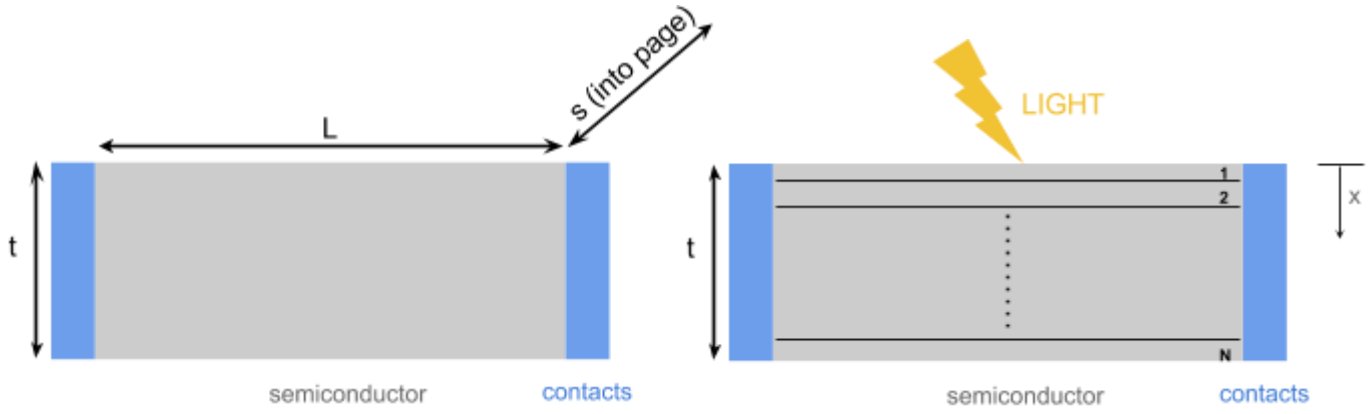
Luisa Barrera
Frances Lenahan

Rupak Chakraborty
Riley Brandt

Extracting mobility and lifetime values for charge carriers from Labview data

The main.m, a Matlab framework, allows to both analyze experimental Labview data and extract key parameters of the charge carriers by using different possible models. The code to extract the data was set up by Lenahan under Brandt's supervision and the models were developed by Barrera under Chakraborty's supervision.
What follows is a breakdown of the framework, with explanations of the models and logic used and a summary of the results to be obtained (everything should be explained in detail in the code).

main provides the framework in which all of the logic and functions are used and called. It starts off by establishing the parameters of the experiment, such as s, the length of the interface between a contact and the semiconductor material, t, the thickness of the sample, and L, the distance between two contacts, as well as the hard-coded constants that are used throughout the code.



One of the leading assumptions in this code is that the generation of the charge carriers is depth-dependent. For easier calculations, this was approximated by considering the slab as composite of N number of slabs of thickness t/N, as can be seen in the diagram. Each slab has a different resistance determined by the depth-dependent conductivity above, and the resistance of the composite can be calculated by summing the resistances in parallel (this will be explained in more detail later on in the document). While this value N can be changed at the beginning of the code, the current number (5000) was shown to be a large enough number of slabs such that the values would not dependent on the thickness of the slabs, but small enough to avoid excess calculations. For more information on this process, one should consult Lenahan's thesis. Another case of minimizing variations would be running the approximation of the charge carrier values several times (right now, numberOfIterations remains at 1).

The characteristics of the charge carriers that are determined here are the lifetime (tau), the mobility (mu or mob) and the surface recombination velocity (srv). Surface recombination velocity, or the speed at

which the charge carriers recombine at the surface of the sample, is an important boundary condition, which for simplicity sake is considered null (this comes into play in later calculations). Because of this simplification, one should note that the logic of the code includes srv in all of the optimization processes and results, but these hold no impact in the actual calculations taking place. The values Ntau, Nsrv, and Nmu are used to find the best fit of tau, mu, and srv to the experimental data. This process will be fully explained in the code and later on.

Using the assumption that the charge generation is depth dependent, one must also consider that the absorption coefficient of the sample is also depth dependent. In the first part of the code, the data from the .abs and the AM1.5 spectrum are compared (this involves logical selections of the data for comparison, which seems overly complicated because of the format of data collection for the AM1.5 spectrum but is simply extracting the same wavelengths from each file). Once this is established, the Labview data can be analyzed.

