8$ Hz. Note that $ln(1.2e8) = 18.6$, consistent with the intercept on the Arrhenius plot.

The calculation is assuming 2% error in the capacitances, as in the `CTworkup.py` source. The log file gives chi square values for each fit to confirm whether 2% was a fair estimate for error. In addition, the log file gives R^2 values and covariance & correlation matricies for each fit.