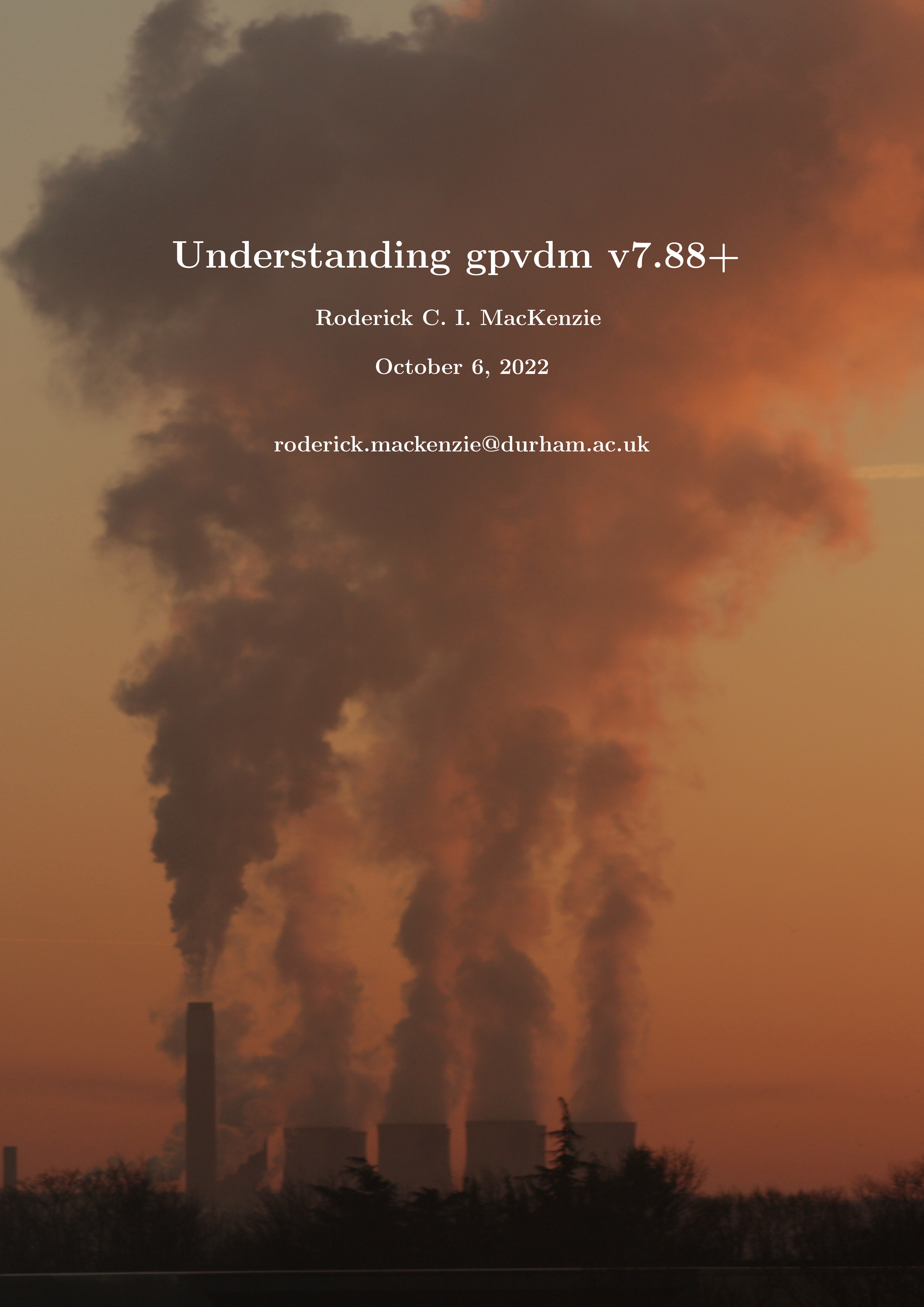


Understanding gpvdm v7.88+

Roderick C. I. MacKenzie

October 6, 2022

roderick.mackenzie@durham.ac.uk

Please do not cite this manual. Please see the section 1.6 on how to cite the model in your work.

Front cover: A picture of a thermal power station in Ratcliffe-on-Soar Nottinghamshire taken on a cold January afternoon in 2017. Most of the emissions you see in the image is water from the cooling towers however the gasses rising form the tall thin chimney on the left hand side of the image are the products of burning hydrocarbons which previously buried in the ground for about 300 million years.

Contents

Chapter 1

Introduction

1.1 What is gpvdm?

Gpvdm stands for *general purpose photovoltaic device model.* As its name suggests it was originally developed to be a general purpose model for simulating photovoltaic devices, including organic and perovskite cells. However, since its initial development the model has expanded to simulate many other classes of optoelectronic devices including, Organic Light Emitting Diodes (OLEDs), Organic Field Effect Transistors (OFETs), large area printed devices, optical filters, photonic crystals and many more. In general gpvdm can simulate any opto-electronic-device where electrons, photons (and also heat - phonons) interact. Although the name no-loner really fits what gpvdm it has stuck as there are hundreds of papers which cite gpvdm, and it has been downloaded by thousands of people across the globe (see figure 1.1) it therefore no longer makes sense to update the name.

Figure 1.2 shows some of the classes of devices gvpdm can simulate. The model should be though of as a general, 1/2D electrical drift-diffusing solver, combined with a 3D thermal solver and advanced optical models. The thermal model is 3D so is the exciton diffusion model. The key feature which sets gpvdm apart from many other models is the efficient way it handles SRH carrier trap states.

Gpvdm is written in small letters because its a UNIX command like grep, it is permissible to write it with a capital letter at the start of a sentence, but it should *never* be fully capitalized as GPVDM.

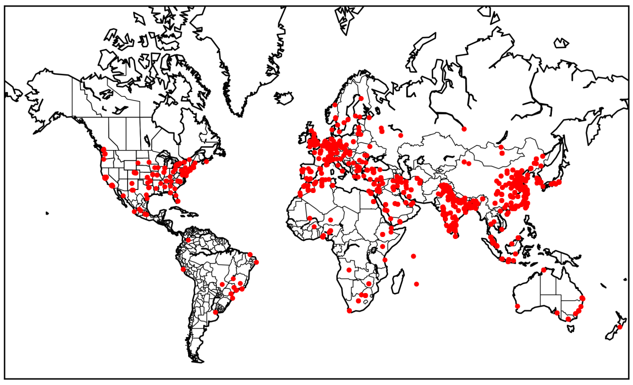


Figure 1.1: A map of locations where gpvdm has been downloaded.

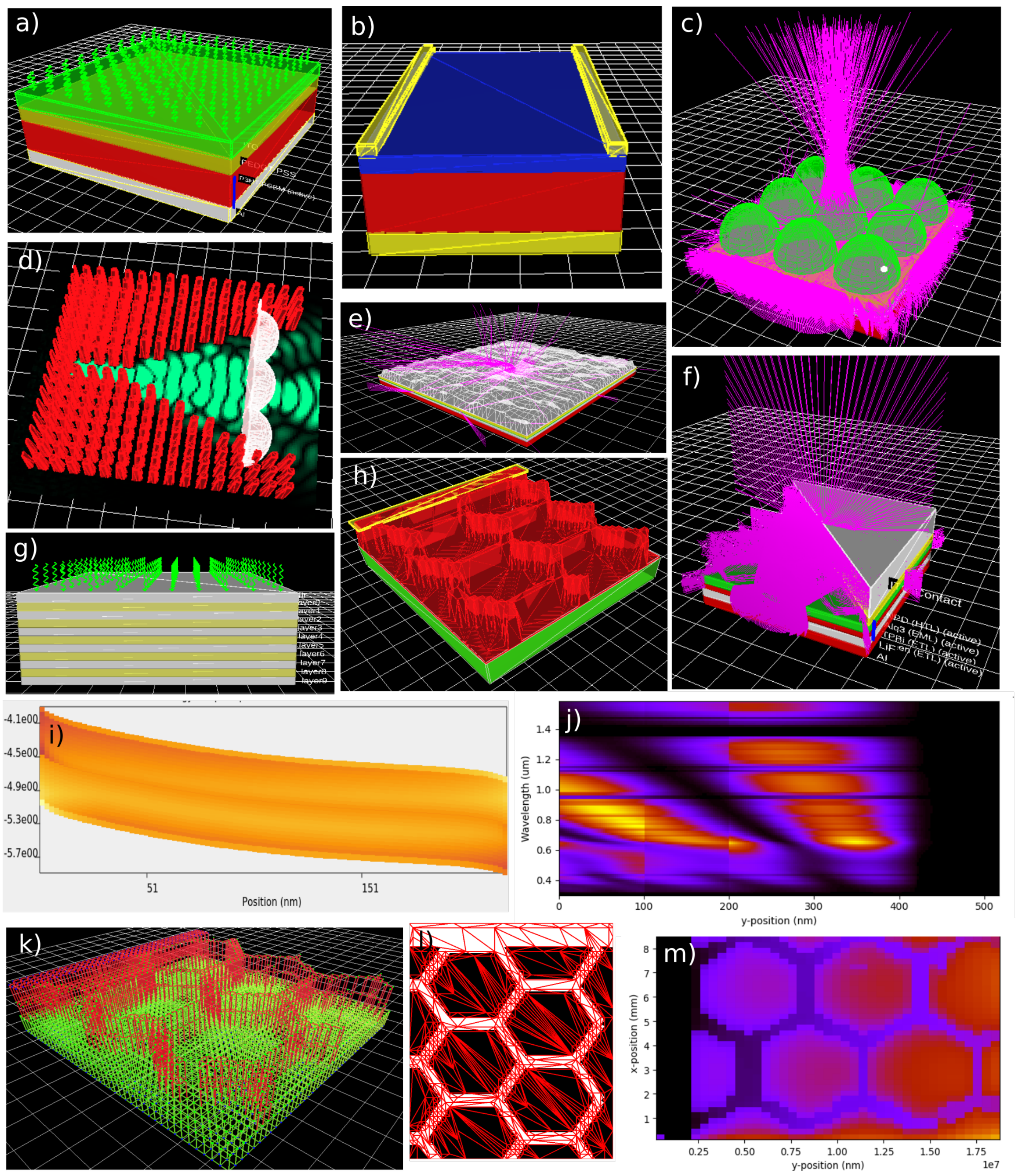


Figure 1.2: a); Organic/Perovskite solar cell simulation; b) OFET transistor simulation; c) Microlens simulation; d) Photonic waveguide simulation; e) light escaping structured films; f) OLED simulation; g) Optical filter simulation; h) large area device modelling (hexogonal contacts); i) Mapping carrier density in position/energy space; j) Building complex 3D meshes; and k) resistance maps of large area devices.

1.2 Why gpvdm?

By burning fossil fuels we are releasing $\sim$ 33.3 gigatonnes of CO_2 per year [1] and thus humanity is steadily changing the composition of Earth's atmosphere. Since 1960, CO_2 in the atmosphere has risen by around 30% this in turn is increasing average global temperatures[2] and making our home planet Earth, a more difficult place to live on. We therefore have two choices, either cut emissions or face an existential crisis. Thin film devices such as solar cells and OLEDs offer a viable way to reduce our CO_2 emissions, either by providing low carbon electricity, or providing an efficient way to use the energy once generated. It is therefore important that technologies based on thin film devices continue to be developed and succeed. By developing and releasing gpvdm, I hope, I am enabling scientists throughout the world to understand these devices a little bit better, which I hope will contribute in a very small way to solving our climate crisis.

Solar cells and OLEDs happen to come from a class of devices called diodes. This class of devices has many uses including optical sensors, medical sensors, switches, rectifiers. Thus as a pleasant side effect of gpvdm the development of these devices is also being helped.

1.3 About this book/manual

This manual is (in the end) intended to be the definitive guide to simulating devices with gpvdm. The idea is that one will be able to read this book and learn in a step by step way how to simulate many modern opto-electronic devices. However, not all aspects of this book are yet finished. I therefore recommend you also watch the YouTube channel (and subscribe! ;)) where I describe many of the features in more detail and give demonstrations on the use of gvpdm. I would suggest you treat the videos as lectures (and take notes) rather than entertaining videos (well I hope they are entertaining too!). New releases are generally announced on Twitter, which I also suggest you follow the gpvdm account and make sure you are using an up-to date version. I often release version every week. Please read papers which were published from this model - do also read the supplementary information (SI) to the papers, as I often write about the model in there. This book starts off with explaining how to simulate organic solar cells. This is because organic solar cells are the easiest class of device to simulate, it then moves onto Perovskite devices, and OLEDs. More complex classes of devices follow. If you are new to simulation work or in indeed gpvdm, I suggest you start with the first chapters and work your way to the more complex devices.

1.4 What is the history of gpvdm?

I started writing gpvdm just after finishing my PhD in 2009 while taking a break for academia and deciding what to do next. At that time it was a simple drift diffusion diode model which did not take account of disorder. The majority of the complex core of gpvdm which deals with disordered materials was written while I was working in the Physics Department at the Imperial College London in 2010-2011. During this time I was working for Jenny Nelson on organic solar cells, it was a very productive and exciting time. Since then the model has been expanded to model many other classes of material system and classes of devices. Since 2009 thousands of people have downloaded gpvdm and hundreds (the list is by definition always out of date) of people have published their own papers using the model.

1.5 What is the roadmap for gpvdm?

The aim is to make gpvdm a completely general opto-electronic model which can be used by anyone to learn about and explore the world of novel opto-electronic devices. I want gpvdm to be an engine which people can use to push their own research forward and for education. The exact road map on how to get there is not defined. As collaborators contact me asking for new features I add them, what comes first depends on what people want.. I never view gpvdm as finished, and release improvements in small increments, therefore if you discover and report a bug, check back in a week or so to see if it is fixed in the next version. The same goes for this book, it evolves weekly as I write it. So if a section is missing, check back next week it might be finished.

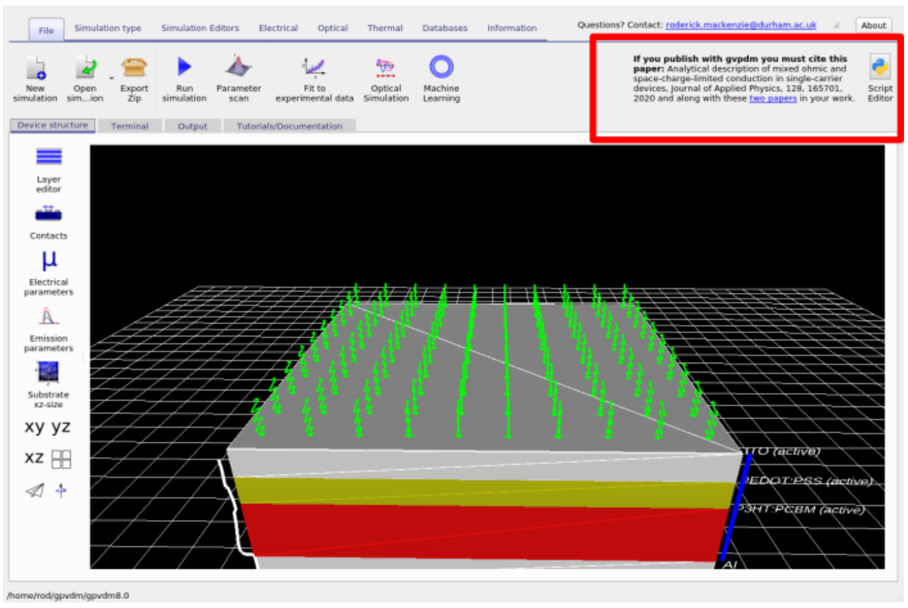
1.6 Using gpvdm in industrial/academic work

You are free to use gpvdm in industrial/academic work. In fact, I'm super happy if you use it in your work, papers or books. However, the following further conditions apply:

1. If you use gpvdm to generate results, then clearly say so in your work. This can be as simple as one sentences saying: "we used gpvdm to perform the simulations"

2. If you publish a book, paper or thesis where gpvdm has been used you must cite at least three papers associated with the model. To find out which papers to cite, click on the area indicated in red in figure 1.3 when using the model. You are free to choose which papers to cite but PLEASE do not cite the manual. I can't include the manual in paper lists when applying for funding.

I ask you to do this because citations are an easy way to demonstrate that people are using gpvdm. Demonstrating use is key to finding money/people to continue the development of gpvdm. So by doing this you are guaranteeing the future of gpvdm and its continued availability for others. Thank you!


Questions? Contact: roderick.mackenzie@durham.ac.uk

If you publish with gvpdm you must cite this paper: Analytical description of mixed ohmic and space-charge-limited conduction in single-carrier devices, Journal of Applied Physics, 128, 165701, 2020 and along with these two papers in your work.

/home/rod/gpvdm/gpvdm8.0

Figure 1.3: If you click on the area indicated by the red box, the model will tell you which papers should be cited.

1.7 Bugs

I get quite a lot of feature requests from people wanting features added or for bugs to be fixed. I really appreciate the feedback! However, I am currently employed at a UK University and my time is split between teaching, research and admin. My performance in my job is measured by the number of high impact papers I push out per year. I therefore have to prioritize feature requests and bug fixes for people who would like to write a paper with me (i.e. my collaborators).... Therefore if you would like:

- A bit of advice on how to do x or y with the model then please do feel free to shoot me a mail, and I will do my best to get back to you. If you don't hear back from me just send the mail again.. I get loads of e-mails, and things get lost if I don't answer quickly.
- If you want to report a bug, then please do that, and I will do my best to fix it in the next release. But I can't promise when it will be fixed.
- If you would like a features added or a steady stream of help (i.e. you are asking for my time) then please consider inviting me to join in your work and collaborate on a joint paper. I am happy to add what ever feature you want to the model, or fix what ever bug you may have but in return I would ask for the inclusion of my name on the author list. By doing this it makes it much easier for me to justify sinking time into your project.

If you don't need help from me to use gpvdm then please feel free to do what you want with the results - no need to contact me.

1.8 Installing gpvdm

1.8.1 Windows (if you have admin rights)

Go to the download page for gpvdm at `http://www.gpvdm.com/windows.php` and download the latest version. Simply double click on it and say yes to all questions. In general I release a new version every couple of weeks and it's worth keeping your version up-to-date. On modern versions of windows, windows will ask you if you want to install an unsigned executable from an unknown author, and warn you that this could damage your computer. The reason you get this message is because I have not cryptographically signed the .exe file. I have not signed it because I do not own a private cryptographic key with which to do this. To get such a key I would have to send my passport off to a key authority to prove who I am and then pay them 500 pounds/year for the privilege of them validating who I am. Needless to say, that I am not very excited about paying 500 pounds/year so you will just have to click away the warnings from windows.

1.8.2 Windows (No admin rights)

If you don't have admin rights to your computer it can be hard to install new software, gpvdm offers the option of running gpvdm while not properly installed. Download the zip file containing gpvdm from `https://www.gpvdm.com/download_no_admin.php`. Once you have downloaded the zip archive, open the zip file and extract the folder *pub* to c:\. Then rename the folder to be calledc:\gpvdm. Once you have done this run the executable c:\gpvdm\gpvdm.exe (see figure 1.4).

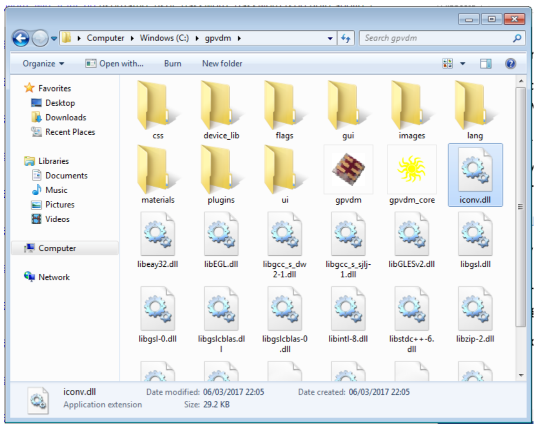


Figure 1.4: Running and installing gpvdm. Double click on the gpvdm icon to run the model.

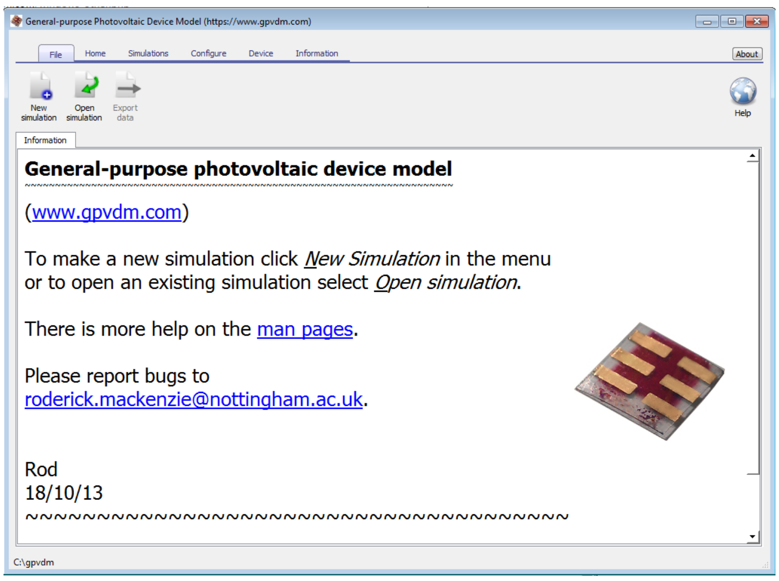
Chapter 2

Getting started

2.1 Simulating a simple solar cell

No matter which type of device you want to simulate, if you are new to gpvdm my advice is to start off with this organic solar cell simulation. Organic solar cells are by far the most simple class of device you can simulate, and will let you understand the basics of the package without having to deal with 2D effects, perovskite ions of light emission.

On both windows gpvdm will install on the start menu, click on it to launch it. Once run, a window resembling that in figure 2.1 will appear.



General-purpose photovoltaic device model

(www.gpvdm.com)

To make a new simulation click *New Simulation* in the menu or to open an existing simulation select *Open simulation*.

There is more help on the man pages.

Please report bugs to roderick.mackenzie@nottingham.ac.uk.

Rod
18/10/13

Figure 2.1: The main gpvdm simulation window.

Click on the *new simulation* button. This will bring up the new simulation window (see figure 2.2). From this window select the *Organic Solar Cell* option and click next. Gpvdm dumps a lot of data to disk, I therefore recommend you save the simulation to a local disk such as the C:\drive, a network drive or usb stick drive will be far too slow for the simulation to run. I would also not save the simulation onto OneDrive or Dropbox as they are also too slow and saving it there will generate a lot of network traffic. If you are a power user doing a lot of fitting of experimental data I would also recommend (at your own risk(!)) disabling any extra antivirus software you have installed, as quite often the antivirus software can't keep up with the read/writes to disk.