JVItxttomat will take the data files obtained from the Labview code and generate a matlab file containing the sample name, temperature values, illumination values, voltage and current data, and start and end temperatures for each measurement.

The code iterates through every data file to obtain each of the above values for each IV measurement under specific tempe arature and illumination. Code begins by checking that the file in question is a data file, then proceeds to locate the temperature and illumination values associated with that measurement in the full name of the data file. The naming of files is very important. “name_IVTraw” must be the title of the file containing all data files. Data files should be called “name_IVT_###K_ill#_0_DATE_TIME.txt”.

> For example, if “name” is “SnS150219o”, then the file name will be “SnS150219o_IVTraw” and one specific data file will be “SnS150219o_IVT_130K_ill0_0_20160302_527PM.txt”. This measurement was taken at 130K under illumination number 0* on the second of March, 2016 at 5:27pm.

When “name” does not contain an underscore, searching for the temperature begins at the second underscore and for the illumination at the third underscore. Main.m should account for these differences and runs the appropriate file accordingly.

All temperature values and illumination numbers are obtained from the data file names are placed into their respective lists with no repeating values. The illumination numbers are changed from index 0 to index 1 to be used as locators for the VI cell, which will shortly be created. The Labview code creates an index 0. To avoid this step, the Labview code will need to be changed.
Illumination numbers refer to “G_ODpercent”, at the beginning of the code for the actual illumination values (how much the sample is actually being illuminated). This is a manually inputted list from a light calibration measurement at the beginning of these experiments. In the data file, an illumination number of 7 will become a MatLab illumination value of 8, which will refer to the eighth input in “G_ODpercent”, which is 1.12, or, 112% light intensity.

To populate the VI cell, first, voltage (“V”) and current (“I”) are taken as data sets from each file, and the temperature reading from temperature sensor A (“TA”), and temperature reading from temperature sensor B (“TB”), are pulled from the header of each file. The temperature sensor information is text which gives both the start temperatures and end temperatures for the measurement. The VI cell, first called “VI1”, is populated with three depths; desired temperature, illumination value, data - where ‘data’ will be “V” in position 1, “I” in position 2, “TA” in position 3, and “TB” in position 4. For example, to find all voltage measurements at 310K and 112% illumination, find first the indices corresponding to 310K and 112%, (which will be 10 and 8 for SnS150219o) and use position 1. VI1(10 , 8 , 1) should give this result. Inputting instead VI1(10 , 8 , 3) will show the start and end temperatures for temperature sensor A.

Finally, once all data files are processed and the loop is closed, the temperature list is rearranged from lowest to highest value. New index values are created which map from the old vector to the new vector. This new list and index values are then used to create a new VI cell, called simply “VI”, instead of “VI1”.

A file is saved with the title: “name_extracted.mat”, where “name” is, again, the name of the sample. For example, “SnS150219o”. This saved file contains “name”, temperature list “T”, illumination list “ill1”, and the ordered VI cell “VI”.

It is also possible to use this code on its own to generate IV plots over illumination and temperature.

Extracted Resistance takes the “name_extracted.mat” file generated by JVItxttomat and finds the resistance of each IV measurement. Geometric constants of the device are inputted at the beginning of the code, all in SI units.

First, an IV curve is generated from each dataset gathered from “VI” (a cell generated in JVItxttomat), running through each temperature in the outside loop and each illumination in the inside loop. A linear fit is taken of each curve and the resistance is recorded as the inverse of the slope of the fit. Each resistance is then put into a list - “RList”.

A file is saved with the title “name_summary.mat”, where “name” is, again, the name of the sample. This file contains “name”, temperature list “T”, illumination list “ill” (which equals ill1), and the resistance list “RList”.

A simplified mobility-lifetime product (“muTaui”) can be calculated but is not saved. This “muTau” does not consider depth dependence or diffusion. The generation rate, “G”, is acquired by taking the sum of photon flux in AM1.5 at every wavelength above the band-gap of the desired material. First, “RList” is multiplied by the area of the device over the device length, to create a resistivity list. The conductivity is recorded as the reciprocal of the resistivity list and called “sigma”. Dark conductivity is the first entry in “sigma” and is labeled as “sigmaDark”. The change in conductivity or the difference between “sigmaDark” and the rest of the entries in “sigma”, is a new list called “deltaSigma”. A for loop calculates the mobility-lifetime product at each illumination using the entries in “deltaSigma” and dividing them by, “q” the charge of an electron, a units factor, and the generation considering illumination “G*G_OD(illi)”.

Several plots can also be generated but are not saved, such as Conductivity or Mobility-Lifetime Product vs. Illumination or Temperature, Illumination and Mobility-Lifetime Product that are given of Log scales.

Once the data is extracted from the files, several variables such as the temperature and the illumination of each sample can be populated (the illumination vector contains the indices for each illumination from the G_OD percent vector). Since there are several temperatures and illuminations to compare, a loop runs through all of these and populates an output table (each run-through the loop, a new row is generated thanks to itcount). The subsequent analysis compares the dark and the illumination as generated by the loop (first dark vs dark, then dark vs the first illumination used, and so on). The analysis can also be done in a simplified manner for one temperature and simply comparing fully dark to fully illuminated, and is commented in the code as the simplified version.

In the loop, the aerial generation rate considering both its wavelength and depth-dependence is calculated as follows:

$$G(x,\lambda) = \alpha N_0(\lambda)\, e^{-\alpha x} = \int_{\lambda_1}^{\lambda_2} \alpha(\lambda)\, \frac{\varphi(\lambda)}{hc/\lambda}\, e^{-\alpha(\lambda) x}\, d\lambda$$

with $\alpha(\lambda)$: absorption coefficient, dependent on wavelength λ

$N_0(\lambda)$: photon flux, dependent on wavelength λ

x : depth from top surface of sample

λ_1 , λ_2 : wavelengths

hc/λ : energy associated to each photon

$\varphi(\lambda)$: energy flux density, which integrated on $d\lambda$ gives energy flux.

For the cryo setup from the experimental values, the energy flux density $\varphi(\lambda)$ is given by the AM1.5 spectrum previously cropped, and the specific wavelengths λ_1 and λ_2 are determined by the neutral density filters that are used. Thus we can replace the areal generation rate above with the more general form and complete the double integration, first by wavelength, then by depth:

$$G = \int_0^t G(x,\lambda)\, dx = \int_0^t \int_{\lambda_1}^{\lambda_2} \alpha(\lambda)\, \frac{\varphi(\lambda)}{hc/\lambda}\, e^{-\alpha(\lambda) x}\, d\lambda\, dx$$

Once the aerial generation rate is calculated, the mobility-lifetime product can be derived, going back to the initial assumption of depth-dependence of the generation but no diffusion of charge carriers. In the absence of diffusion, the generation rate must equal the recombination rate at each point in the sample, such that:

$$G(x) = U(x) = \Delta n(x) / \tau \qquad \text{thus} \qquad \Delta n(x) = \tau\, G(x)$$

where

τ : electron lifetime

G : depth-dependent generation rate

$\Delta n(x)$: depth-dependent excess electron density

Note that the excess electron density is depth dependent, which makes the conductivity also depth-dependent:

$$\sigma_{light}(x) = e\,(\mu_e\,\Delta n(x) + \mu_h\,p_o)$$

In order to calculate the measured resistance in the light, this depth-dependence must be taken into account. As mentioned previously, one could consider the sample slab as a composite of N slabs, each of thickness t/N. Each slab has a different resistance determined by the depth-dependent conductivity above, and the resistance of the composite can be calculated by summing the resistances in parallel:

$$\frac{1}{R_{light}} = \sum_{1}^{N} \frac{1}{R_i} \qquad \text{with } \frac{1}{R_i} = \frac{\sigma\, s\, (t/N)}{L}$$

Combining these equations and taking the limit of the slab thickness to be 0, we have

$$R_{light} = \left(\int_0^t \frac{s}{L}\, e\,(\mu_e \Delta n(x) + \mu_h\, p_0)\, dx\right)^{-1} = \left(\int_0^t \frac{s}{L}\, e\,(\mu_e\, \tau\alpha\, N_0\, e^{-\alpha x} + \mu_h\, p_0)\, dx\right)^{-1}$$