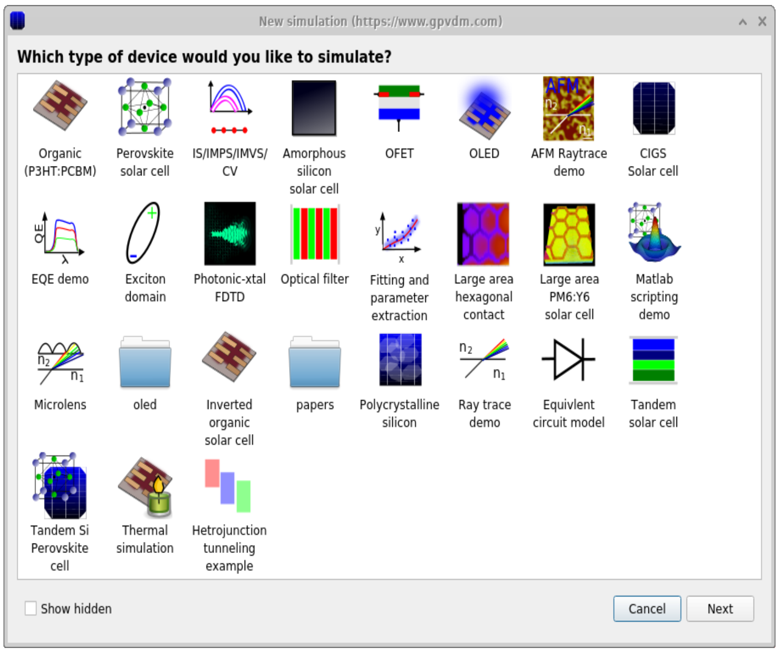


Figure 2.2: New simulation window

Once you have saved the simulation, the main gpvdm simulation window will be brought up (see figure 2.3). You can look around the structure of the solar cell, by dragging the picture of the solar cell with your mouse. Try pressing on the buttons beneath the red square, they will change the orientation to the *xy*, *yz* or *xz* plane. Notice the x,y,z origin marker in the bottom left of the 3D window. The icon with four squares will give you an orthographic view of the solar cell.

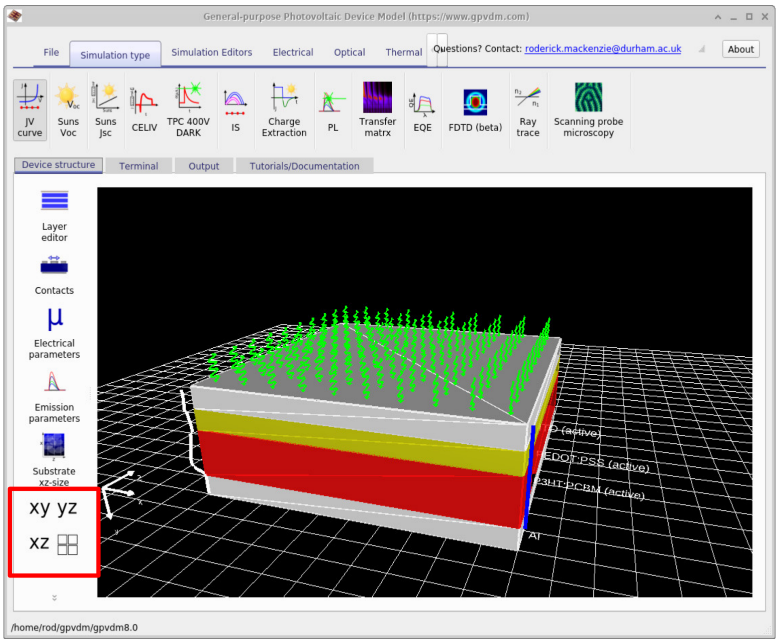


Figure 2.3: The main gpvdm simulation window

Click on the button called *Run simulation* (hint it looks like a blue play button and is located in the *file* one to the right of the "Simulation type ribbon"). This will run the simulation. On slower computers it could take a while. Once the simulation is done, click on the *Output* tab (see figure 2.1), there you will see a list of files the simulation has written to disk.

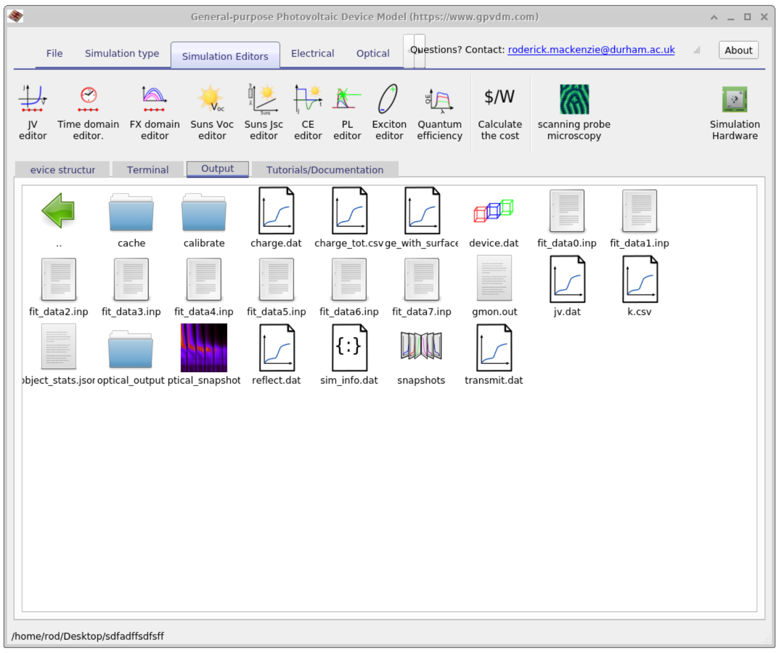


Figure 2.4: The *Output* tab this is just like windows file explorer, you can explore the simulation directory tree.

Key files the simulation produces are listed in the table below:

File name	Description
$jv.dat$	Current voltage curve
$charge.dat$	voltage charge density
$device.dat$	The 3D device model
$fit_data * .inp$	Experimental fit data (for demo)
$k.csv$	Recombination constant k
$reflect.dat$	Optical reflection from device
$transmit.dat$	Optical transition through device
$snapshots$	Electrical snapshots see 15.1
$optical\_snapshots$	Optical snapshots see 15.3
$sim\_info.dat$	Calculated V_{oc}, J_{sc} etc.. see 15.6.1
$cache$	Cache see 15.4

Table 2.1: Files produced by the JV simulation

Try opening *jv.dat*. This is a plot of the voltage applied to the solar cell against the current generated by the device. These curves are also sometimes called the *charistic diode curve*, we can tell a lot about the solar cell's performance by looking at these curves. Hit the 'g' key to bring up a grid.

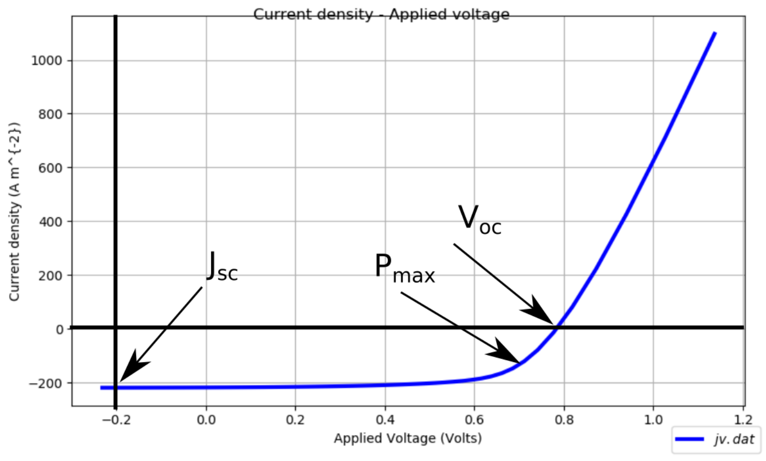


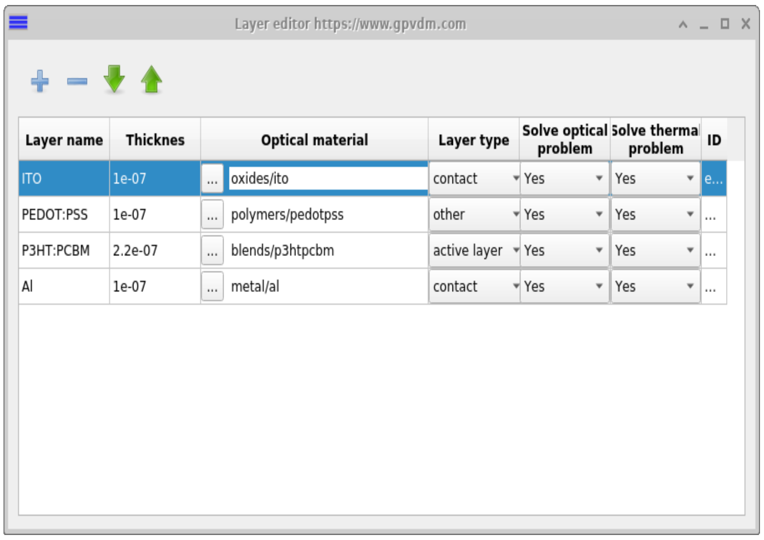
Figure 2.5: The output tab

Now try opening up the file *sim_info.dat*, this file displays information on the performance of the solar cell, such as the Open Circuit Voltage (V_{oc} - the maximum Voltage the solar cell can produce when iluminated), efficiency (η - the efficiency of the cell) , and short circuit current (J_{sc} - the maximum current the cell can produce when it is illuminated). Figure 2.5, shows where you can find these values on the JV curve. The *sim_info.dat* file contains a lot of other parameters, these are described in detail in section 15.6.1.

Question 1: What is the J_{sc}, V_{oc} and Fill Factor (FF) of this solar cell? How do these number compare to a typical Silicon solar cell? (Use the internet to find typical values for a Silicon solar cell.)

2.1.1 Editing device layers

Any device in gpvdm consists of a series of layers (this is sometimes referred to as the epitaxy - this is a term which comes from inorganic semiconductors). The layer editor can be accessed from the main simulation window, under the *device structure* tab. This is visible towards the top of figure 2.3, and the layer editor is visible in figure 2.6. Within the window is a table which describes the structure of the device. The column thickness describes the thickness of each layer. The P3HT:PCBM layer is the layer of material which converts photons into electrons and holes, this is commonly called the active layer. An active layer thickness of 50nm is considered very thin for an organic solar cell, while an active layer of 400nm is considered very thick (too thick for efficient device operation). Vary the active layer between 50 nm and 400 nm, for each thickness record the device efficiency (I suggest you perform the simulation for at least eight active layer widths).



Layer name	Thicknes	Optical material	Layer type	Solve optical problem	Solve therma problem	ID
ITO	1e-07	... oxides/ito	contact	Yes	Yes	e...
PEDOT:PSS	1e-07	... polymers/pedotpss	other	Yes	Yes	...
P3HT:PCBM	2.2e-07	... blends/p3htpcbm	active layer	Yes	Yes	...
Al	1e-07	... metal/al	contact	Yes	Yes	...

Figure 2.6: The layer editor window.

Task 2: Plot a graph (using excel or any other graphing tool), of device efficiency v.s. thickness of the active layer. What is the optimum efficiency/thickness of the active layer? Also plot graph V_{oc} , J_{sc} and FF as a function of active layer thickness. J_{sc} is generally speaking the maximum current a solar cell can generate, try to explain your graph of J sc v.s. thickness, [Hint, the next section may help you answer this part of the question.]

More on the layer editor

The layer editor has the following columns:

- Layer name: Is the English name describing the layer. You can call your layers what you want (i.e. ITO, PEDOT, fred or bob) it has no physical meaning.
- Thickness: Is the layer thickness given in meters.
- Optical material: Specifies the n/k data which is used to describe the materials optical properties. In the simulation the n/k data are taken from experimental values stored in the optical database 10.1 and have nothing to do with the electrical material properties such as effective band gap.
- Layer type: Specifies to the simulation how the layer is treated when performing a simulation. There are three types of layer
  - active: This type of layer is electrically active and the drift diffusion solver will solve the electrical equations in this layer type. See section 18.1. You can have as many active layers as you like but they must be contiguous.
  - contact: This tells the model that a layer is a contact and a voltage should be applied, see section 2.1.6 for more details.
  - other: Any layer which is not a contact or active.

Which layers should be active?: A common mistake people make when starting to simulate devices is to try to make all the layers in their device active because their logic is: Current must be flowing through them so they must be active right? However, in for example a solar cell only the BHJ or in a perovskite device the perovskite layer will have both species of carriers (electrons+holes) and complex effects such as photogeneration, recombination and carrier trapping. So in this layer it makes sense to solver the drift diffusion equations. Other layers which don't have both species of carriers can be treated simple parasitic resistances see section 2.1.4. I would only recommend setting other layers of the device to active (such as the HTL/ETL) if you are trying to investigate effects such as s-shaped JV curves or devices which clearly need multiple active layers such as OLEDs. In general, try to minimize the number of active layers and always keep simulations as simple as possible to explain the physical effects you see.

2.1.2 How do solar cells absorb light?

In this section we are going to learn how a solar cells interact with light. Firstly, let's have a look at the solar spectrum. Sunlight contains many wavelengths of light, from ultraviolet light, though to visible light to infrared. The human eye can only see a small fraction of the light emitted by the sun. Gpvdm stores a copy of the suns spectrum to perform the simulations. Let's have a look at this spectrum, to do this go to the *Database* tab, the choose *Optical database*. This should, bring up a window as shown in figure 2.7

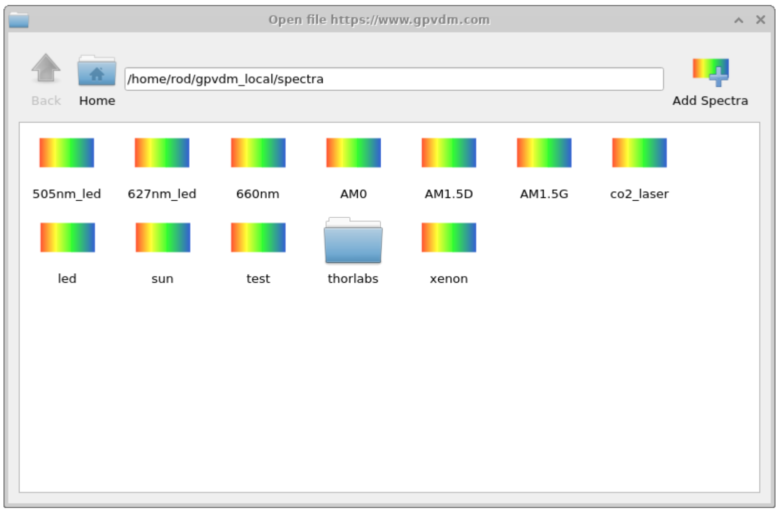


Figure 2.7: The optical database viewer

Double click on the icon called, *AM1.5G*, this should bring up a spectrum of the sun's spectrum. Have a look at where the peak of the spectrum is. Now close this window, and open the spectrum called *led*. Where is the peak of this spectrum.

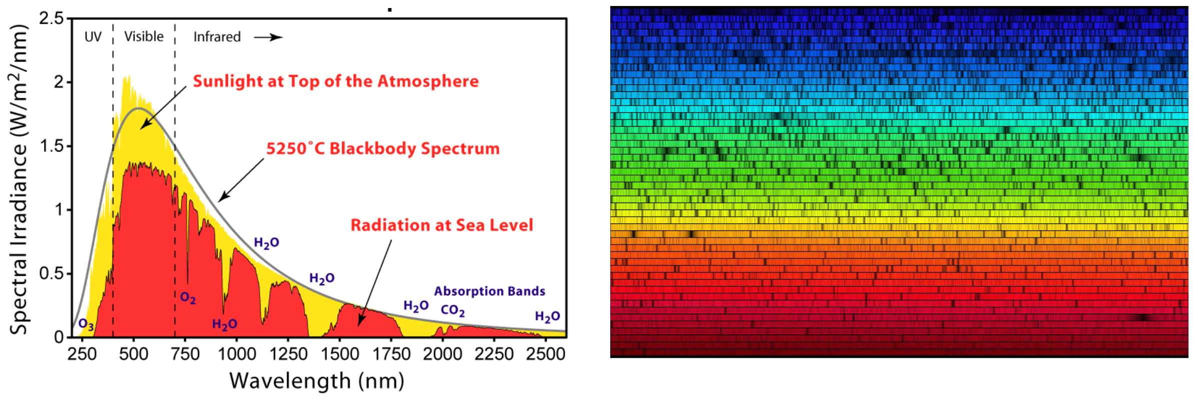


Figure 2.8: a: A plot of the entire solar spectrum. b: The image below shows the solar spectrum at 392 nm (blue) to 692 nm (red) as observed with the Fourier Transform Spectrograph at Kitt Peak National Observatory in 1981. R. Kurucz

Question 3: Describe the main differences between the light which comes from the LED and the sun. Rather than referring to the various regions of the spectrum by their wavelengths, refer to them using English words, such as *infrared*, *UltraViolet*, *Red*, and *Green* etc... you will find which wavelengths match to each color on the internet. If you were designing a material for a solar cell, what wavelengths would.

2.1.3 Light inside solar cells

As you will have seen from when you fist opened the simulation, the solar cells are often made from many layers of different materials. Some of these materials, are designed to absorb light, some are designed to conduct charge carriers out of the cell. The simulator has a database of these materials, to look at the database, click on the *Database* tab, the click on *Materialdatabase*. This should bring up a window as shown in figure 2.9, once this is open navigate to the directory *polymers*, and double click on the material *p3ht*, in the new window click on the tab *Absorption* (see figure 2.10). This plot shows how light is absorbed in the material as a function of wavelength.

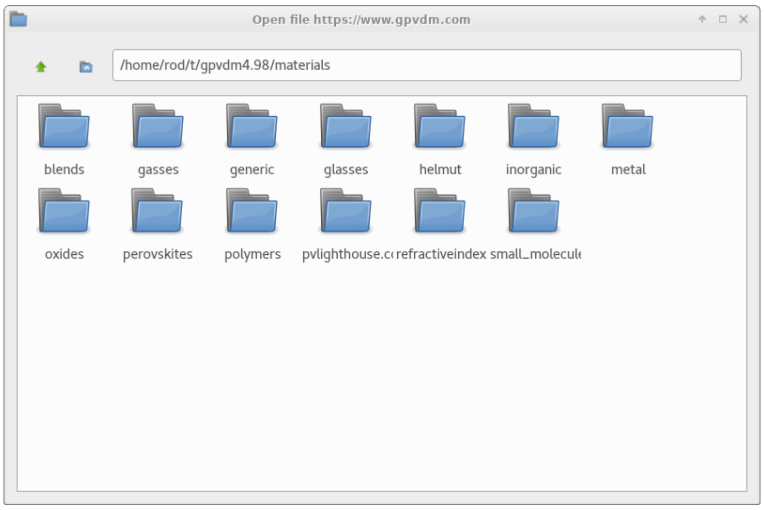


Figure 2.9: The materials database

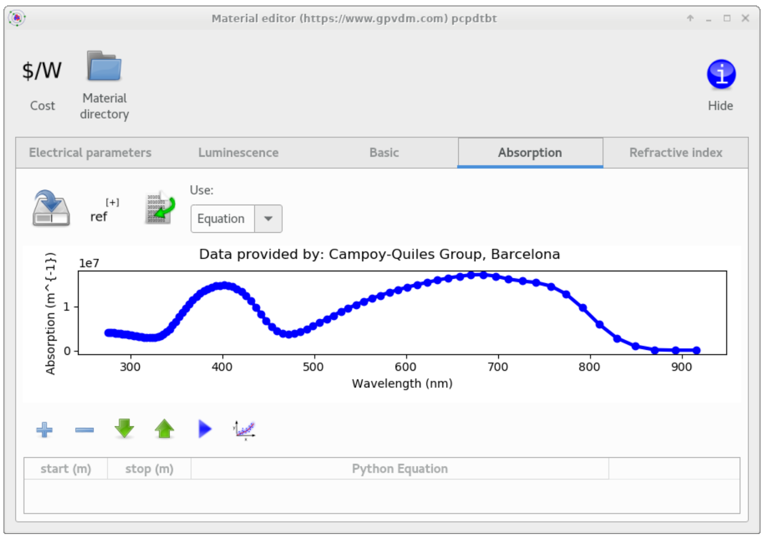


Figure 2.10: Optical absorption of the light.

Question 4: What color of light does the polymer *p3ht* absorb best? Which material in the *polymers* directory do you think will absorb the suns light best?

2.1.4 Parasitic elements

Many devices have parasitic shunt and series resistances associated with them. Shunt resistances (R_s) are caused by conduction straight through the device in thin novel devices this is often caused by impurities in the material system. Parasitic series resistances (R_s) are often associated with the resistance of the contacts, the resistance of the HTL/ETL or any other resistances which are not associated with the active layer. These resistance can be seen for a typical solar cell in figure 2.11 also shown in the figure is the ideal diode of the device. These resistances can be set in the parasitic component window shown in figure 2.12

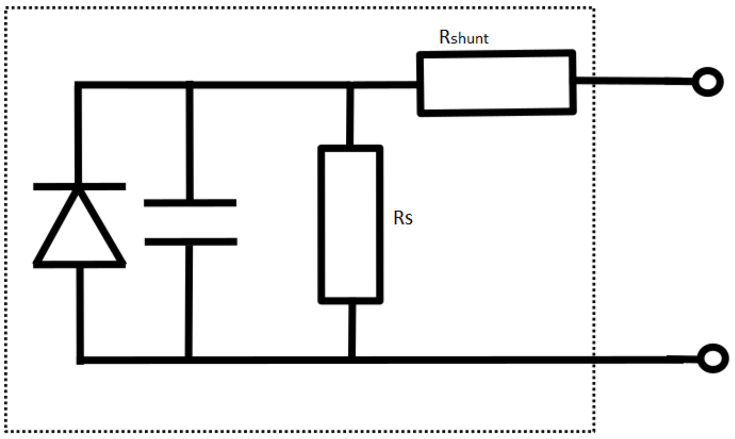
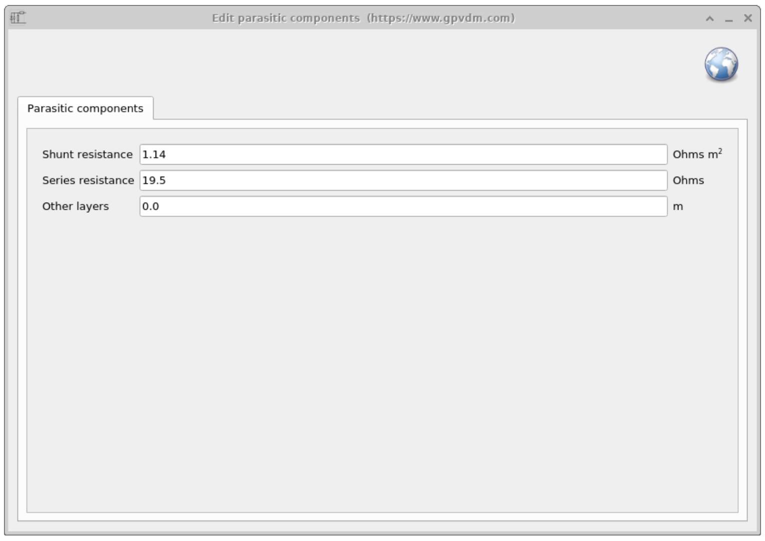


Figure 2.11: Circuit model of a solar cell.

You can change the values of series and shunt resistance in gpvdm, by going to the *Device structure* tab and then clicking on the *Parasitic components* button.



Shunt resistance 1.14 Ohms m^2
Series resistance 19.5 Ohms
Other layers 0.0 m

Figure 2.12: The parasitic component editor

Due to the flat broad contacts on a solar cell, there is often a capacitance associated with the device, this is important for transient measurements and can be calculated with the equation:

$$C = \frac{\epsilon_r \epsilon_0 A}{d + \Delta} \tag{2.1}$$

where A is the area of the device ϵ are the hyperactivities, and d is the thickness of the device. Often for various reasons the measured capacitance of the device does not match what one would expect from the above equation. Therefore the term ”Other layers” (Δ) has been added to the parasitic window to account for differences between measured capacitance and layer measured layer thicknesses.