From R_{light}, the mobility-lifetime product can be extracted:

$$\begin{aligned}
\frac{1}{R_{light}} &= \left(\int_0^t \frac{s}{L}\, e\,(\mu_e \tau\alpha\, N_0\, e^{-\alpha x} + \mu_h\, p_0)\, dx\right) \\
&= \frac{es}{L}\left(\frac{\sigma_{dark}}{e}\, t + \int_0^t \mu_e \tau\alpha\, N_0\, e^{-\alpha x}\, dx\right) \\
&= \frac{es}{L}\left(\frac{\sigma_{dark}}{e}\, t + \mu_e \tau \int_0^t \alpha\, N_0\, e^{-\alpha x}\, dx\right) \\
&= \frac{es}{L}\left(\frac{\sigma_{dark}}{e}\, t + \mu_e \int_0^t \Delta n(x)\, dx\right)
\end{aligned}$$

Using the definition of the areal generation rate in units of [carriers/(m^2*s)] gives:

$$\frac{1}{R_{light}} = \frac{es}{L}\left(\frac{\sigma_{dark}}{e}\, t + \mu_e \tau G\right)$$

making: $$\frac{1}{R_{light}} - \frac{1}{R_{dark}} = \frac{es}{L}\,\mu_e \tau G$$

$$\mu_e \tau = \frac{1}{\frac{es}{L} G}\left(\frac{1}{R_{light}} - \frac{1}{R_{dark}}\right)$$

Since all the values of this equation are known or measured, the mobility-lifetime product is defined specifically for this material (assuming depth-dependence).

One should note, however, that the result obtained is the mobility-lifetime product, not the individual values of each parameter. To find these (and the value for srv once used), a process of optimization is done in **optimizeParameters**, modified from Brandt's code. The three values are optimized by

calculating the difference of RlightCalculate using some inputted value for each (chosen at random through many iterations) with the experimental value calculated previously Rlightfilter. This is done by taking a certain number of points (Ntau, Nmu, Nsrv as mentioned before) over the established ranges of mu, srv, and tau and choosing at random points within the intervals.

This function has two plot outputs: RMS (the error value for the difference calculated before) vs the mu-tau value calculated here; RlightCalculate (Rlight calculated from the different values tried out) vs the mu-tau value. These plots both appear and are saved in a folder specific to the data set being analysed. It also outputs the bestTau, bestMu, and bestSRV fit using the RMS closest to 0.

Now knowing the best values for tau, mu, and srv, we want to back-calculate the value of Rlight using two possible models, using again a loop to cycle through very close values for each variable. The two possible models (one needs to be commented out) are:
solver1 that uses the initial assumption of depth-dependence for the generation, which means it should give consistent results with what was found previously, meaning that simply $\Delta n(x) = \tau\, G(x)$ was used.
solver2 that includes the diffusion of charge carriers as they are generated. This involves solving the second order differential equation resulting from steady-state photoconductivity modeling.

$$G - R + D\frac{\delta^2 \Delta n}{\delta x^2} = \frac{\delta \Delta n}{\delta t} = 0$$

Where G: generation of the charge carriers
R: recombination of the charge carriers
Here approximated as $R = \frac{1}{\tau_{bulk}}\Delta n$
The third term is the diffusion term, where D is the diffusion coefficient
The boundary conditions for this second order differential equation are:

$$\frac{\delta \Delta n(x=0)}{\delta x}\Big|_{x=0} = \frac{s}{D}\Delta n(x=0) \qquad \text{and} \qquad \frac{\delta \Delta n(x=t)}{\delta x}\Big|_{x=t} = \frac{s}{D}\Delta n(x=t)$$

As a simplification, the surface recombination velocity s (srv in the code) is considered to be 0. This gives Dirichlet boundary conditions to the differential equation, which combined with some finite element analysis and approximation of the second derivative gives:

$\Delta n(x) = A / F$

Where A is a coefficient matrix A as set up in the code
and F is the depth-dependent generation rate (already integrated in terms of wavelength, and with a much smaller step than previously used in the code, so requiring an interpolating function to fill in the gaps)

An example of the system of equations/matrices for the first five equations (i = 1:4 as i goes from 0 to t, total depth or thickness of the sample):
Using the approximation of the second derivation, where $\Delta n_i = U_i$ will be substituted in

$$\frac{\delta^2 \Delta n_i}{\delta x^2} = \frac{\Delta n_{i-1} - 2\Delta n_i + \Delta n_{i+1}}{\Delta x^2} = \frac{U_{i-1} - 2U_i + U_{i+1}}{\Delta x^2}$$

The original equation then becomes, for a depth i:

$$D\frac{\delta^2 \Delta n}{\delta x^2} - \frac{1}{\tau_{bulk}}\Delta n = -G \;\leftrightarrow\; D\frac{\delta^2 U_i}{\delta x^2} - \frac{1}{\tau_{bulk}}U_i = -G_i$$
$$\leftrightarrow\; D\frac{U_{i-1} - 2U_i + U_{i+1}}{\Delta x^2} - \frac{1}{\tau_{bulk}}U_i = -G_i$$
$$\leftrightarrow\; D\left(\frac{1}{\Delta x^2}U_{i-1} - \frac{2}{\Delta x^2}U_i + \frac{1}{\Delta x^2}U_{i+1}\right) - \frac{1}{\tau_{bulk}}U_i = -G_i$$

$$\leftrightarrow D(\frac{1}{\Delta x^2}U_{i-1} - (\frac{2}{\Delta x^2} + \frac{1}{\tau_{bulk}})U_i + \frac{1}{\Delta x^2}U_{i+1}) = -G_i$$

Furthermore, knowing that $\frac{\delta^2 \Delta n}{\delta x^2}|_{x=0} = 0$ from the boundary condition means that the false boundary condition of $\frac{\delta \Delta n_{i+1} - \delta \Delta n_{i-1}}{2\delta x} = 0$.

For i from 1 to 5, this gives the following matrix: (insert image with proper formatting, showing the pattern that is built in the code)

$$D \begin{bmatrix} -\frac{2}{\Delta x^2} - \frac{1}{\tau_{bulk}} & \frac{1}{\Delta x^2} & & \\ \frac{1}{\Delta x^2} & -\frac{2}{\Delta x^2} - \frac{1}{\tau_{bulk}} & \frac{1}{\Delta x^2} & \\ & \frac{1}{\Delta x^2} & -\frac{2}{\Delta x^2} - \frac{1}{\tau_{bulk}} & \frac{1}{\Delta x^2} \\ & & \frac{1}{\Delta x^2} & -\frac{2}{\Delta x^2} - \frac{1}{\tau_{bulk}} \end{bmatrix} \begin{bmatrix} U_1 \\ U_2 \\ U_3 \\ U_4 \end{bmatrix} = \begin{bmatrix} G_1 - U_0 \\ G_2 \\ G_3 \\ G_4 - U_5 \end{bmatrix}$$

Where $U_0 = U_2$ and $U_5 = U_3$ from the false boundary condition, as U_0 and U_5 are unknown.

Thus, to get the depth-dependent and diffusion considering values for delta n, simply divide the two matrices.

Once these solvers populate the deltaN value for that specific tau, srv, and mu entry, **calculateRlight** calculated Rlight using:

$$R_{light} = 1/(\frac{1}{R_{dark}} + \frac{es}{L}\,\mu_e\,\Delta n\,)$$

And the deltaR matrix is populated by taking the difference of the recently calculated Rlight and the original experimental data one.

The final output plot of this section is the deltaR matrix contour plot vs the mu-tau value calculated. (One should note that currently the absolute value is also being plotted and that this could easily be replaced by more contour plots for each value of srv.)

The final lines of code serve as the save mode for all of the figures and data: a specific folder is created for the data set analyzed, and the output plots (as .mat files), initial parameters inputted (as a .mat file), and summary table of all the fitted values (as a .txt file) is saved under that folder with the appropriate name. One should note that the table that appears in Matlab might not display the values properly, so to get the specific scientific notations open the .txt file.