Task 5: In the optical tab you will find a control called *Light intensity*, this controls the amount of light which falls on the device in Suns. Set it to zero so that the device is in the dark. Then run two JV curve simulations, one with a shunt resistance of 1 $Ohm\ m^2$ and one with a shunt resistance of $1x10^6$ $Ohm\ m^2$ (Hint you will have to enter $1e6$ in the text box). What happens to the dark JV curve? Now try running the same same simulations again but in the light.

2.1.5 Solar cells in the dark

So far, all the simulations we have run have been performed in the light. This is a logical, as usually we are interested in solar cell performance only in the light. However, a lot of interesting information can be gained about solar cells by studying their performance in the dark. We are now going to turn off the light in the simulation. From the 'Home' tab set the light intensity to 0.0 Suns. The photons in the 3D image should disappear as seen in figure 2.13.

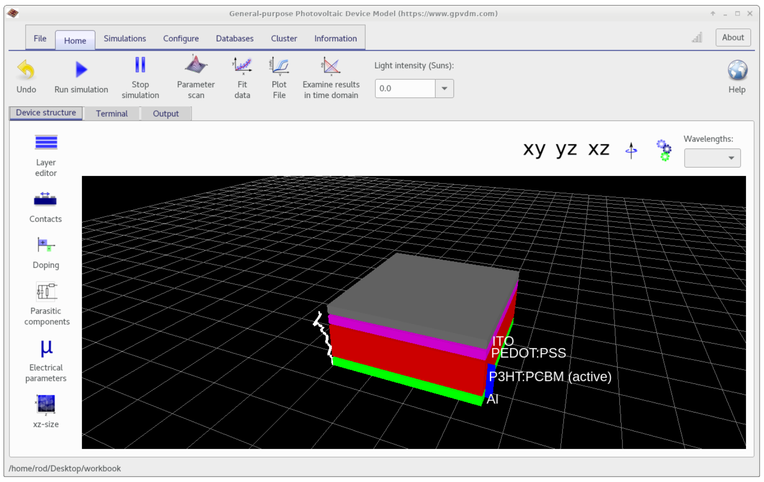


Figure 2.13: Running gpvdm in the dark.

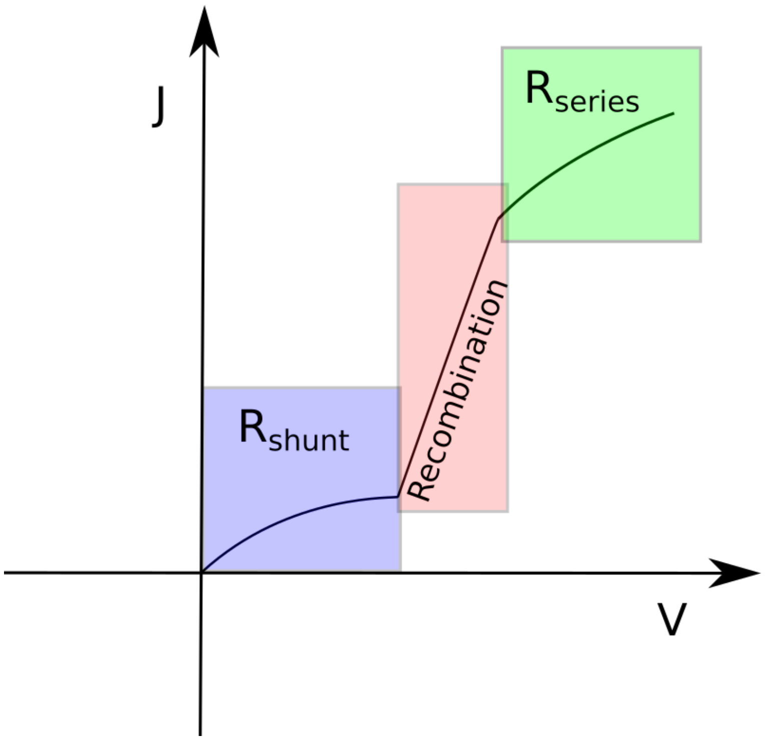


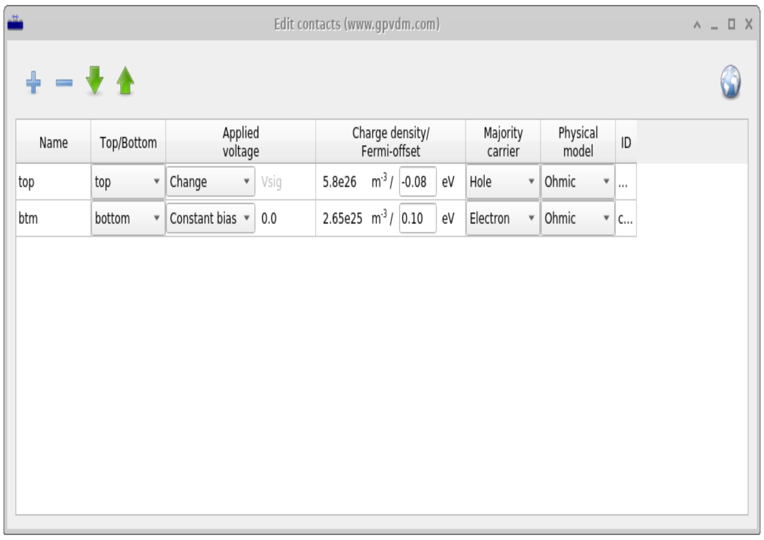
Figure 2.14: A sketch of a typical dark JV curve.

Now set the shunt resistance to $1M\Omega$, and run a simulation. Plot the jv curve. It is customary to plot jv curves on a x-linear y-log scale. To do this in the plot window, hit the 'l' key to do this. The shape should resemble, the JV curve in figure 2.14. Certain solar cell parameters affect different parts of the dark JV curve differently, the lower region is affected very strongly by shunt resistance, the middle part is affected strongly by recombination, and the upper part is strongly affected by the series resistance.

Question 6: What values of series and shunt resistance, would produce the best possible solar cell? Enter these values into the device simulator and copy and paste the dark JV curve into your report.

2.1.6 The contact editor

The contact editor is used to configure the electrical contacts. Which layers act as contacts is configured in the layer editor see section
refsec:layereditor The contact editor has the following fields:



Name	Top/Bottom	Applied voltage		Charge density/ Fermi-offset				Majority carrier	Physical model	ID
top	top	Change	Vsig	5.8e26	m^{-3} /	-0.08	eV	Hole	Ohmic	...
btm	bottom	Constant bias	0.0	2.65e25	m^{-3} /	0.10	eV	Electron	Ohmic	c...

Figure 2.15: The contact editor

- Name: The name of the contact, this can be any English word. It has no physical meaning.

- Top/Bottom: Sets if the contact is on the top, bottom or in 2D simulation left and right of the device are also valid.

- Applied voltage: Sets the applied voltage on the contact. You first have to select what type of applied voltage you want:

  - Ground: This will set the contact to zero volts i.e. ground. 0V is always taken as ground.
  - Constant bias: This will apply a constant bias to a contact. It can be set to zero, and would then be equivalent to ground. In OFET simulations the voltage value can be set to bias one contact to a desired constant voltage.
  - Change: If a contact is set to 'Change' this tells the simulation to apply a changing voltage to this contact. For example if you are performing a JV sweep, the sweep voltage will be applied to this contact. Similarly if you are doing an IS simulation (TPV, TPC, ToF etc..) the voltage will be applied/measured to this contact.

- Charge density: This sets the majority charge density on the contacts. The Fermi-offset is calculated from the charge density. The model does not use Fermi-offset as an input, it uses charge density.

- Majority carrier: This sets the majority carrier density to electrons or holes.

- Physical model: This selects if you have ohmic contacts or schottky contacts. I recommend using ohmic contacts.

Task 7: For a good contact which results in a high efficiency device, the Fermi-offset will be exactly 0 eV or very small. Firstly set the Fermi-offset to zero for both contacts, and run a simulation. What efficiency cell do you get? Now set the Fermi-offset to $0.3eV$ what efficiency cell do you now have? Make a note of the charge densities on the contacts which these Fermi-offsets produce.

2.1.7 Electrical parameters

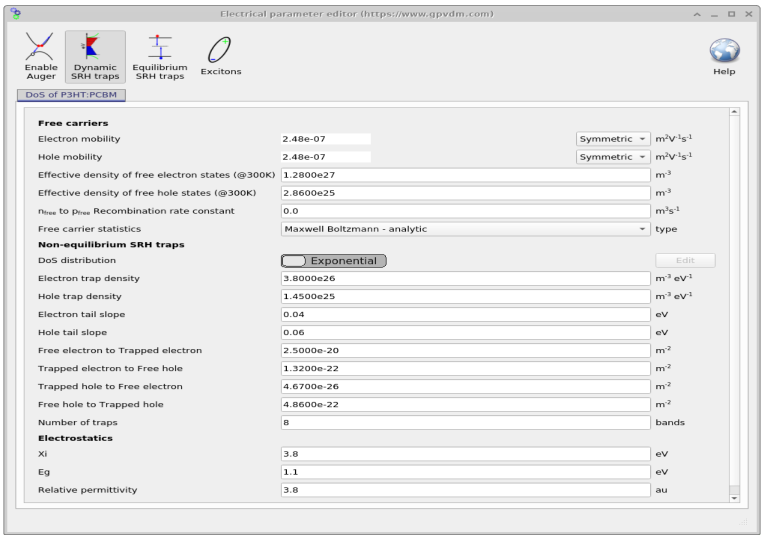
The electrical parameter editor enables you to change the electrical parameters associated with the active layers. Here you can change mobilities, trap constants etc. The toolbar at the top of the window allows you to turn off and on various electrical mechanisms including:

- Auger recombination: This switches on and off Auger recombination. See 16.5.4 for more information.

- Dynamic SRH traps: This is used to turn on and off dynamic SRH traps. See section 16.5 for more information.

- Equilibrium SRH traps: This can be used to introduce a single equilibrium trap level. See section 16.5 for more information.

- Excitons: This enables the excition diffusion equation to be solved along with the electrical equations. See section 8.2 for more information.



Free carriers		
Electron mobility	2.48e-07 [Symmetric]	$m^2V^{-1}s^{-1}$
Hole mobility	2.48e-07 [Symmetric]	$m^2V^{-1}s^{-1}$
Effective density of free electron states (@300K)	1.2800e27	m^{-3}
Effective density of free hole states (@300K)	2.8600e25	m^{-3}
n_{free} to p_{free} Recombination rate constant	0.0	m^3s^{-1}
Free carrier statistics	Maxwell Boltzmann - analytic	type
Non-equilibrium SRH traps		
DoS distribution	Exponential	
Electron trap density	3.8000e26	m^{-3} eV^{-1}
Hole trap density	1.4500e25	m^{-3} eV^{-1}
Electron tail slope	0.04	eV
Hole tail slope	0.06	eV
Free electron to Trapped electron	2.5000e-20	m^{-2}
Trapped electron to Free hole	1.3200e-22	m^{-2}
Trapped hole to Free electron	4.6700e-26	m^{-2}
Free hole to Trapped hole	4.8600e-22	m^{-2}
Number of traps	8	bands
Electrostatics		
Xi	3.8	eV
Eg	1.1	eV
Relative permittivity	3.8	au

Figure 2.16: Electrical parameter window

Task 8: The values of electron mobility dictate how easily charge can move in the device. You can think of this value as akin to resistance or a sort of microscopic resistance. Try try increasing the mobilities by two orders of magnitude and look what happens to the light JV curve of the device and the efficiency, FF, V_{oc} and J_{sc} Do you think it is good to have a low or high value of mobility?

Task 9: Recombination is described later in detail but for now we can simply think of it as how many electrons and holes meet each other in a given time. As stated above there are various types of recombination which can happen in organic semiconductors, but for now we will *just consider* the case when a free electron meets a free hole. This is sometimes called bi-molecular recombination, the equation for this is given by:

$$R(x) = kn(x)p(x) \tag{2.2}$$

Where $n(x)$ is the density of electrons and $p(x)$ is the density of holes, and k is a rate constant. Before trying to understand this rate, firstly turn off the more complex SRH recombination by clicking on the *Dynamic SRH traps* in figure 2.16. You will notice lots of text boxes disappear. Then try changing the value of k which is set in the text box called n_{free} to p_{free} Recombination rate constant, from 1e-15 to 1e-20 in five steps. Run a simulation each time you change the value and make a graph of the efficiency of the cell as you change the value.

Chapter 3

Simulation modes and simulation editors

Gpvdm has plugins which enable the base simulation code to perform a variety of simulation types. For example there is a plugin to perform steady state JV simulations, another plugin to perform frequency domain simulations, and another to calculate the Quantum Efficiency etc... In the simulation editors ribbon (see figure 3.1) you can see the icons for each plugin. By clicking on an icon in this ribbon you will be able to edit how the plugin performs the various simulation types. For example under the JV plugin editor one can change the start/stop voltages of a voltage sweep. The simulation mode ribbon shown in figure 3.2 enables the user to select which simulation mode is run. You can see in the image that the *JV curve* icon is depressed which means when the user clicks run, a JV curve will be simulated.

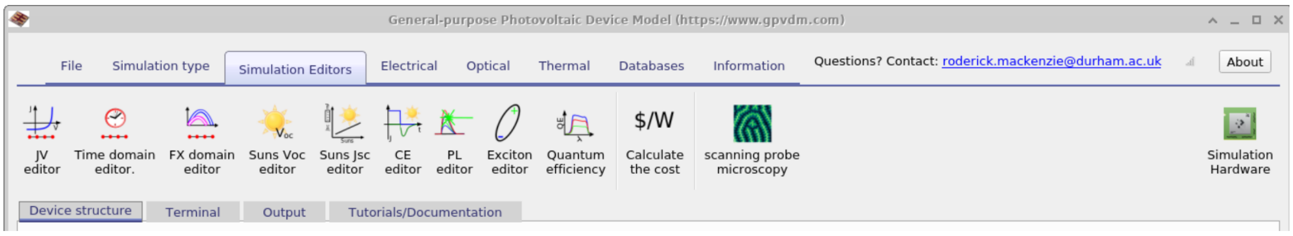


Figure 3.1: Simulation editors

Each plugin editor can generate multiple simulation modes. For example the time domain simulation editor (see figure 3.1) is responsible for the generating both the *CELIV* and the *TPC 400mV DARK* icon in Figure 3.2. This enables the user to define new types of simulation modes at will. For example there may be no TDCF simulation setup, but using the time domain editor you can set one up. Below the various simulation editors are described in detail.

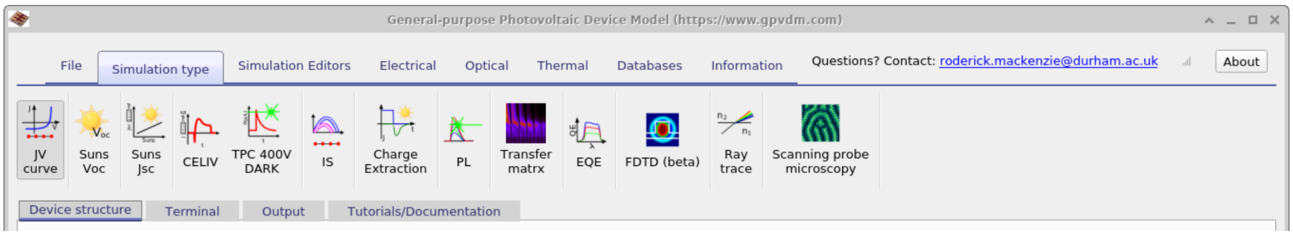


Figure 3.2: Selectig a simulation mode

3.1 JV editor (Steady state simulation editor)

If you click on the JV editor icon in figure 3.3, the JV editor window will open shown below in figure 3.4.

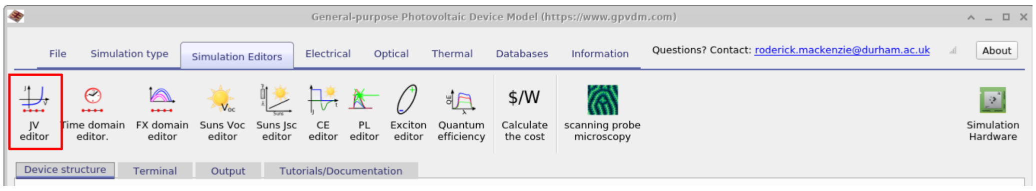
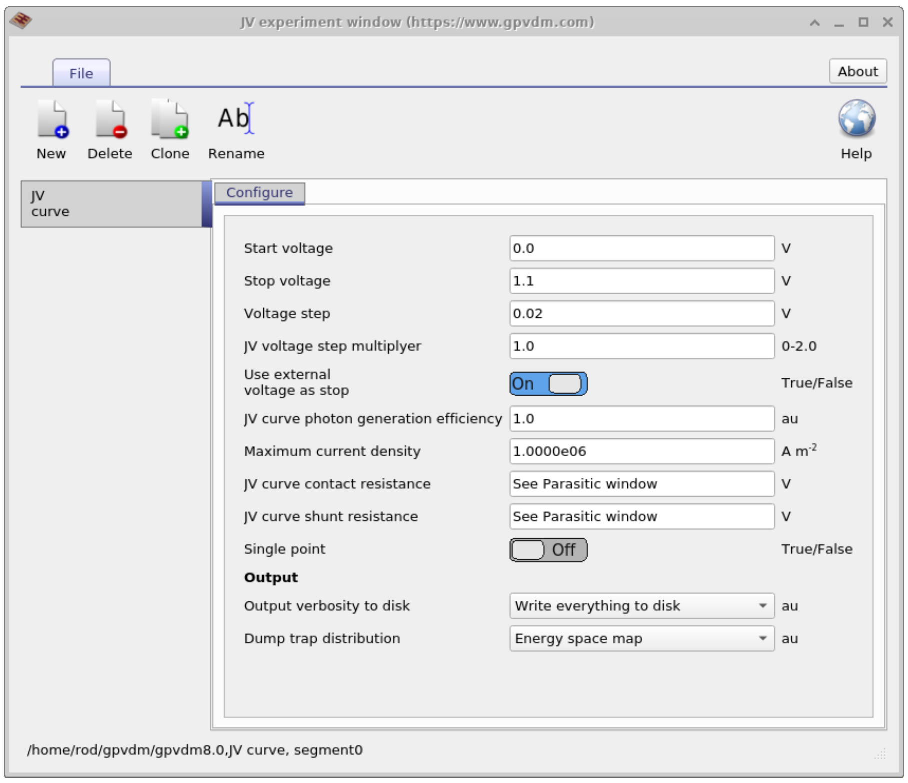


Figure 3.3: Opening the JV editor from the simulation editor ribbon.

This window can be used to configure steady state simulations. It does not matter if you are running a current-voltage sweep on a solar cell or an OFET. This plugin will steadily ramp the voltage from a start voltage to a stop voltage. The voltage will be applied to the *active* contact as defined in the *contact editor*. You can set the start voltage, stop voltage and step size. Use *JV voltage step multiplayer* to make the voltage step grow each step. The default is 1.0, i.e. no growth.



Start voltage	0.0	V
Stop voltage	1.1	V
Voltage step	0.02	V
JV voltage step multiplyer	1.0	0-2.0
Use external voltage as stop	On	True/False
JV curve photon generation efficiency	1.0	au
Maximum current density	1.0000e06	A m^{-2}
JV curve contact resistance	See Parasitic window	V
JV curve shunt resistance	See Parasitic window	V
Single point	Off	True/False
Output		
Output verbosity to disk	Write everything to disk	au
Dump trap distribution	Energy space map	au

/home/rod/gpvdm/gpvdm8.0.JV curve, segment0

Figure 3.4: The JV editor editor window, use this to configure steady state simulations.

The files produced by the JV simulation mode are given in table 3.1.

File name	Description
$jv.dat$	Current voltage curve
$charge.dat$	voltage charge density
$k.csv$	Recombination constant k
$sim\_info.dat$	Calculated V_{oc}, J_{sc} etc.. see 15.6.1

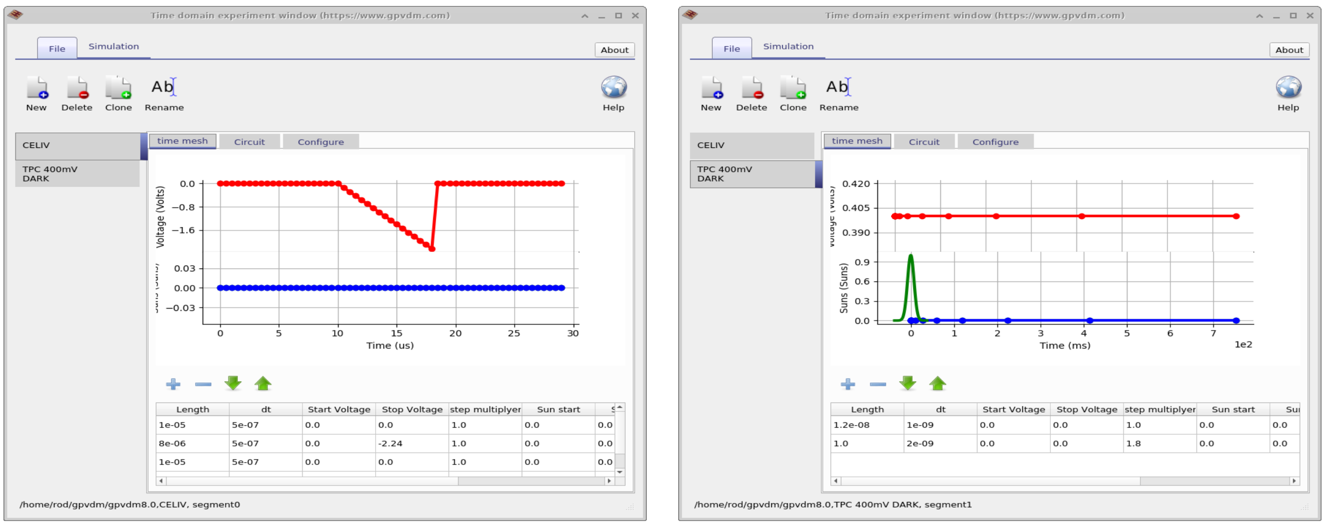
Table 3.1: Files produced by the JV simulation

3.2 Time domain editor

Related YouTube videos:

Simulating optoelectronic sensors made from polymers.

The time domain editor can be used to configure time domain simulations, this is shown in figure 3.5. You can see that one simulation editor can be used to edit multiple simulations. The panel on the left shows the editor being used to edit a CELIV simulation while the panel on the right shows the editor being used to edit a TPC simulation. The new, delete and clone buttons in the top of the window can be used to make new simulation modes. The table in the bottom of the window can be used to setup the time domain mesh, apply voltages or light pulses.



Length	dt	Start Voltage	Stop Voltage	step multiplyer	Sun start	S
1e-05	5e-07	0.0	0.0	1.0	0.0	0.0
8e-06	5e-07	0.0	-2.24	1.0	0.0	0.0
1e-05	5e-07	0.0	0.0	1.0	0.0	0.0

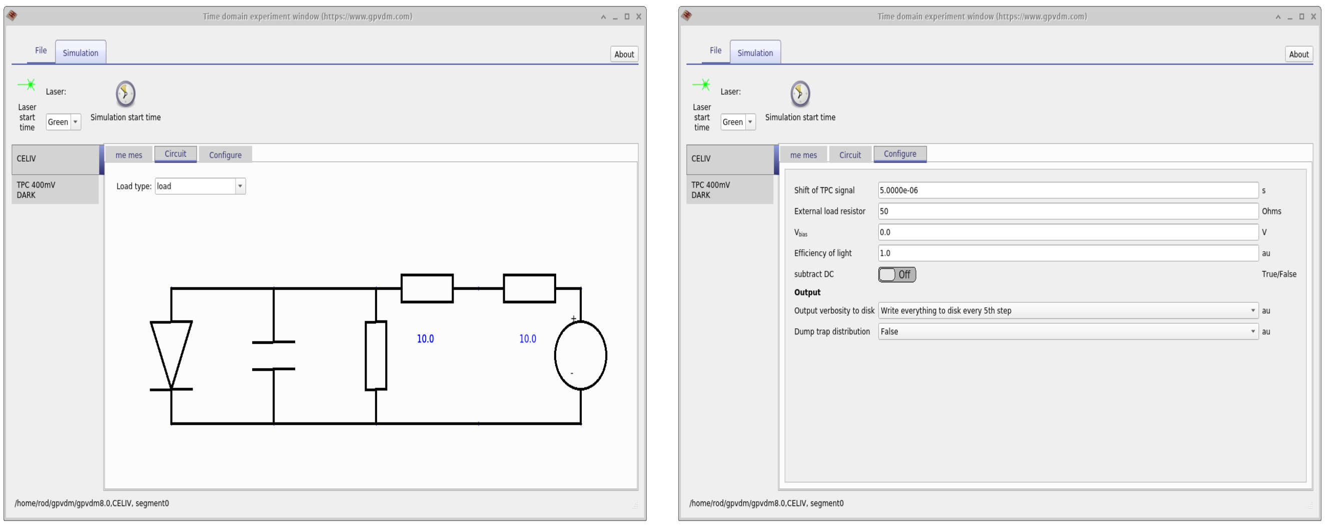
/home/rod/gpvdm/gpvdm8.0,CELIV, segment0

Length	dt	Start Voltage	Stop Voltage	step multiplyer	Sun start	Su
1.2e-08	1e-09	0.0	0.0	1.0	0.0	0.0
1.0	2e-09	0.0	0.0	1.8	0.0	0.0

/home/rod/gpvdm/gpvdm8.0,TPC 400mV DARK, segment1

Figure 3.5: The time domain editor showing the user editing the duration of light/voltage pulses.

Figure 3.6 shows different tabs in of the time domain editor. The image on the left shows the circuit diagram used to model the CELIV experiment. From the left of the image the diode on the left accounts for the drift diffusion simulations it is in effect a perfect diode. Then comes a capacitor used to model the charge on the plates of the device, then a shunt resistance and then the series resistance. The final resistor on the right represents the external resistance of the measuring equipment. The right hand figure shows the configuration options of the time domain window.



Laser: Green

Load type: load

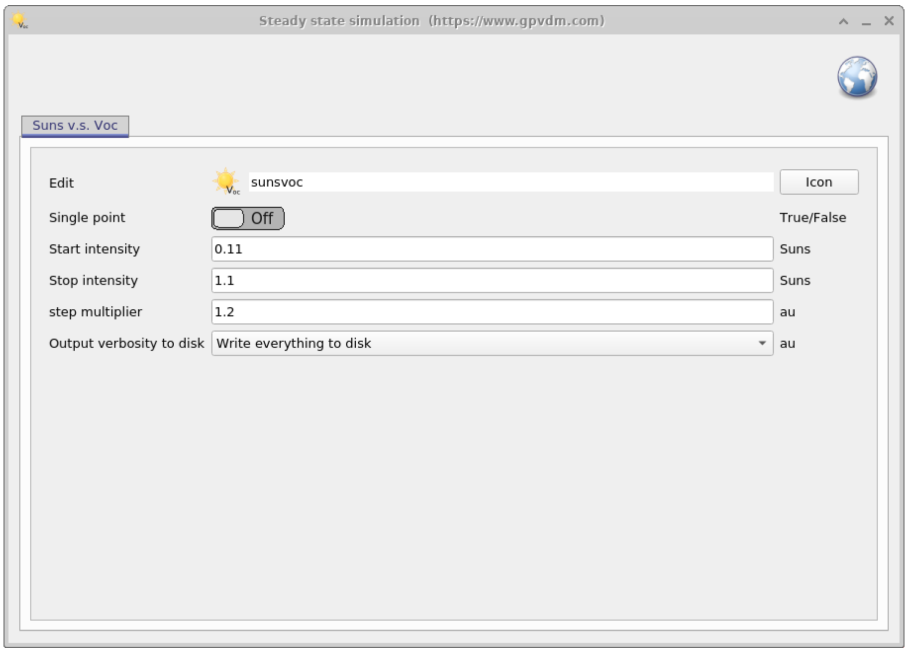
Shift of TPC signal	5.0000e-06	s
External load resistor	50	Ohms
V_{bias}	0.0	V
Efficiency of light	1.0	au
subtract DC	Off	True/False
Output		
Output verbosity to disk	Write everything to disk every 5th step	au
Dump trap distribution	False	au

/home/rod/gpvdm/gpvdm8.0,CELIV, segment0

Figure 3.6: Configuring the time domain editor

3.3 Suns-Voc editor

The Suns-Voc plugin can be used to calculate how open circuit voltage changes as a function of light intensity. This can be useful for understanding tail slope and disorder in devices. A picture of the suns-voc editor window can be seen below in figure 3.7. The window can be used to set the start and stop light intensity.

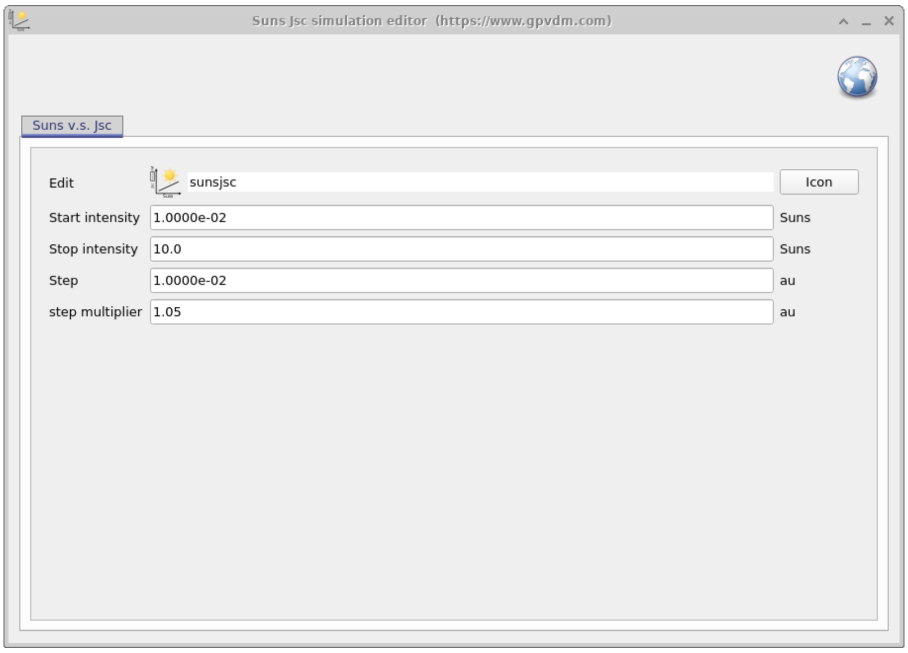


Edit	sunsvoc	
Single point	Off	True/False
Start intensity	0.11	Suns
Stop intensity	1.1	Suns
step multiplier	1.2	au
Output verbosity to disk	Write everything to disk	au

Figure 3.7: The suns-voc editor window

3.4 Suns-Jsc editor

The Jsc editor can be used to configure suns-Jsc simulations. It enables you to set the start light intensity, stop light intensity and how big the steps are. This is shown in figure 3.8.



Edit	sunsjsc	
Start intensity	1.0000e-02	Suns
Stop intensity	10.0	Suns
Step	1.0000e-02	au
step multiplier	1.05	au

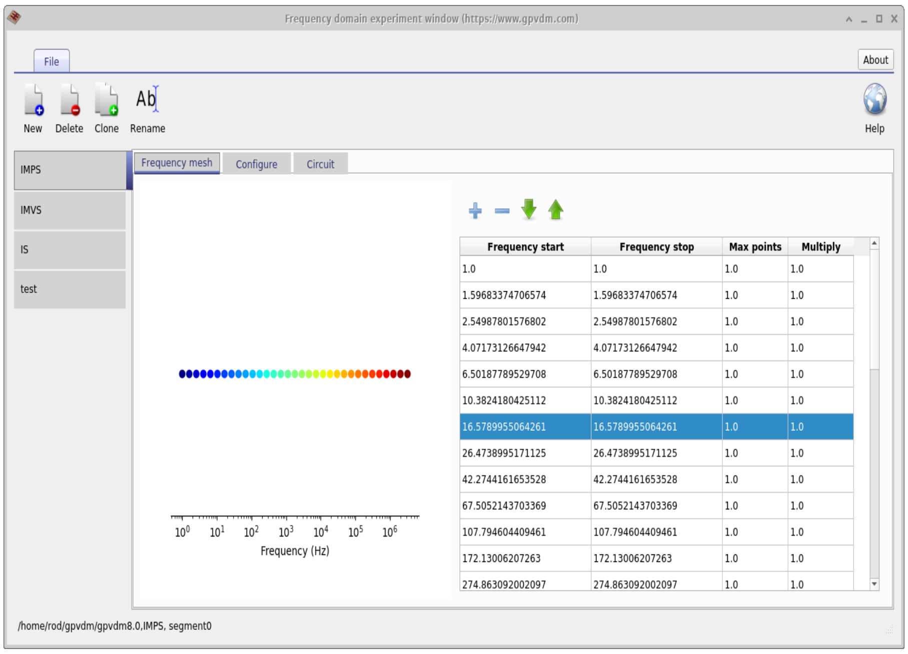
Figure 3.8: The JV curve editor window

3.5 Frequency domain editor

The frequency domain editor allows you to configure frequency domain simulations. This is shown in figure 3.9. On the right of the window you can see the frequency points which will be simulated. These are also displayed on the graph to the right of the window. Notice that in the figure there are multiple simulations configured, IMPS, IMVS, IS and "test". Each of these simulations will appear as an separate simulation mode in the simulation mode ribbon.

Related YouTube videos:

Simulating impedance spectroscopy (IS) in solar cells.



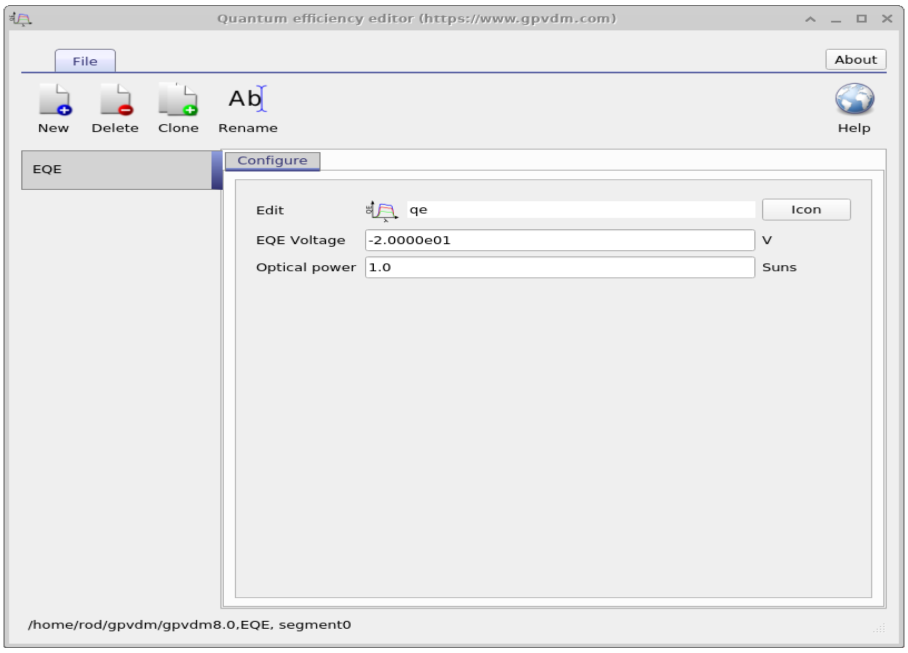
Frequency start	Frequency stop	Max points	Multiply
1.0	1.0	1.0	1.0
1.59683374706574	1.59683374706574	1.0	1.0
2.54987801576802	2.54987801576802	1.0	1.0
4.07173126647942	4.07173126647942	1.0	1.0
6.50187789529708	6.50187789529708	1.0	1.0
10.3824180425112	10.3824180425112	1.0	1.0
16.5789955064261	16.5789955064261	1.0	1.0
26.4738995171125	26.4738995171125	1.0	1.0
42.2744161653528	42.2744161653528	1.0	1.0
67.5052143703369	67.5052143703369	1.0	1.0
107.794604409461	107.794604409461	1.0	1.0
172.13006207263	172.13006207263	1.0	1.0
274.863092002097	274.863092002097	1.0	1.0

/home/rod/gpvdm/gpvdm8.0,IMPS, segment0

Figure 3.9: The frequency domain editor window

3.6 Quantum efficiency editor

The quantum efficiency editor simulates both EQE and IQE. The configuration window can be used to set the voltage at which EQE and IQE are performed.



Edit qe

EQE Voltage -2.0000e01 V

Optical power 1.0 Suns

/home/rod/gpvdm/gpvdm8.0,EQE, segment0

Figure 3.10: The quantum efficiency editor window

Chapter 4

2D simulations - OFETs

Tutorial on OFET simulation.

Gpvdm contains a 2D electrical solver which can be used for simulating OFETs and other 2D structures. To perform 2D simulations use the default OFET simulation in gpvdm as a starting point. You can do this by double clicking on *OFET simulation* in the new simulation window (see figure 4.1).

Note: The 2D electrical solver is a separate plug in to the 1D solver, if you select the default OFET simulation gpvdm as a starting point for your own 2D simulations gpvdm will be all set up to do 2D electrical simulations. If you try to convert a 1D simulation such as a solar cell to a 2D simulation (not recommended) please read section 16.9 on how to select the correct solver.

To make a new OFET simulation, click on the new simulation button. In the new simulation window and select the OFET simulation (see figure 4.1). This will bring up the initial OFET simulation shown in figure 4.2.

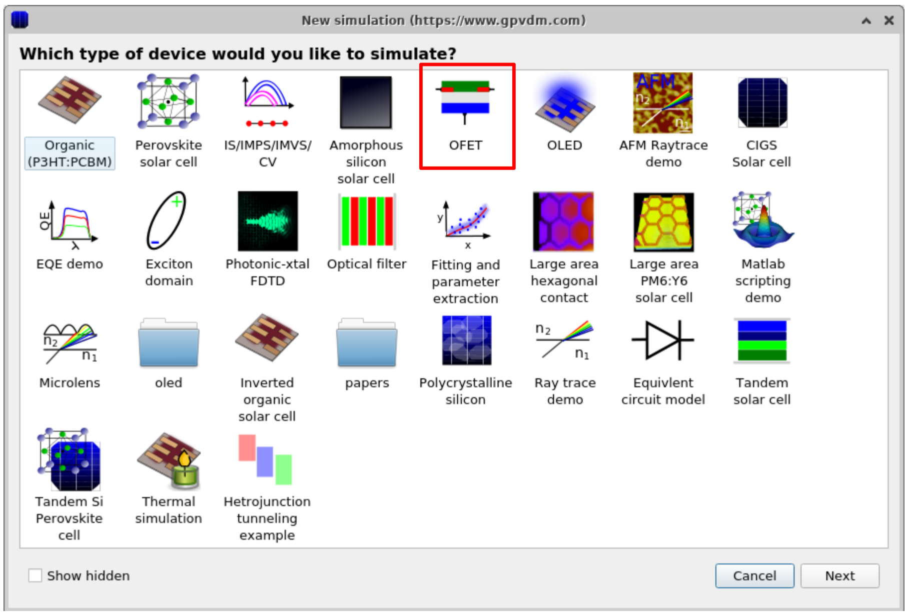
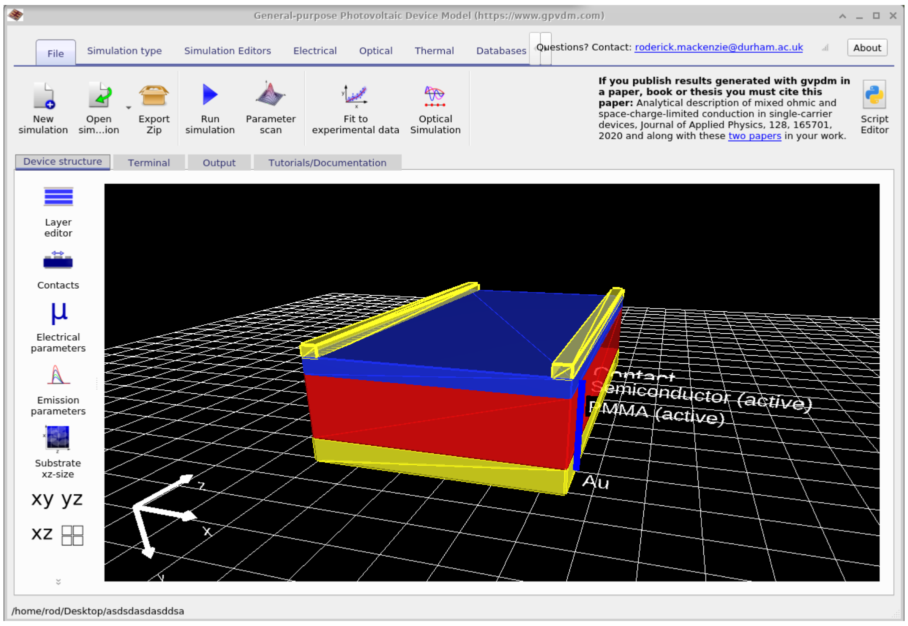


Figure 4.1: Opening a new ofet simulation

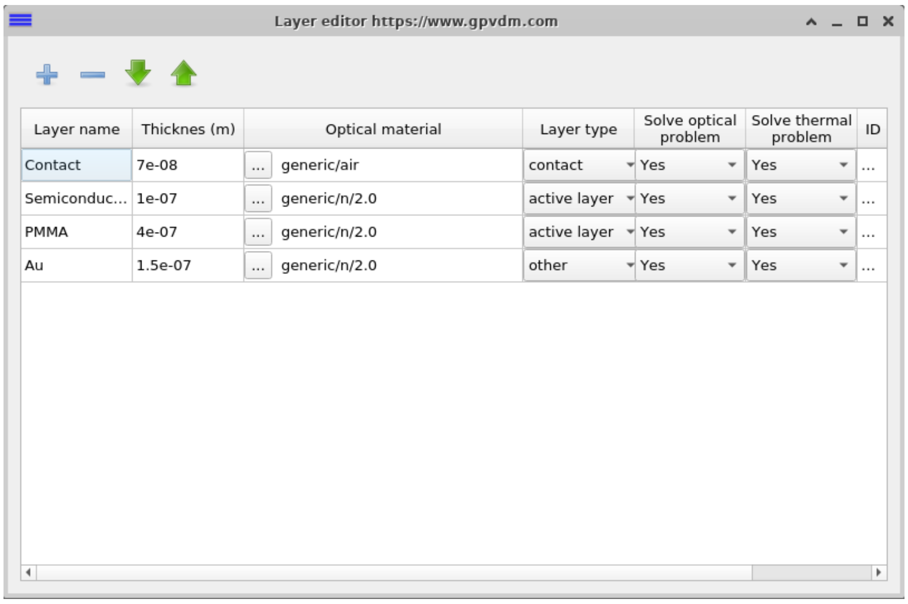
4.0.1 The anatomy of a 2D simulation



If you publish results generated with gvpdm in a paper, book or thesis you must cite this paper: Analytical description of mixed ohmic and space-charge-limited conduction in single-carrier devices, Journal of Applied Physics, 128, 165701, 2020 and along with these two papers in your work.

Figure 4.2: The default ofet simulation.

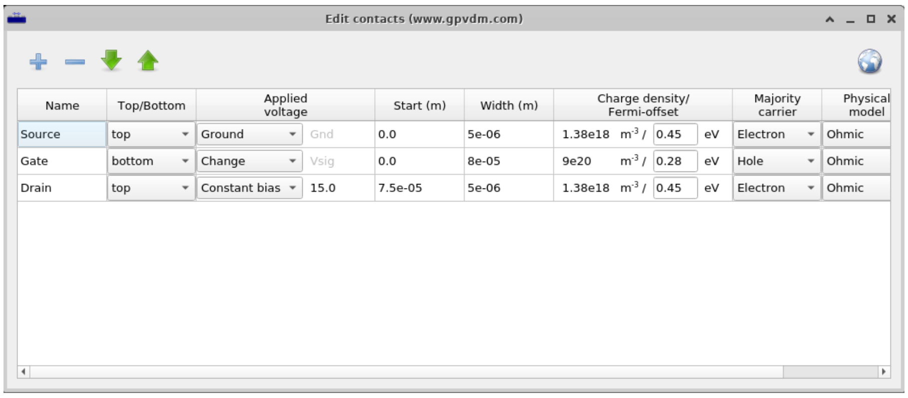
The OFET structure shown in figure 4.2 consists of a *gate* and *drain* contact shown on the top of the simulation as gold bars, a semiconductor layer is shown in blue and an insulating later shown in red. A *gate* contact is also visible at the bottom of the structure. This layer structure is defined in layer editor, see figure 4.3. The layer editor has been described in detail in section 2.1.1. It can be seen that the top and bottom layers have been set to *contact* and the insulator (PMMA) and semiconducting layer have been set to active. This means that the drift diffusion equations will be solved over the semiconductor and insulator layers and the contacts will be used as boundary conditions. As this structure is not emitting light the *Optical material* column has no impact on the simulation results.



Layer name	Thicknes (m)	Optical material	Layer type	Solve optical problem	Solve thermal problem	ID
Contact	7e-08	... generic/air	contact	Yes	Yes	...
Semiconduc...	1e-07	... generic/n/2.0	active layer	Yes	Yes	...
PMMA	4e-07	... generic/n/2.0	active layer	Yes	Yes	...
Au	1.5e-07	... generic/n/2.0	other	Yes	Yes	...

Figure 4.3: The layers of an OFET device

The contacts are defined in the contact editor shown in figure 4.4. The contact editor has been described in detail in section 2.1.6, however because this is a 2D simulation another two extra columns have appeared. They are *start* and *width*. These define the start position of the contact on the x-axis and width which describes the width of the contact on the x-axis. The *source* starts at 0 m and extends to $5\mu m$, the *drain* starts at 75 μm and extends to $5\mu m$, while the gate starts at 0 m and extends to cover the entire width of the device which is 80 μm. If you are unsure which is the x-axis, the origin marker is visible at the bottom of figure 4.2.


Edit contacts (www.gpvdm.com)

Name	Top/Bottom	Applied voltage		Start (m)	Width (m)	Charge density/ Fermi-offset		Majority carrier	Physical model
Source	top	Ground	Gnd	0.0	5e-06	1.38e18 m^{-3} /	0.45 eV	Electron	Ohmic
Gate	bottom	Change	Vsig	0.0	8e-05	9e20 m^{-3} /	0.28 eV	Hole	Ohmic
Drain	top	Constant bias	15.0	7.5e-05	5e-06	1.38e18 m^{-3} /	0.45 eV	Electron	Ohmic

Figure 4.4: Editing the contacts on a 2D device.

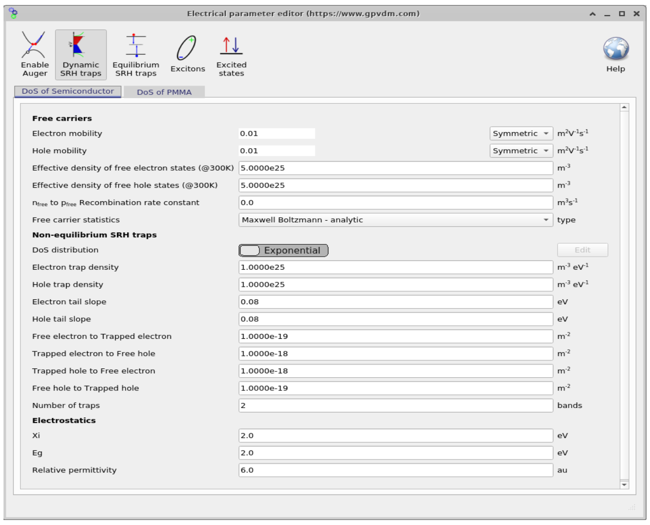
The electrical parameters for both the semiconductor and the insulator can be seen in figure 4.5, these can be accessed through the *Electrical parameter* editor. The *Electrical parameter* editor is described in detail in section 2.1.7. To the left of the figure the parameters for the semiconducting layer are described and to the right of the figure the parameters for the insulating layer are described. Below I have made comments on each parameter in relation to OFET simulation.

- Mobility: The free carrier mobility in the active layer will be defined by the semiconductor you are trying to simulate. The mobility in the insulator is usually set to a low value such as $1e-12-1e-15$ to limit current flow into the region. However, the value should not be set too low (see section 16.9.1) or the solver may become numerically unstable.

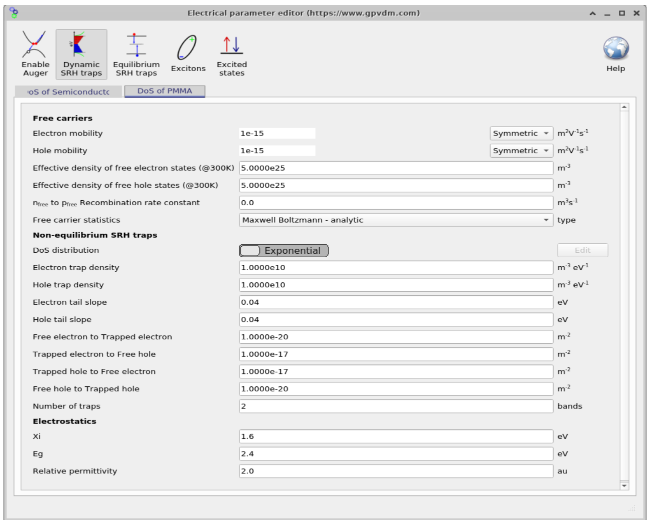
- Effective density of states: Keep these the same for both layers, just to keep things simple.

- Number of trap states: You can set this to what value you want to but if you choose a large number in 2D it will significantly affect the speed of the simulation, simply because *number of equation solved=XMESHPOINTS* $\times$ *YMESHPOINTS* $\times$ *NUMEROFTRAPS.*

- Eg and Xi: Although it is tempting to simply enter the experimental values for Xi and Eg for both the insulator and the semiconductor, one has to be careful in doing this as some insulators (SiO_2) have very big band gaps which mean the number of carriers get very small and make the simulation unstable (read section 16.9.1 for an explanation). If you want to simulate a jump in the band gap into an insulator, my is to make the jump significantly bigger than $3/2kT = 25meV$ which is the average kinetic energy of a charge carrier. If the gap is between $0.5 - 1.0V$ charge carriers will have problems penetrating the barrier and there is no need to simulate bigger steps.



Free carriers			
Electron mobility	0.01	Symmetric	$m^2V^{-1}s^{-1}$
Hole mobility	0.01	Symmetric	$m^2V^{-1}s^{-1}$
Effective density of free electron states (@300K)	5.0000e25		m^{-3}
Effective density of free hole states (@300K)	5.0000e25		m^{-3}
n_{free} to p_{free} Recombination rate constant	0.0		m^3s^{-1}
Free carrier statistics	Maxwell Boltzmann - analytic		type
Non-equilibrium SRH traps			
DoS distribution	Exponential		
Electron trap density	1.0000e25		$m^{-3}\ eV^{-1}$
Hole trap density	1.0000e25		$m^{-3}\ eV^{-1}$
Electron tail slope	0.08		eV
Hole tail slope	0.08		eV
Free electron to Trapped electron	1.0000e-19		m^{-2}
Trapped electron to Free hole	1.0000e-18		m^{-2}
Trapped hole to Free electron	1.0000e-18		m^{-2}
Free hole to Trapped hole	1.0000e-19		m^{-2}
Number of traps	2		bands
Electrostatics			
Xi	2.0		eV
Eg	2.0		eV
Relative permittivity	6.0		au



Free carriers			
Electron mobility	1e-15	Symmetric	$m^2V^{-1}s^{-1}$
Hole mobility	1e-15	Symmetric	$m^2V^{-1}s^{-1}$
Effective density of free electron states (@300K)	5.0000e25		m^{-3}
Effective density of free hole states (@300K)	5.0000e25		m^{-3}
n_{free} to p_{free} Recombination rate constant	0.0		m^3s^{-1}
Free carrier statistics	Maxwell Boltzmann - analytic		type
Non-equilibrium SRH traps			
DoS distribution	Exponential		
Electron trap density	1.0000e10		$m^{-3}\ eV^{-1}$
Hole trap density	1.0000e10		$m^{-3}\ eV^{-1}$
Electron tail slope	0.04		eV
Hole tail slope	0.04		eV
Free electron to Trapped electron	1.0000e-20		m^{-2}
Trapped electron to Free hole	1.0000e-17		m^{-2}
Trapped hole to Free electron	1.0000e-17		m^{-2}
Free hole to Trapped hole	1.0000e-20		m^{-2}
Number of traps	2		bands
Electrostatics			
Xi	1.6		eV
Eg	2.4		eV
Relative permittivity	2.0		au

Figure 4.5: Electrical parameters for both the semiconductor (left) and the insulator (right)

4.0.2 Running a 2D simulation

2D simulations are run in the same way as 1D simulations, simply click on the play button, see figure 4.6.

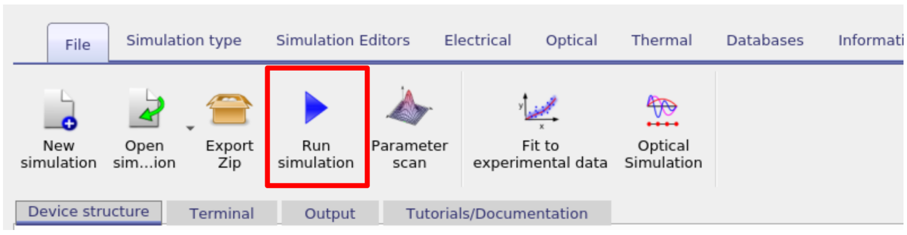
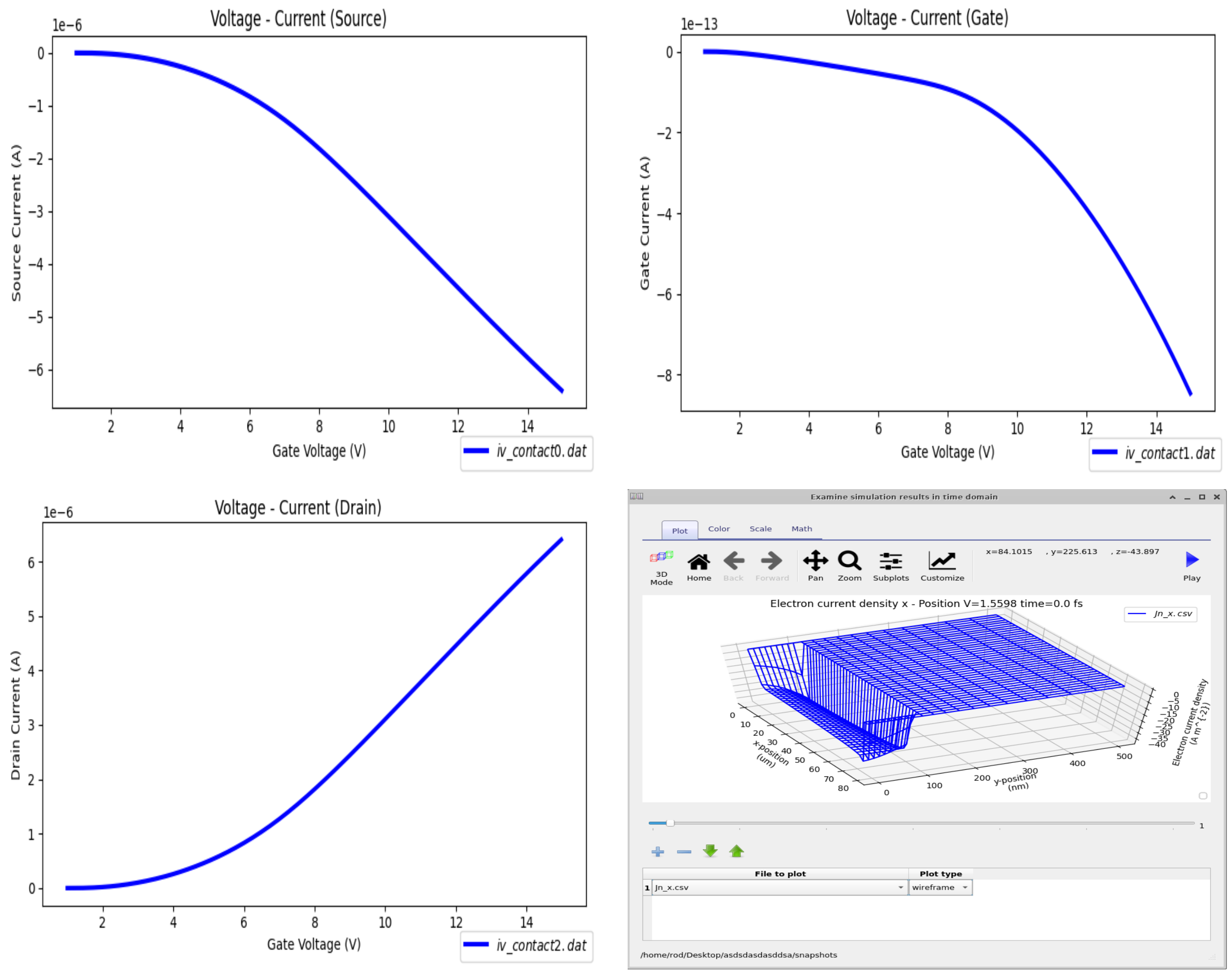


Figure 4.6: Running an OFET simulation

The simulation will take longer than it's 1D counterparts simply because there will be more equations to solve. If you have set a contact at a high starting voltage the solver will initially ramp the contact voltage in a stepwise way until the desired voltage is achieved before the desired voltage sweep is applied to the *active contact*. After the simulation has run the following files will be produced showing the current density from each contact.

- contact_iv0.dat:
- contact_iv1.dat:
- contact_iv2.dat:
- contact_jv0.dat:
- contact_jv1.dat:
- contact_jv2.dat:
- snapshots:

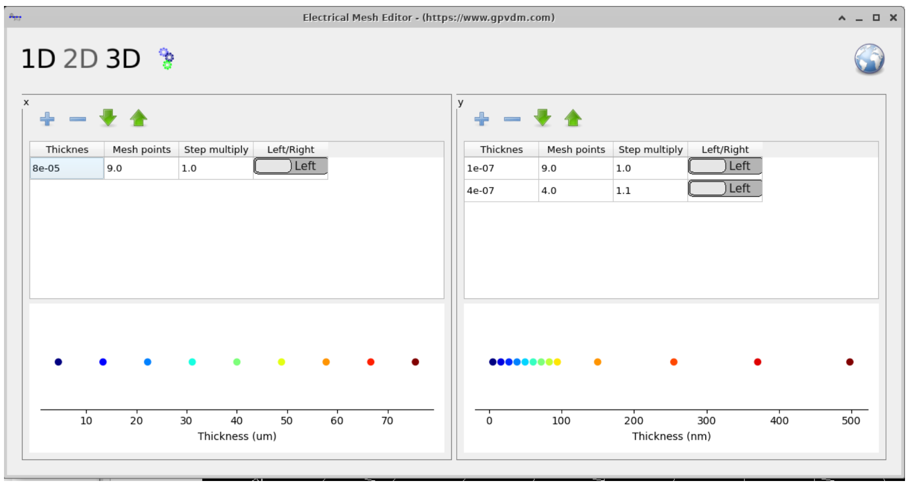


File to plot	Plot type
1 Jn_x.csv	wireframe

/home/rod/Desktop/asdsdasdasddsa/snapshots

Figure 4.7: Electrical parameters

4.0.3 Meshing in 2D



x

Thicknes	Mesh points	Step multiply	Left/Right
8e-05	9.0	1.0	Left

y

Thicknes	Mesh points	Step multiply	Left/Right
1e-07	9.0	1.0	Left
4e-07	4.0	1.1	Left

Figure 4.8: Meshing

Chapter 5

Simple circuit simulations

Chapter 6

Large area device simulation

Gpvdm primarily focuses on drift diffusion modelling of small area device such as solar cells and OFETs. Drift and diffusion simulations are good at describing the microscopic operation of devices. They allow you to understand how carriers, potential and recombination interact on the nanometer scale. However, sometimes one wants to simulate large area devices such as printed substrates spanning over many square centimetres. For this type of simulation one needs to use less detailed and more efficient circuit models, this section describes how to do that.

Related YouTube videos:

Tutorial on designing large area contacts for flexible electronics

Understanding Printed Hexagonal Contacts for Large Area Solar Cells

6.1 Designing contacts for large area devices

A common problem is designing large area contacts for solar cells. This paper [3] gives an overview of such a problem. To start designing large area contacts open the new simulation window in the file ribbon, and select the *Large area hexagonal contact* simulation (see figure 6.1). Once you have opened it you should get a window which looks like figure 6.2. This simulation consists for a hexagonal solver contact printed on top of a PEDOT substrate. We are going to find out how the resistance of this contact varies as a function of position.

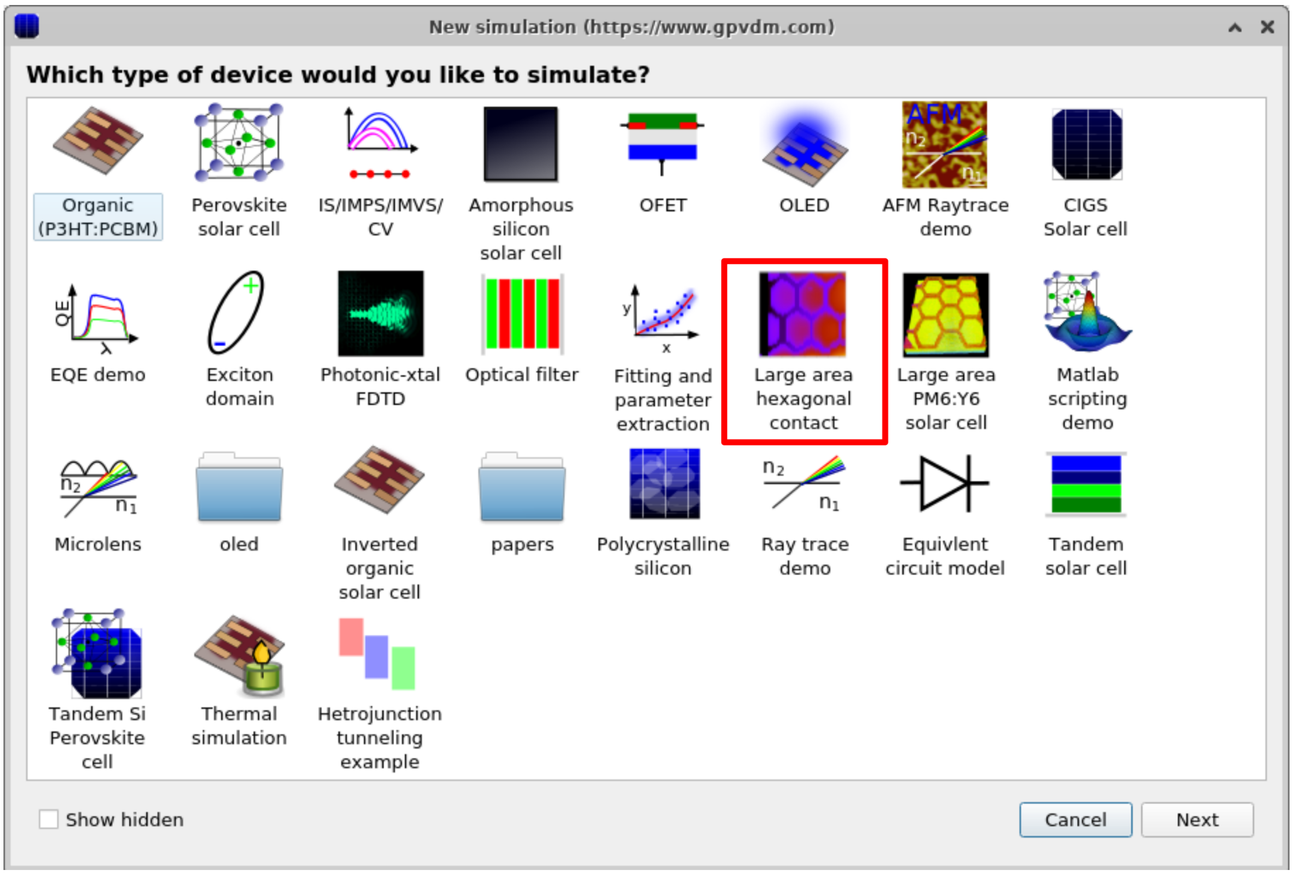


Figure 6.1: Selecting the large area contact simulation

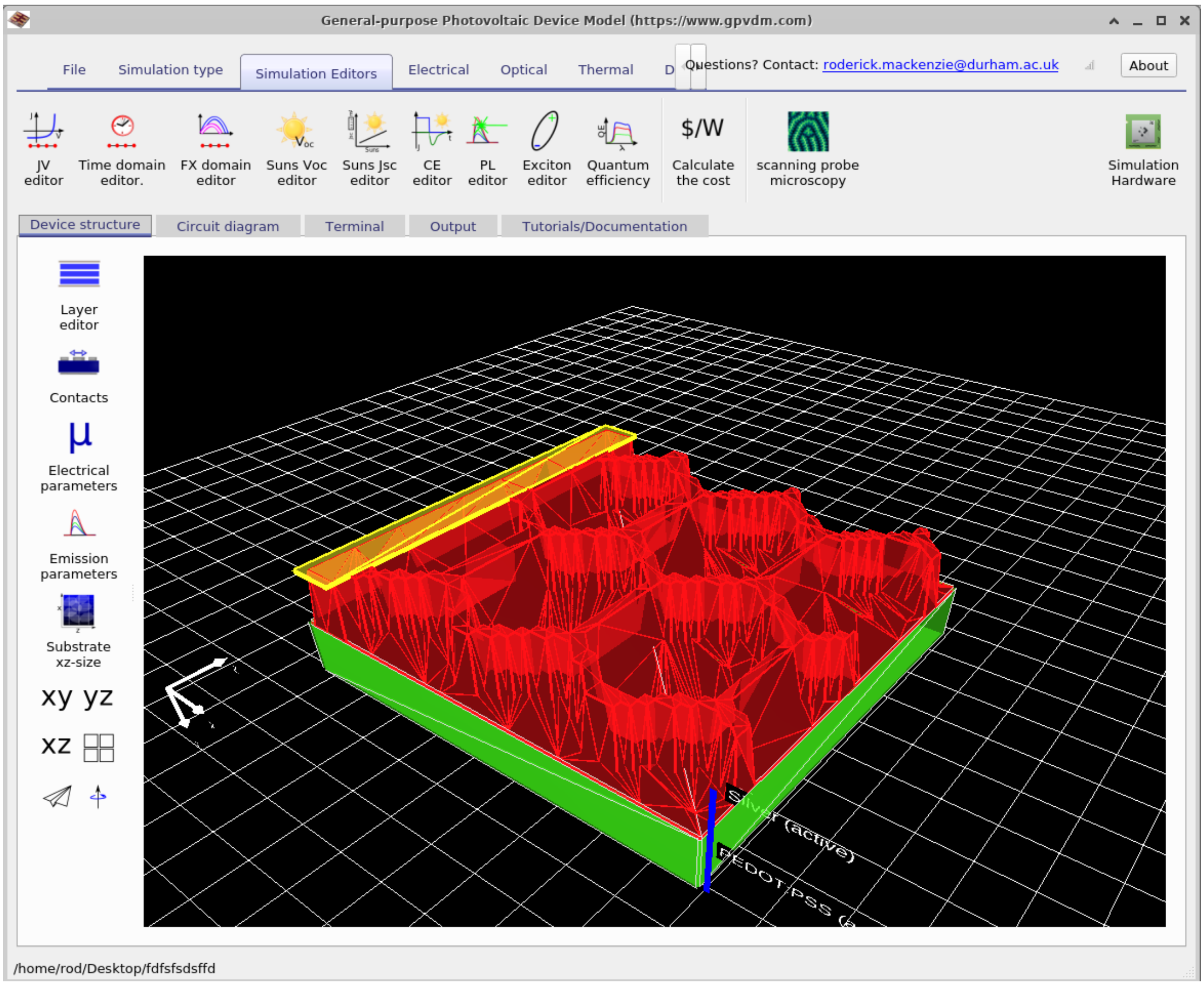


Figure 6.2: A 3D image of the contact printed contact.

The next step in the simulation is to build a network of resistors which approximates the shape of the contact. To do this select the Circuit diagram tab and then click the refresh button. This will build a resistor network of the shape shown in the device structure tab, see figure 6.3. Here you can zoom in and examine the individual resistors, each line represents a resistor.

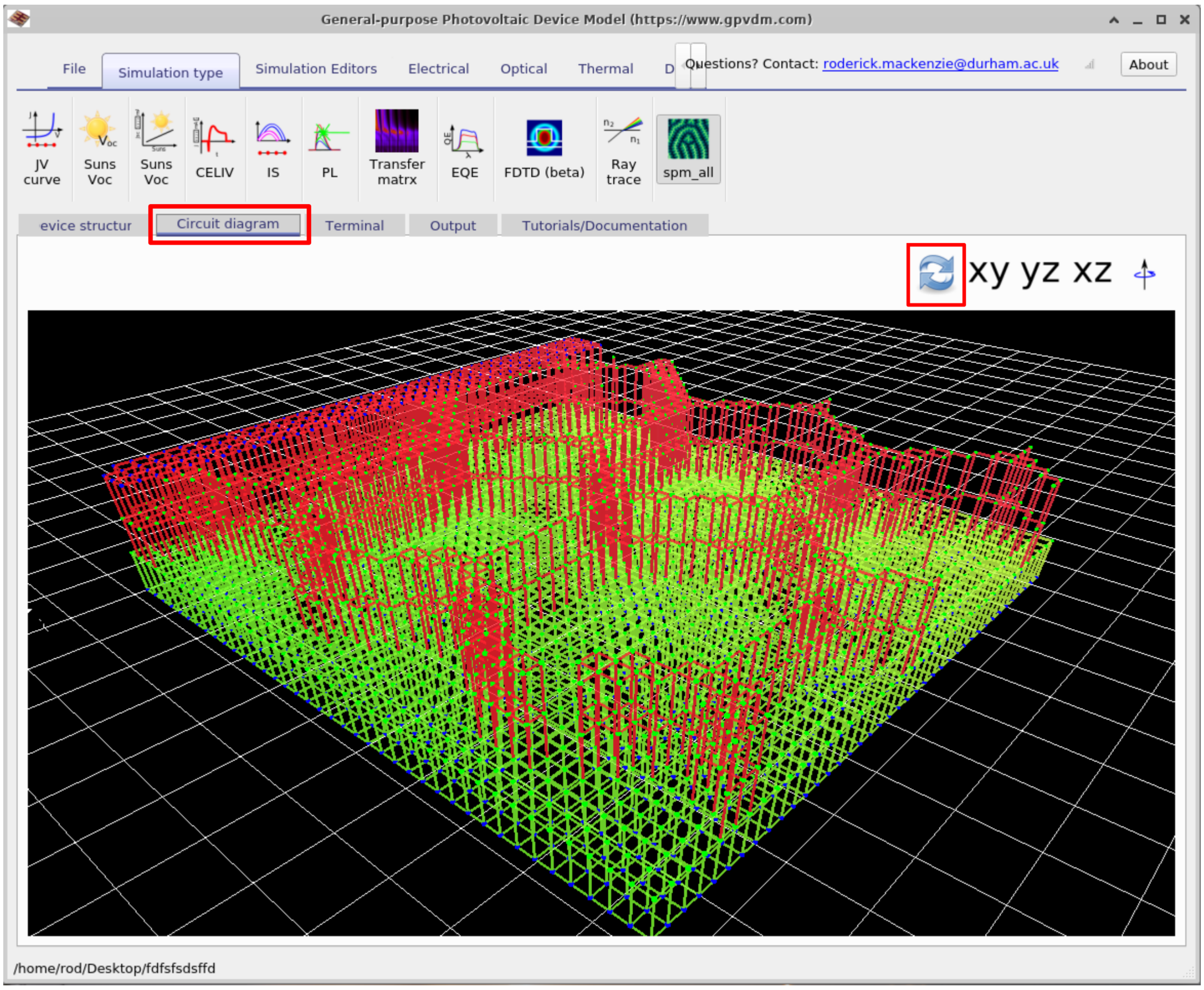
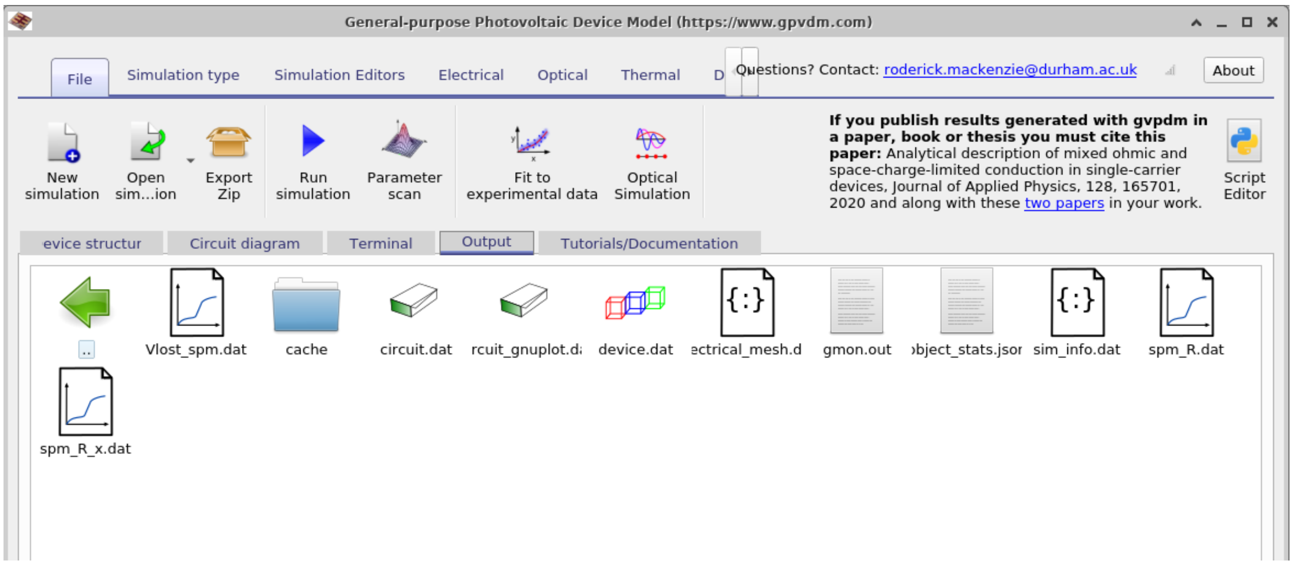


Figure 6.3: Building the 3D circuit mesh of the contact structure.

Once this is built we can run a full simulation and calculate the resistance between the bottom of the PEDOT:PSS layer (bottom of the green layer in figure 6.2) and the extracting silver contact (far left yellow strip on the top of figure 6.2). Run the simulation by clicking on the play button in the file ribbon.



If you publish results generated with gvpdm in a paper, book or thesis you must cite this paper: Analytical description of mixed ohmic and space-charge-limited conduction in single-carrier devices, Journal of Applied Physics, 128, 165701, 2020 and along with these two papers in your work.

Figure 6.4: A 1D diagram of the mesh

The simulation may take a while to run, once it has finished you can open the output files in the *Output* tab, see figure ??. If you open the file called *spm_R.dat* it will show you a resistance map of the structure which can be seen in figure 6.5. Other output files are listed below in table 6.1.

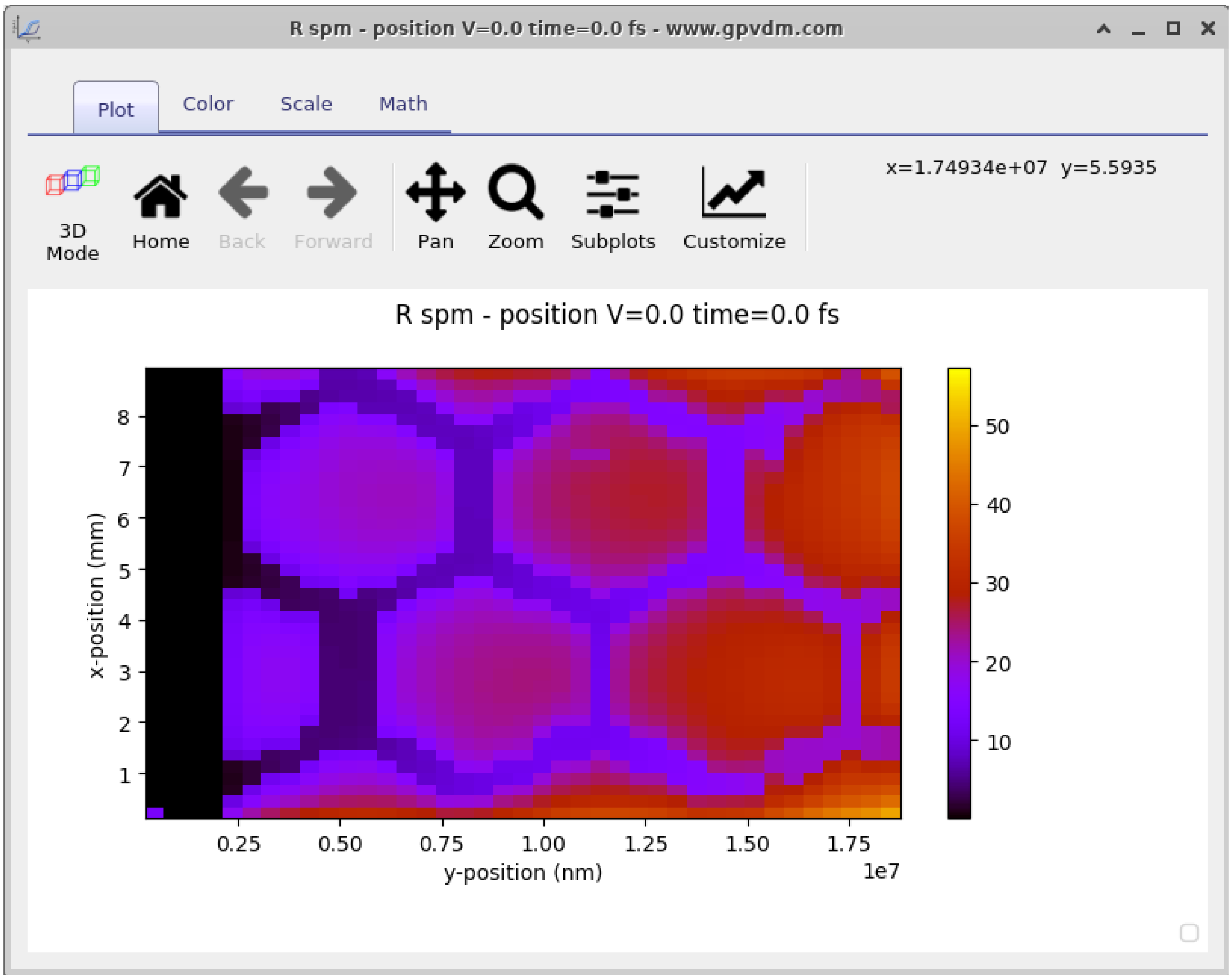


Figure 6.5: A 2D resistance plot across the surface of the device.

File name	Description
spm_R.dat	2D plot of resistance
spm_R_x.dat	A resistance plot down the centre of the device.

Table 6.1: Files produced by the SPM simulaton.

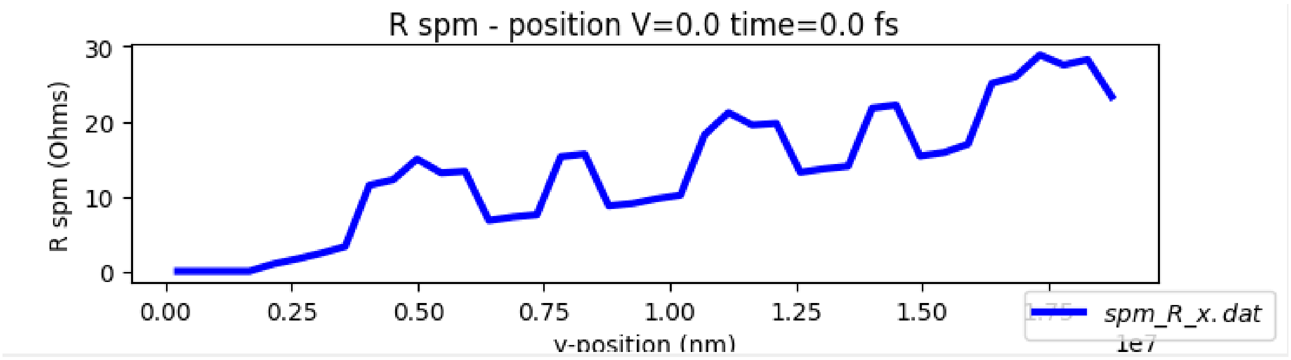
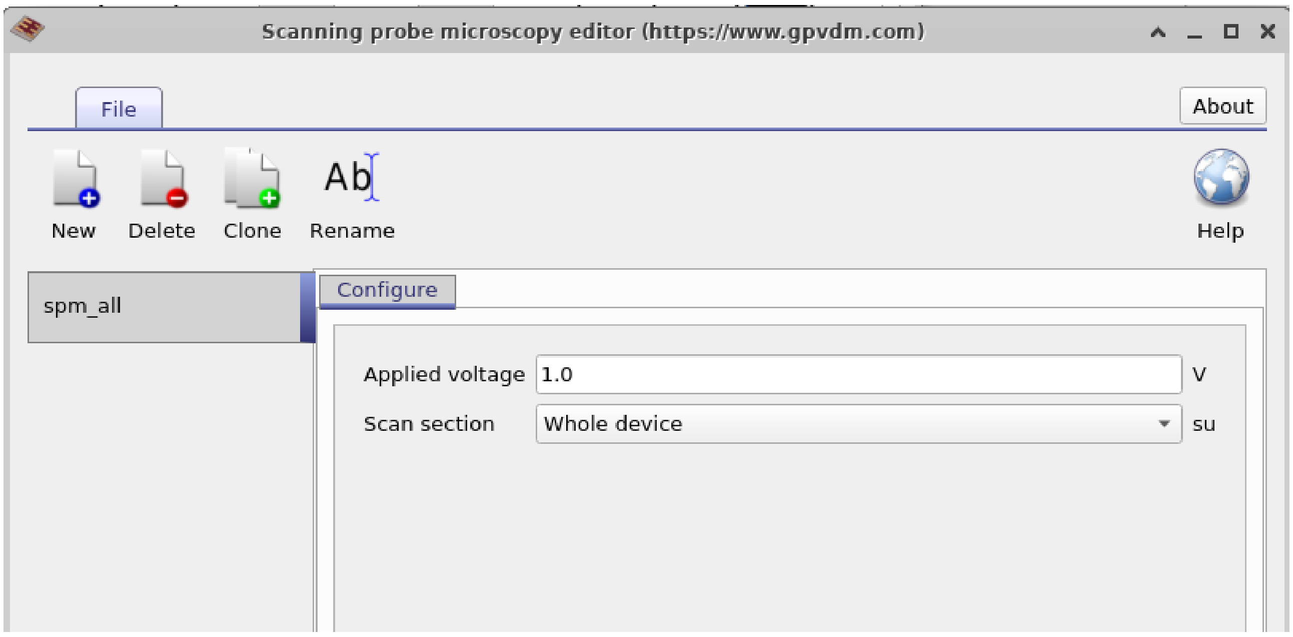


Figure 6.6: A 1D resistance plot taken through the centre of the device.

The scanning probe microscopy editor can be found in the *Simulation Editors* ribbon in the main window. This can be used to select if one scans the entire device or only section of it. The editor can be seen in figure 6.7



Applied voltage 1.0 V
Scan section Whole device su

Figure 6.7: The output files from the simulation.

6.2 Simulating large area solar cells

Chapter 7

2D simulation of bulk-heterojunctions

To be written but there is an example simulation in the new simulation window.

Chapter 8

Modelling excitons/geminate recombination - organics only

8.1 Why you should not model excitons

There are a number of models to calculate the number of geminate pairs which get converted to free charge carriers the Onsager-Braun model for example will give you the exciton dissociation efficiency. There are other models which will enable you to calculate the distribution of excitons in a device as a function of position. However, these models will generally require a number of parameters which are often not reliably known for a material system. Such parameters include exciton life-time, diffusion length and dissociation rate. So although it's possible (and interesting) to write a model to simulate geminate recombination, one is usually better off simply introducing a *photon efficiency factor* η_{photon}. This number ranges between 0.0 and 1.0 and is multiplied by the number of photons absorbed at any point in the device to account for geminate recombination losses.

$$G = G_{abs} \cdot \eta_{photon} \tag{8.1}$$

where G is the charge carrier generation rate in $m^{-3}s^{-1}$ in equations 16.22 and 16.25.

This factor can be obtained to a reasonable degree by comparing the difference between the simulated and experimental J_{sc}. This parameter can set in the configuration section of the optical simulation window. So therefore my advice is that in most cases you should not be modelling excitons explicitly but rather using the 'photon efficiency factor'. If you really want to model excitons read on..

8.2 Modelling excitons

So if you have read section 8.1 and still think you want to model excitons this section will explain how to do it. Gvpdm includes an exciton solver. This sits between the optical model and electrical model. If the exciton model is turned off then generation is simply the number of photons absorbed at any point in the device multiplied by the *photon efficiency factor* see equation 16.22. If the exciton model is turned on then optical absorption will feed straight into the exciton diffusion equation.

$$\frac{\partial X}{\partial t} = \nabla \cdot D \nabla X + G_{optical} - k_{dis}X - k_{FRET}X - k_{PL}X - \alpha X^2. \tag{8.2}$$

where X is the exciton density as a function of position, D is the diffusion constant, $G_{optical}$ exciton generation rate. This value is taken straight from the optical model. The constant k_{dis}

is exciton dissociation rate to free charge carriers. When the exciton model is switched on G in equations equals $k_{dis}X$. k_{FRET} is the Föster resonance energy transfer, $k_{PL}X$ is the radiative loss and α is an exciton-exciton annihilation rate constant. The diffusion term is defined as

$$D = \frac{L^2}{\tau} \tag{8.3}$$

Where L is exciton diffusion length and τ is the exciton lifetime.

8.3 Modeling excitions in a device

8.4 Modeling excitions in a unit cell

Chapter 9

The gpvdm file format

9.1 the .gpvdm simulation file format

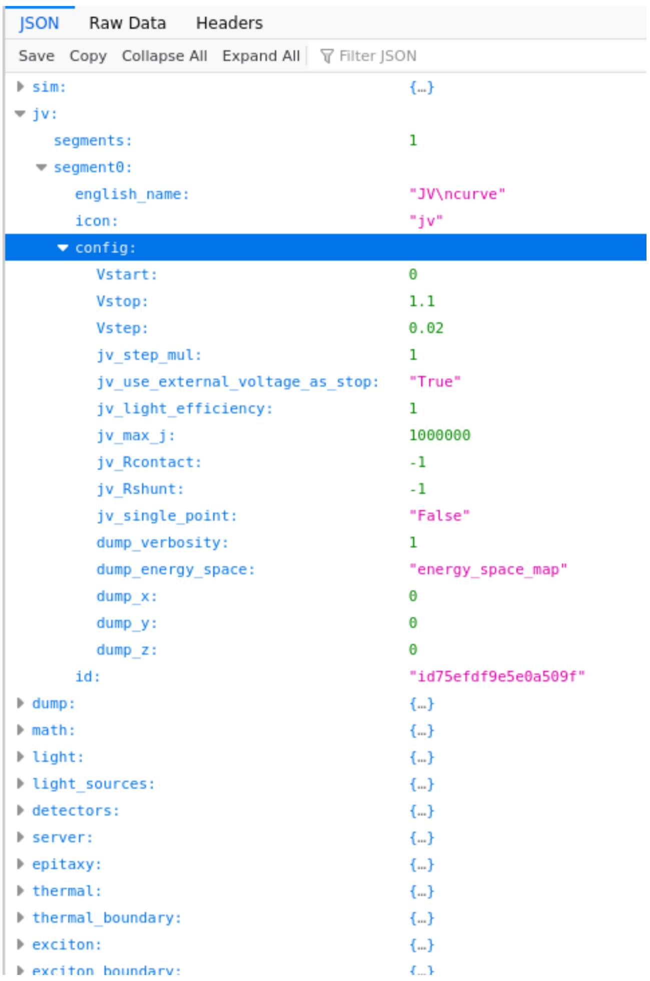
The .gpvdm file is simply a zip file. If you rename the file so to be called gpvdm.zip you will be able to open it in windows explorer or your favourite zip viwer. Inside the .gpvdm file is another file called sim.json. You can view this file in any text editor but the file is quite long so I recommend you use firefox to view it as it has a built in json viewer. Json is a simple way of storing text and configuration information first developed for Java, you can see examples here: `https://json.org/example.html`. Or below in code listing 2.

If you make a copy of sim.json outside the .gpvdm archive, then rename the sim.zip back to sim.gpvdm, gpvdm will ignore the sim.json file within the sim.gpvdm archive and revert to the plain text file stored in the simulation directory. This feature can be useful for automation of simulations as you can simply edit the sim.json file using your favourite programming language without having to learn about reading and writing zip files. If you open the sim.json file in firefox it will look like 9.1, also have a look at the file in notepad to get a sense of what is in it.

```
{
  "color_of_dog": "brown",
  "dog_age": 5,
  "dogs_toys": {
                "rabbit": "True",
                "stick": "False"
                }

}
```

Listing 1: JSON example



```
▸ sim:                                  {…}
▾ jv:
    segments:                           1
  ▾ segment0:
      english_name:                     "JV\ncurve"
      icon:                             "jv"
    ▾ config:
        Vstart:                         0
        Vstop:                          1.1
        Vstep:                          0.02
        jv_step_mul:                    1
        jv_use_external_voltage_as_stop: "True"
        jv_light_efficiency:            1
        jv_max_j:                       1000000
        jv_Rcontact:                    -1
        jv_Rshunt:                      -1
        jv_single_point:                "False"
        dump_verbosity:                 1
        dump_energy_space:              "energy_space_map"
        dump_x:                         0
        dump_y:                         0
        dump_z:                         0
      id:                               "id75efdf9e5e0a509f"
▸ dump:                                 {…}
▸ math:                                 {…}
▸ light:                                {…}
▸ light_sources:                        {…}
▸ detectors:                            {…}
▸ server:                               {…}
▸ epitaxy:                              {…}
▸ thermal:                              {…}
▸ thermal_boundary:                     {…}
▸ exciton:                              {…}
▸ exciton_boundary:                     {…}
```

Figure 9.1: An example sim.json file opened in firefox.

You can see that the json file has various headings, key headings are listed below in table 9.1

Heading	Description
sim	General simulation information
jv	JV curve configuration
dump	Defines how much information is written to disk
math	Math configuration for the solver
light	Optical transfer matrix configuration
light_sources	Configuration of light sources
epitaxy	Defines the structure of the device
thermal	Thermal configuration
thermal_boundary	Thermal boundary config.
exciton	Exciton config
exciton_boundary	Exciton boundary config.
ray	Ray tracing config.
suns_voc	Suns-Voc
suns_jsc	Suns-Jsc
ce	Charge Extraction config.
transfer_matrix	Light transfer matrix config
pl_ss	PL in steady state
eqe	EQE config.
fdtd	FDTD config.
fits	Fitting config.
mesh	Electrical mesh config.
time_domain	Time domain config.
fx_domain	FX-domain config
cv	CV config.
parasitic	Parasitic components
spm	Scanning Probe Microscopy config.
hard_limit	Setting hard limits for sim params.
perovskite	Perovskite solver config.
electrical_solver	Electrical solver config.
spctral2	SPCTRAL2
lasers	fs Lasers
circuit	Circuit solver config.
gl	OpenGL config
world	Defines the world box

Table 9.1: Key headings/sections in the sim.json file.

If you wish to programmatically drive gpvdm you can simply use one of the many available json editors most languages have them freely available.

9.2 Qwerks of the gpvdm json format

- Gpvdm json does not support standard json lists. If there is a list of items it is defined by firstly declaring the variable segments, with the number of items in the list so for example "segments",0 . Each item in the list is then stored under, "segment0", "segment1" etc... This is because gpvdm does not implement json arrays. This can be seen in figure 9.1 where there is a list with 1 segment.

- Many items in the json file will be given an id number which is a 16 digit hex code, this can be used to uniquely reference the item. An ID number can also be seen in figure 9.1

9.3 Encoding

The .json files read by gpvdm and written by gpvdm are always stored in UTF-8 format. Gpvdm can not handle UTF-16 or any other text encoding standards. Nowadays windows notepad and most other apps default to UTF-8, so if you don't know what these text storage formats are it probably does not matter. This will only rear it's head if you start programmatically generating .gpvdm files in a language such as C++ and are using a language such as Chinese or Russian with non Latin characters in it's alphabet.

Chapter 10

Databases

There are a series of databases used to define material parameters, shapes, emission spectra and solar spectra etc... These are described within this section. From the graphical user interface they can be accessed from the database ribbon, see figure 10.1.

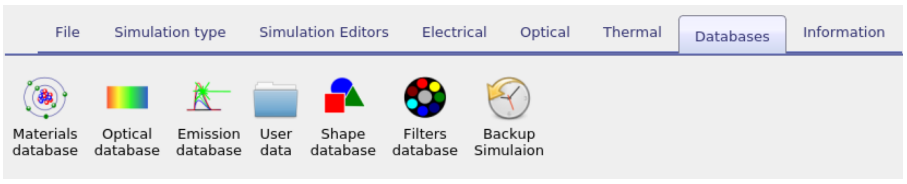


Figure 10.1: The database ribbon

There are two copies of these databases, one copy in the install directory of gpvdm C:\Program Files\gpvdm\ and one in your home directory in a folder called gpvdm_local. When the model starts for the first time it copies the read only materials database from, to the gpvdm_local folder in your home directory. If you delete the copy of the materials database in the gpvdm_local folder it will get copied back next time you start the model, this way you can always revert to the original databases if you damage the copy in gpvdm_local.

The structure of the databases are simple, they are a series of directories with one directory dedicated to each material or spectra etc.. E.g. there will be one directory called Ag in the optical database which defines silver, and another directory in the spectra database called am1.5g which defines the solar spectrum. Each of these database directories will from now on be referred to an object. Within each object there is a data.json file which defines basic material properties and configuration information. There will may be a couple of .bib files which contain reference information for the object in bibtex format and either .gmat files for n/k spectral data or .inp files for other types of data. All these files are just human readable text files, so you can open them in your preferred text editor such as notepad.

10.1 Materials database

Related YouTube videos:

A tutorial on adding new materials to gvpdm

This database primarily contains n/k data but also contains some electrical information and thermal information. Each subdirectory within the materials database identifies the material name. In each sub directory there are two key files *alpha.gmat* and *n.gmat*, these files are standard text files can be opened with any text editor such as wordpad. Alpha.gmat contains the absorption coefficient of the material while n.gmat contains the the refractive index. The first column of the file contains the wavelength in m (not cm or nm), and the second column of the file contains the absorption coefficient in m^{-1} (for alpha.omat) and the real part of the refractive index (i.e. n) in au (for n.omat). The data.json defines the material color and any known electrical or thermal data.

10.2 Adding new materials - the hard way

If you wish to add materials to the database which do not come as standard with the model you can do it in the following way: Simply copy an existing material directory (say gpvdm_local\oxieds\ito) to a new directory (say gpvdm_local\oxieds\mynewmaterial). Then replace alpha.gmat and n.gmat with your data for the new material. You can ignore the data.json file, although if you know the energy levels you can add the values in the file.

If you don't have data to hand for your material, but you do have a paper containing the data, you use the program Engauge Digitizer, written by Mark Mitchell `https://github.com/markummitchell/engauge-digitizer` to export data from publications. After you have finished updating the new material directory, whenever a new simulation is generated the new material files will automatically be copied into the active simulation directory ready for use.

10.3 Adding new materials - the easy way

To add a new material go to the data base ribbon and click on *Materials database* as shown in figure 10.2.

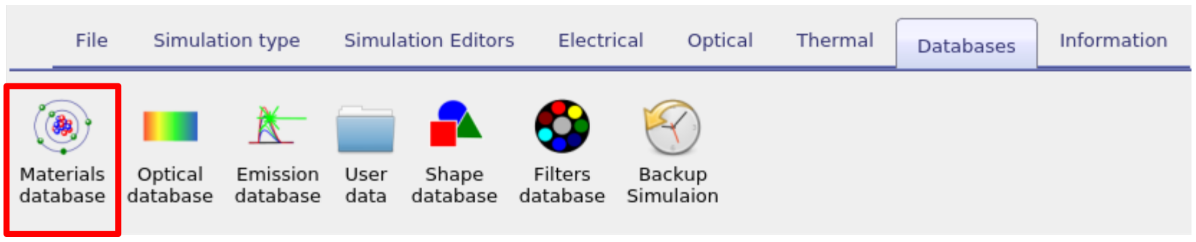


Figure 10.2: Opening the materials database

Then click *add material* in the top right of the window, this will bring up a dialogue box which will ask you to give a name for your new material, this is visible in figure 10.4.

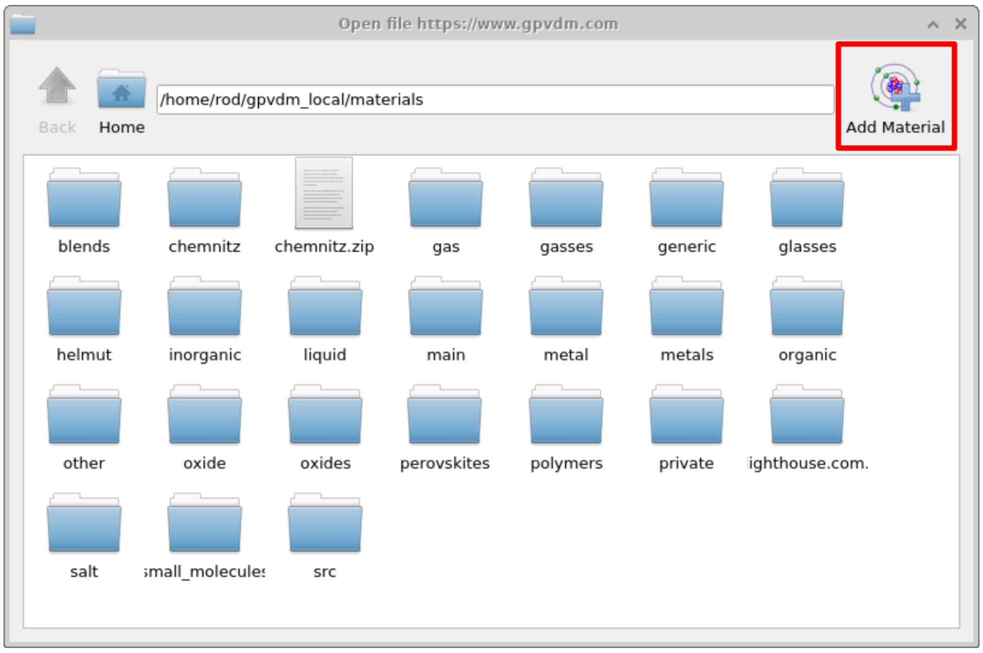


Figure 10.3: Select Add material

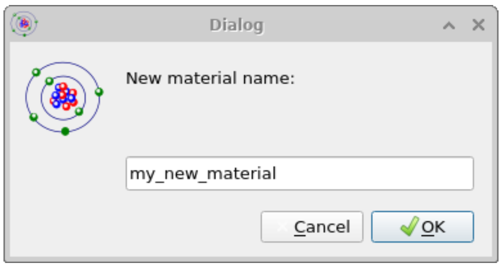


Figure 10.4: Type the name of the new material

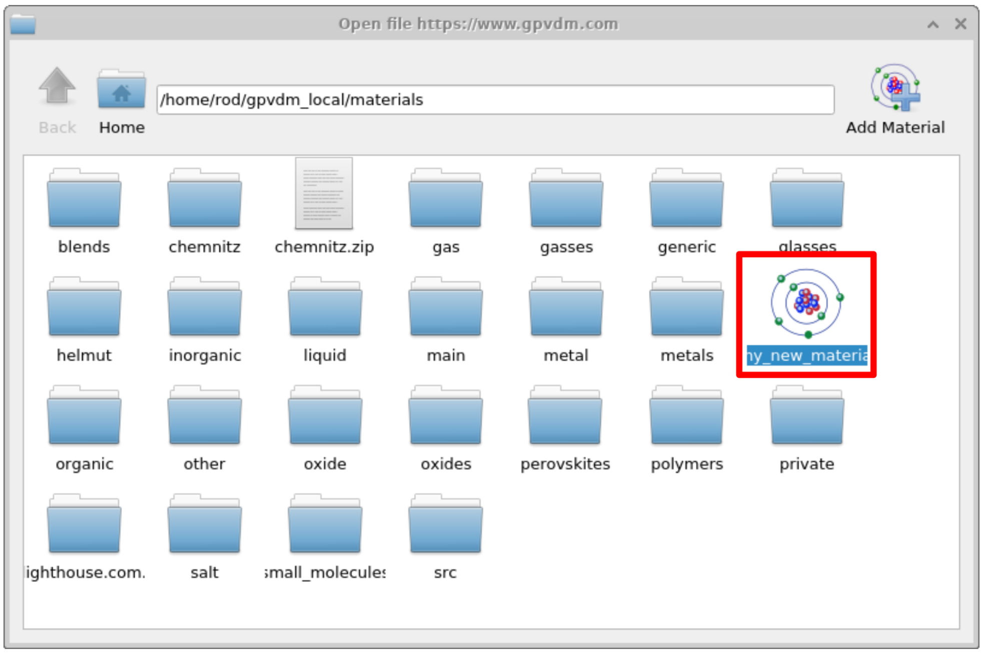


Figure 10.5: Open the new material

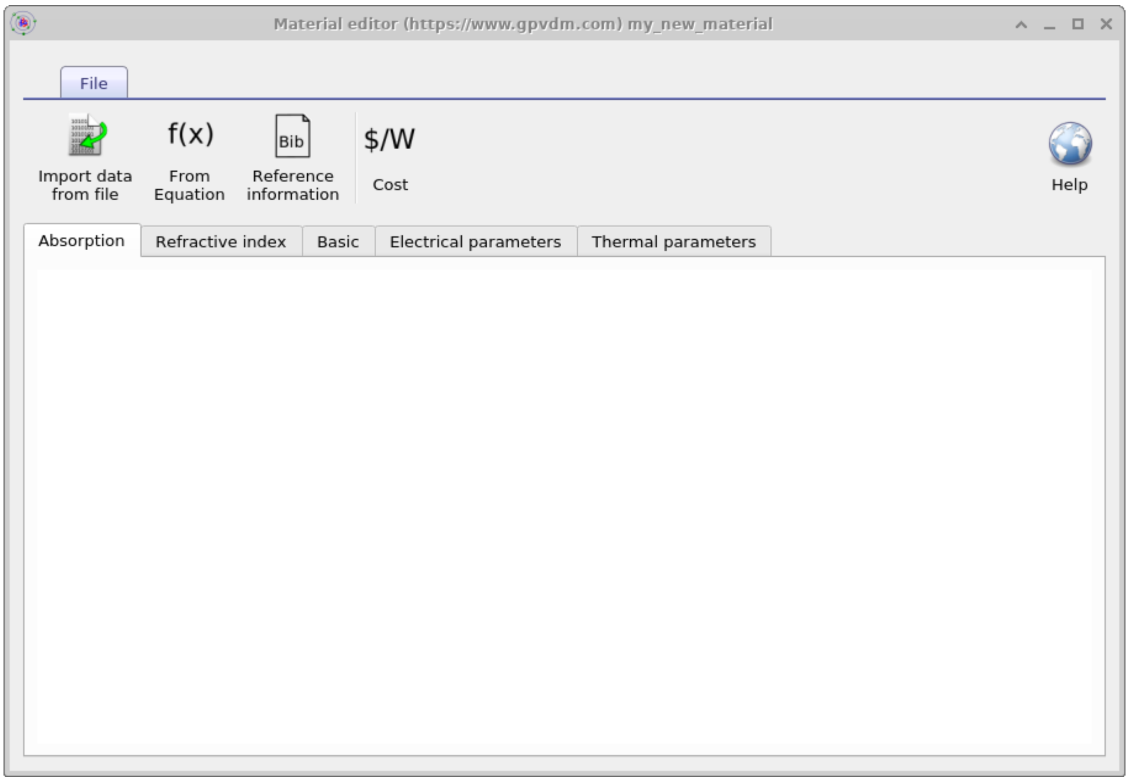
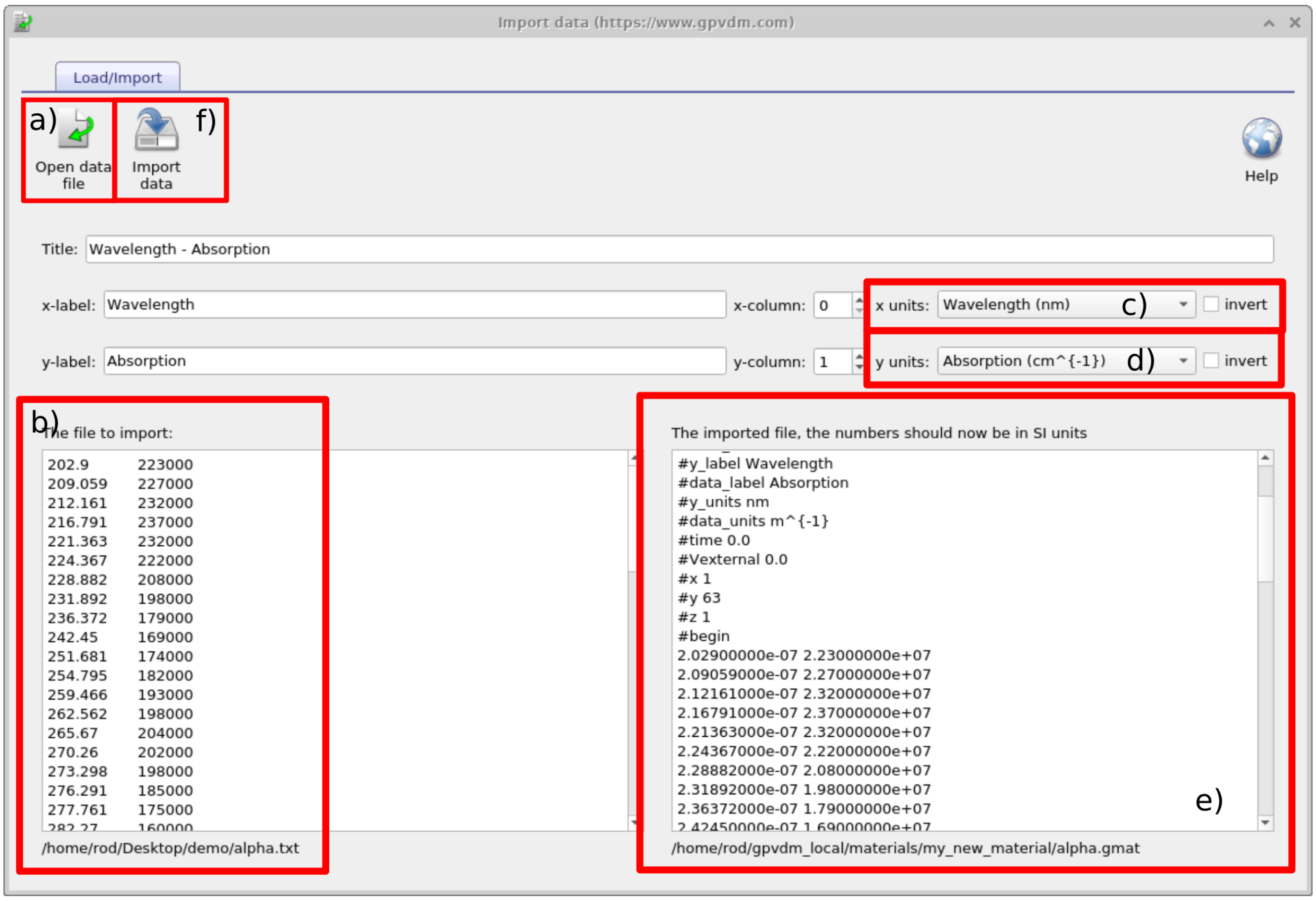


Figure 10.6: The new material without any data



Title: Wavelength - Absorption

x-label: Wavelength x-column: 0 x units: Wavelength (nm)

y-label: Absorption y-column: 1 y units: Absorption (cm^{-1})

The file to import:

```
202.9     223000
209.059   227000
212.161   232000
216.791   237000
221.363   232000
224.367   222000
228.882   208000
231.892   198000
236.372   179000
242.45    169000
251.681   174000
254.795   182000
259.466   193000
262.562   198000
265.67    204000
270.26    202000
273.298   198000
276.291   185000
277.761   175000
282.27    160000
```

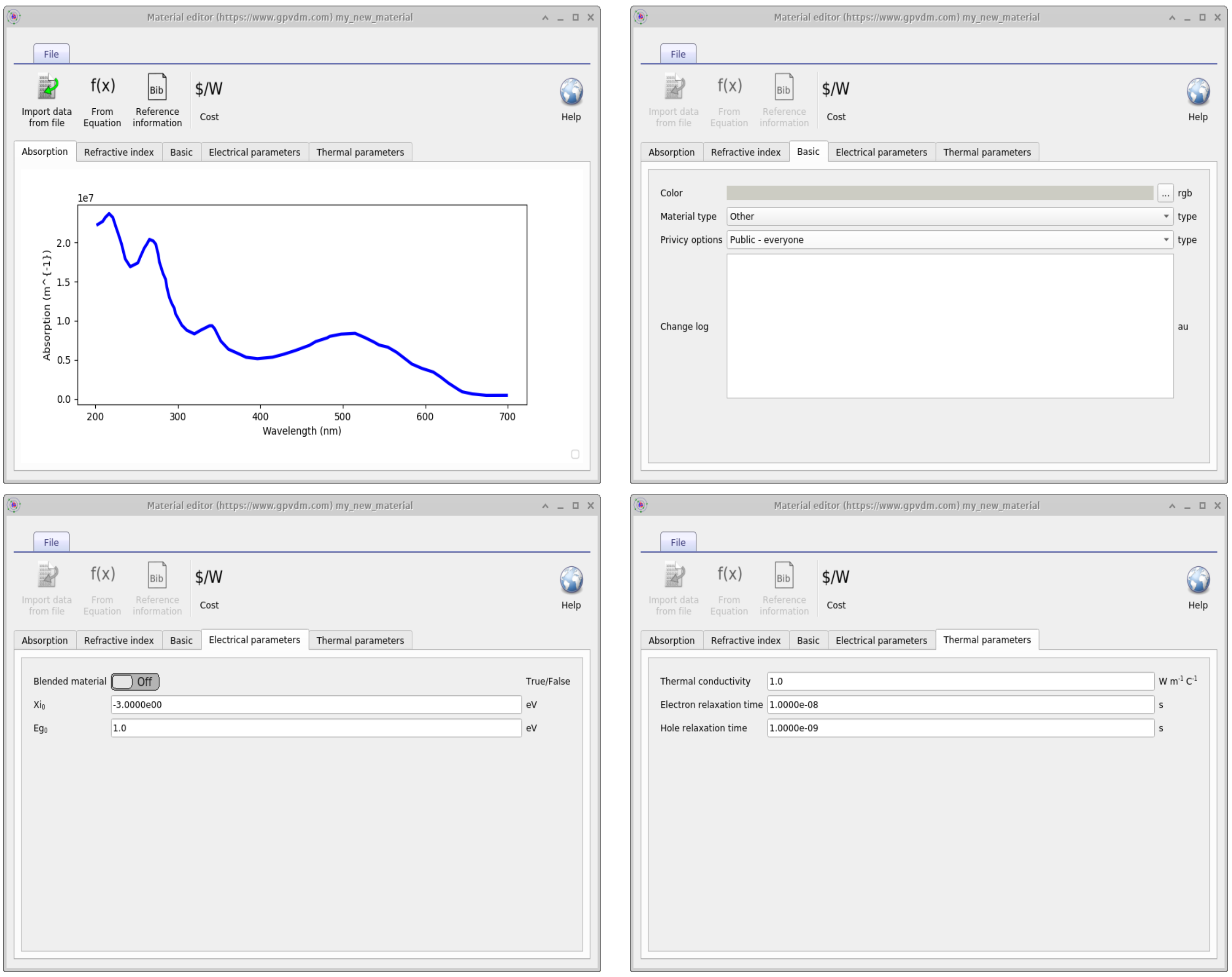
/home/rod/Desktop/demo/alpha.txt

The imported file, the numbers should now be in SI units

```
#y_label Wavelength
#data_label Absorption
#y_units nm
#data_units m^{-1}
#time 0.0
#Vexternal 0.0
#x 1
#y 63
#z 1
#begin
2.02900000e-07 2.23000000e+07
2.09059000e-07 2.27000000e+07
2.12161000e-07 2.32000000e+07
2.16791000e-07 2.37000000e+07
2.21363000e-07 2.32000000e+07
2.24367000e-07 2.22000000e+07
2.28882000e-07 2.08000000e+07
2.31892000e-07 1.98000000e+07
2.36372000e-07 1.79000000e+07
2.42450000e-07 1.69000000e+07
```

/home/rod/gpvdm_local/materials/my_new_material/alpha.gmat

Figure 10.7: The data importer window



Color	...	rgb
Material type	Other	type
Privicy options	Public - everyone	type
Change log		au

Blended material	Off	True/False
Xi_0	-3.0000e00	eV
Eg_0	1.0	eV

Thermal conductivity	1.0	W m^{-1} C^{-1}
Electron relaxation time	1.0000e-08	s
Hole relaxation time	1.0000e-09	s

Figure 10.8: Clockwise from the top left; The imported absorption sepctrum; The basic material parameters; The electrical parameters; and the Thermal parameters.

10.4 Emission database

This contains emission spectra for OLED materials.

10.5 Shape database

All physical objects within a simulation are *shapes*. For example the following things are all shapes; a layer of a solar cell; a layer of an OLED; a lens; a complex photonic crystal structure; contact stripe on an OFET; the complex hexagonal contact on a large area device (see figure 1.2 for more examples). These *shapes* are defined using triangular meshes for example a box which is used to define layers of solar cells, and layers of LEDs is defined using 12 triangles, two for each side. This box structure can be seen in figure 10.9.

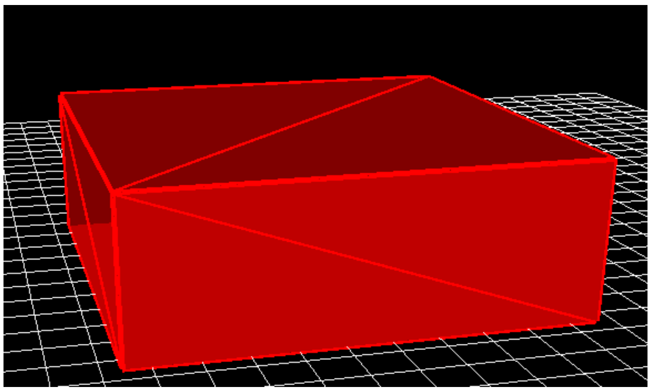


Figure 10.9: the box *shape*.

Shapes are stored in the shape database, this can be accessed via the database ribbon and clicking on the Shapes icon, see figure 10.10. By clicking on the *shape database* icon the shape database window can be brought up see figure 10.11.

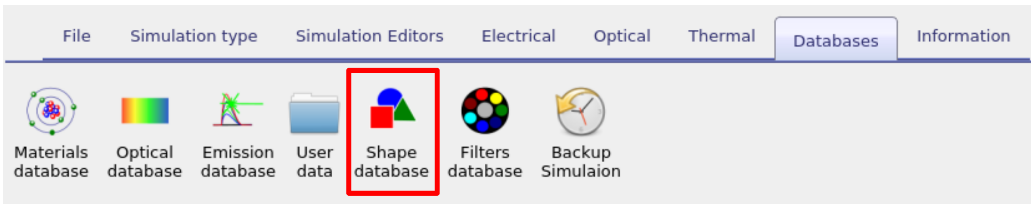


Figure 10.10: Opening the shape database

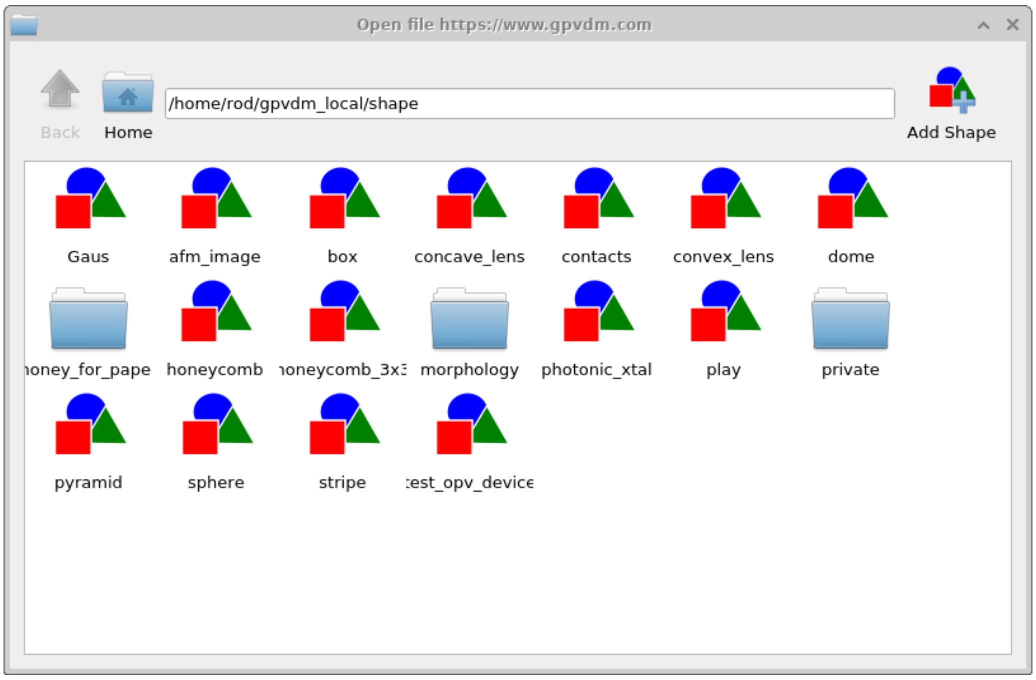


Figure 10.11: The shape database window

Try opening some of the shapes and have a look at them. You will get a window much like that shown in figure 10.12. Figure 10.12 shows a honeycomb contact structure of a solar cell. On the left of the window is the 3D shape, and on the right of the window is the 2D image which was used to generate it. Overlaid on the 2D image is a zx projection of the 3D mesh. The process of generating a shape involves first defining a 2D png image which you want to turn into a shape, in this case the 2D image is a series of hexagons and a bar at the top. This image is then converted into a triangular mesh using a discretization algorithm.

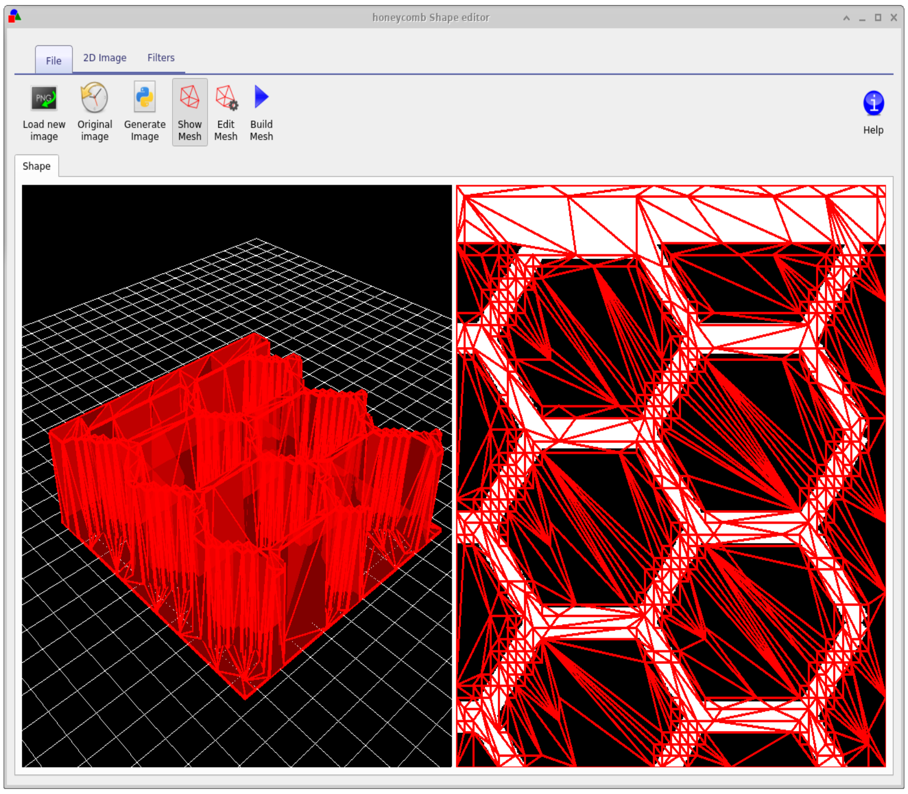


Figure 10.12: An example of a shape generated from a 2D png image. The 3D shape representing a hexagonal contact from a solar cell is on the left of the figure while the original 2D image is on the right.

Now try opening the shape *morphology/1* and you should see a window such as the one shown in figure 10.13, in the file ribbon find the icon which says *show mesh*. Try toggling it of and on, you will see the 2D mesh become hidden and then visible again. This example is a simulated bulk heterojunction morphology, but you can turn any 2D image into a shape by using the *load new image* button in the file ribbon. Try opening the mesh editor by clicking on *Edit Mesh* the *file* ribbon, you should get a window looking like figure 10.14. This configure window has three main options *x-triangles*, *y-triangles* and *method*. The values in *x-triangles*, *y-triangles* determine the maximum number of triangles used to discretize the image on the x and y axis. Try reducing the numbers to 40 then click on *Build mesh* in the file ribbon.

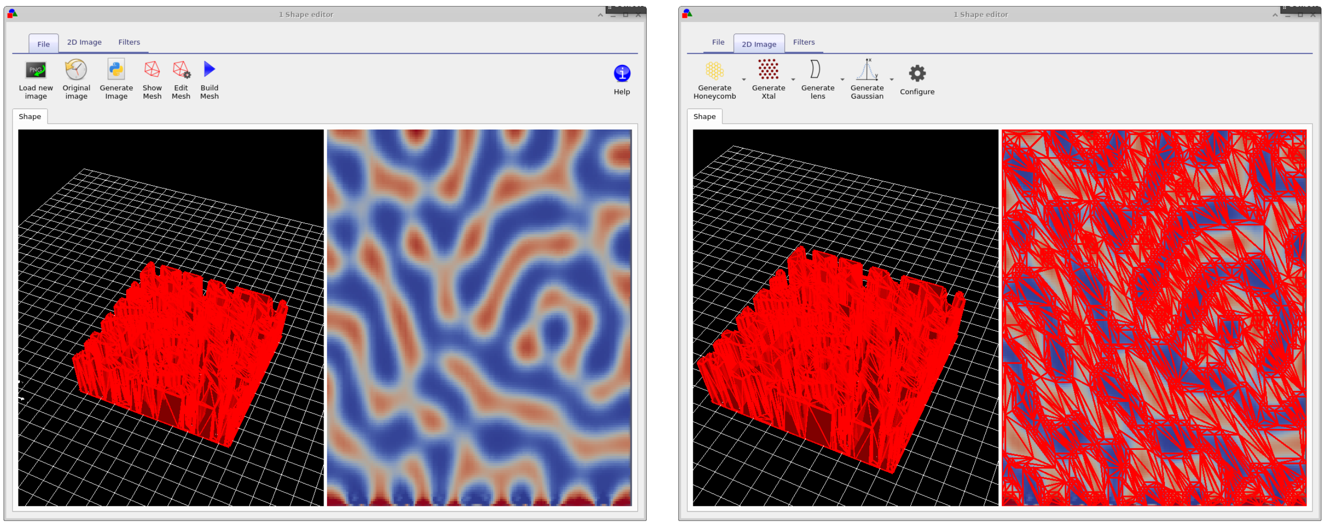
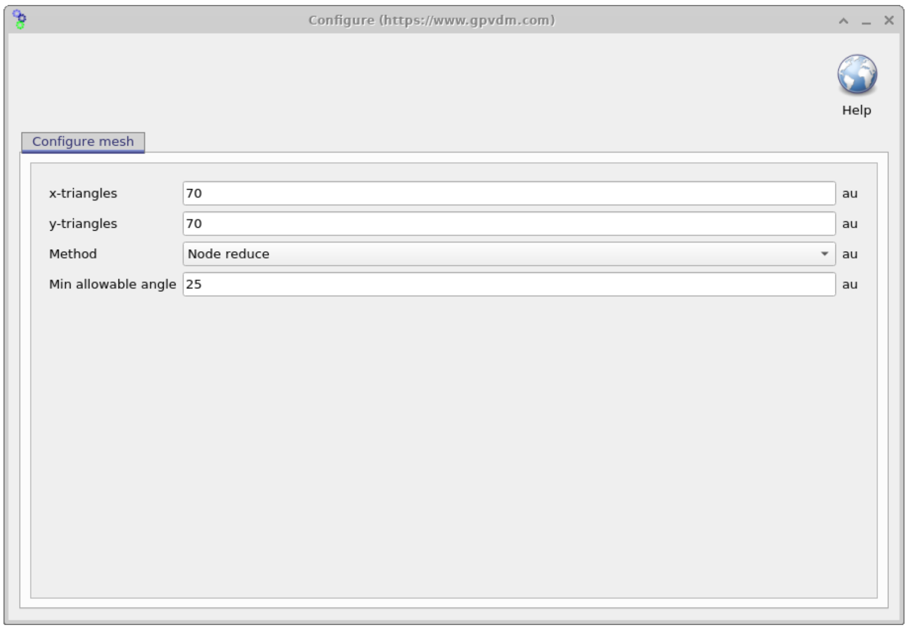


Figure 10.13: Clockwise from the top left; The imported absorption sepctrum; The basic material parameters; The electrical parameters; and the Thermal parameters.

You will see that the number of triangles used to describe the image reduce. The more triangles that are used to describe the shape the more accurately the shape can be reproduced, however the more triangles are used the more memory a shape will take up and the slower simulations will run. There is always a trade off between number of triangles used to discretize a shape. Try going back to the *Edit Mesh* window and set *method* to *no reduce* and then click on *Build mesh* from the file menu again. You will see that the complex triangular mesh as been replaced by a periodic triangular mesh, which is more accurate but requires the full 70x70 triangles. The difference between the *no reduce* and *Node reduce* options are that *no reduce* simply uses a regular mesh to describe and object and *Node reduce* starts off with a regular mesh then uses a node reduction algorithm to minimize the number of triangles used in the mesh.



x-triangles	70	au
y-triangles	70	au
Method	Node reduce	au
Min allowable angle	25	au

Figure 10.14: The mesh editor window, accessed via the file ribbon.

As well as loading images from file, the shape editor can generate it's own images for standard objects used in science, the 2D image ribbon is visible in the right hand panel of figure 10.13. There are options to generate lenses, honeycomb structures and photonic crystals.

Each button has a drop down menu to the right of it which can be used to configure exactly what shape is generated.

The final ribbon to be discussed is the *Filters* ribbon. This is used to change loaded images, try turning on and off the threshold function. This applies a threshold to an image so that RGB values above a given value are set to white and those below are set to black. There are also other functions such as Blur, and Boundary which can be used to blur and image and apply boundaries to an image.

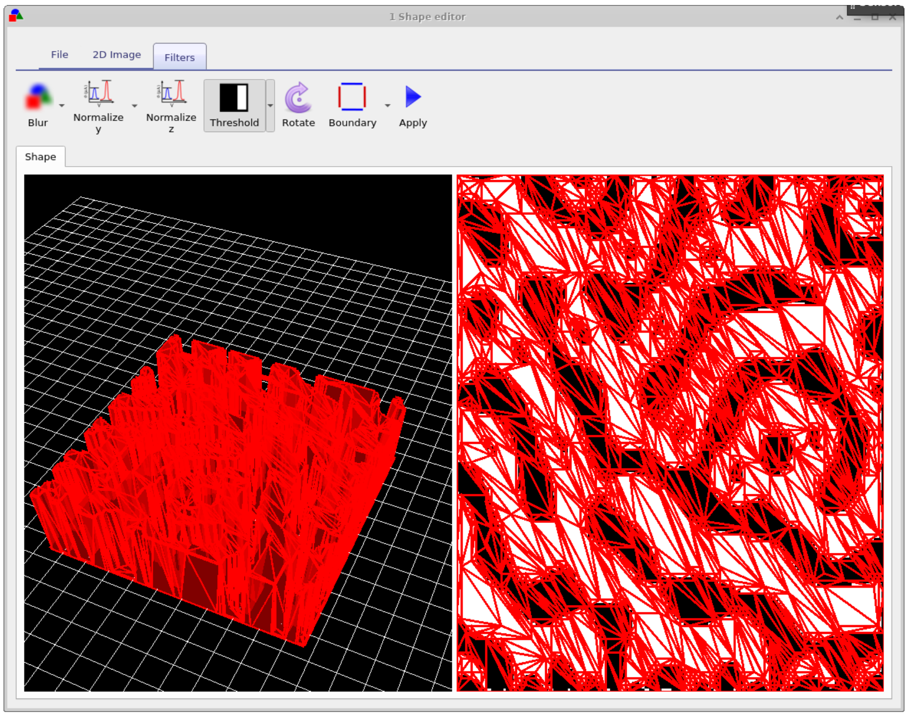


Figure 10.15: The shape database

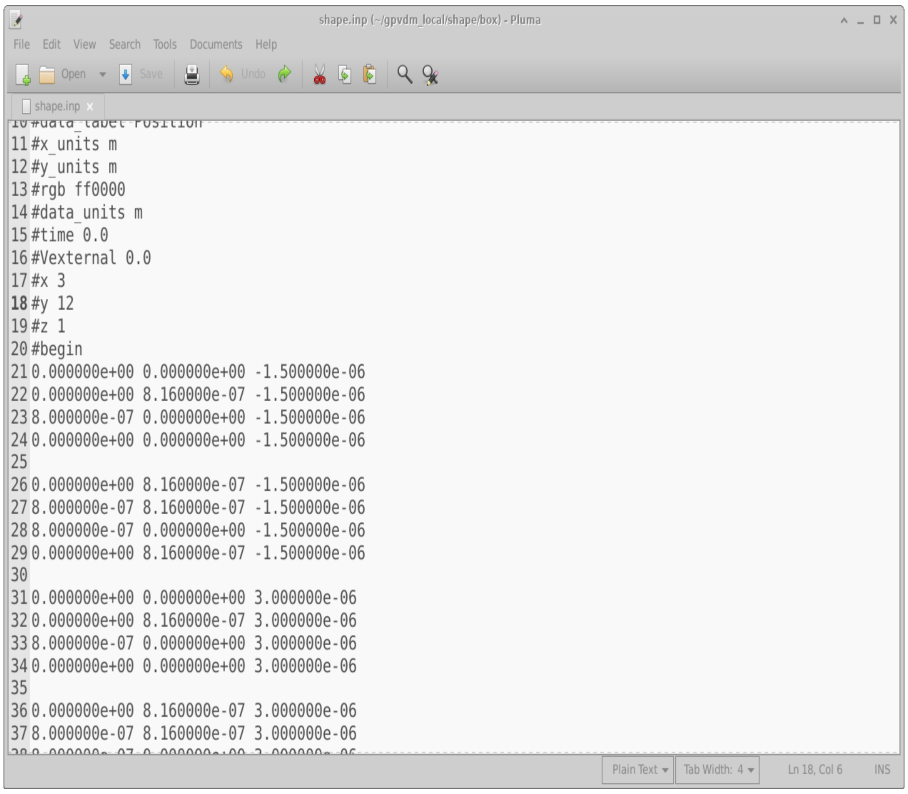
10.5.1 The shape file format

A shape has to be a fully enclosed volume, if you use the built in shape discretizer this will be done for you automaticity. However if you are building shapes by hand you will have to enforce this condition. Each shape directory contains the following files

File name	Description
data.json	Holds the configuration for the shape file
image_original.png	backup of the imported image
image_out.png	The final processed image
image.png	The imported image which may be modified.
shape.inp	The discretized 3D structure.

Table 10.1: The files within a shape directory

The png files are of images in various states of modification. The data.json file stores the configuration of the shape editor and the shape.inp file contains the 3D structure of the object. An example of a shape.inp file is shown below in 10.16. The file format has been written so that gnuplot can open it using the splot command without any modification. As such each triangle is comprised of four z,x,y points (lines 21-24), the first three lines define the triangle, and the forth line is a repeat of the first line so that gnuplot can plot the triangle nicely. The number of triangles in the file is defined on line 18 using the #y command. The exact magnitudes of the z,x,y values do not matter because as soon as the shape is loaded all values are normalized so that the minimum point of the shape sits at 0,0,0 and the maximum point from the origin sits at 1,1,1. When being inserted into a scene, the shape is then again renormalized to the desired size of the object in the device.



```
10 #data_label Position
11 #x_units m
12 #y_units m
13 #rgb ff0000
14 #data_units m
15 #time 0.0
16 #Vexternal 0.0
17 #x 3
18 #y 12
19 #z 1
20 #begin
21 0.000000e+00 0.000000e+00 -1.500000e-06
22 0.000000e+00 8.160000e-07 -1.500000e-06
23 8.000000e-07 0.000000e+00 -1.500000e-06
24 0.000000e+00 0.000000e+00 -1.500000e-06
25
26 0.000000e+00 8.160000e-07 -1.500000e-06
27 8.000000e-07 8.160000e-07 -1.500000e-06
28 8.000000e-07 0.000000e+00 -1.500000e-06
29 0.000000e+00 8.160000e-07 -1.500000e-06
30
31 0.000000e+00 0.000000e+00 3.000000e-06
32 0.000000e+00 8.160000e-07 3.000000e-06
33 8.000000e-07 0.000000e+00 3.000000e-06
34 0.000000e+00 0.000000e+00 3.000000e-06
35
36 0.000000e+00 8.160000e-07 3.000000e-06
37 8.000000e-07 8.160000e-07 3.000000e-06
38 8.000000e-07 0.000000e+00 3.000000e-06
```

Figure 10.16: An example of the shape.inp file.

10.6 Filters database

This contains optical filters.

10.7 Backups of simulations

Very often when running a simulation you want to make a copy of it before continuing to play with the parameters. To do this click on the backup simulation button in the database ribbon (see figure 10.1), this will bring up the backup window, see figure 10.17. If you click on the ”New backup” icon on the top right of the window, a backup will be made of your current simulation. And an icon representing the backup will appear in the backup window. To restore the backup double click on the icon representing your stored simulation. Note this backup is only stored in the your local simulation directory, and is more of a checkpoint than a real backup.... so make sure you have other copies of your simulation if it is very important to you..

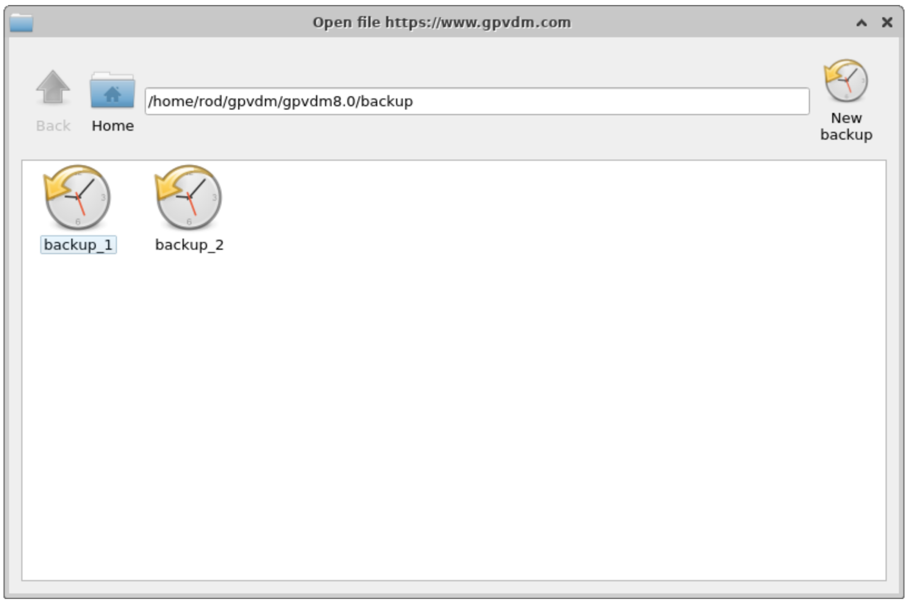

Open file https://www.gpvdm.com
Back
Home
/home/rod/gpvdm/gpvdm8.0/backup
New backup
backup_1
backup_2

Figure 10.17: Backing up a simulation

Chapter 11

Optical models

11.1 Light sources

In gpvdm light comes from light sources, which are defined in the light source editor which can be found in the Optical ribbon of the main window see figure 11.1.

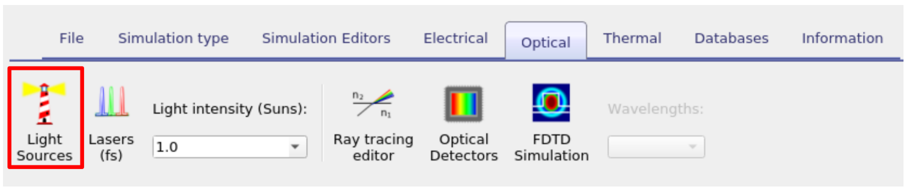


Figure 11.1: Opening the light source editor.

The light source editor is shown below in figure ??. Each light source consists of of illumination spectra and optical filters. In this way you can define a light source to emit AM1.5G but then filter out various components of the spectrum. This enables one to simulate for example a device under AM1.5G spectra but behind a thick glass contact which takes sunlight below 300 nm. Light sources can be combined in the ”Light source” tab, so for example one could combine AM1.5G and light given off by a fluorescent tube. Each spectrum is multiplied by a ”Multiplayer” defined in the ”Multiplayer” column this defines the relative intensity of the light sources.

In the filters tab you can define the optical filters. They can be enabled or disabled using the Enable switch, you can select the material which will act as a filter, and decide how much the filter attenuates in dB.

The configure tab of each filter can be used to select where the light comes from, options are:

1. Top: Light will come form the top of the device. (y0) This is usually used with the transfer matrix solver. See the bottom left of figure 11.3.

2. Bottom: Light will come from the bottom of the device. (y1) This is usually used with the transfer matrix solver. See the bottom right of figure 11.3.

3. xyz: The light can come from an xyz position in the simulation space this used for FDTD simulations or ray tracing simulations. This can be seen in to bottom right of figure 11.3.

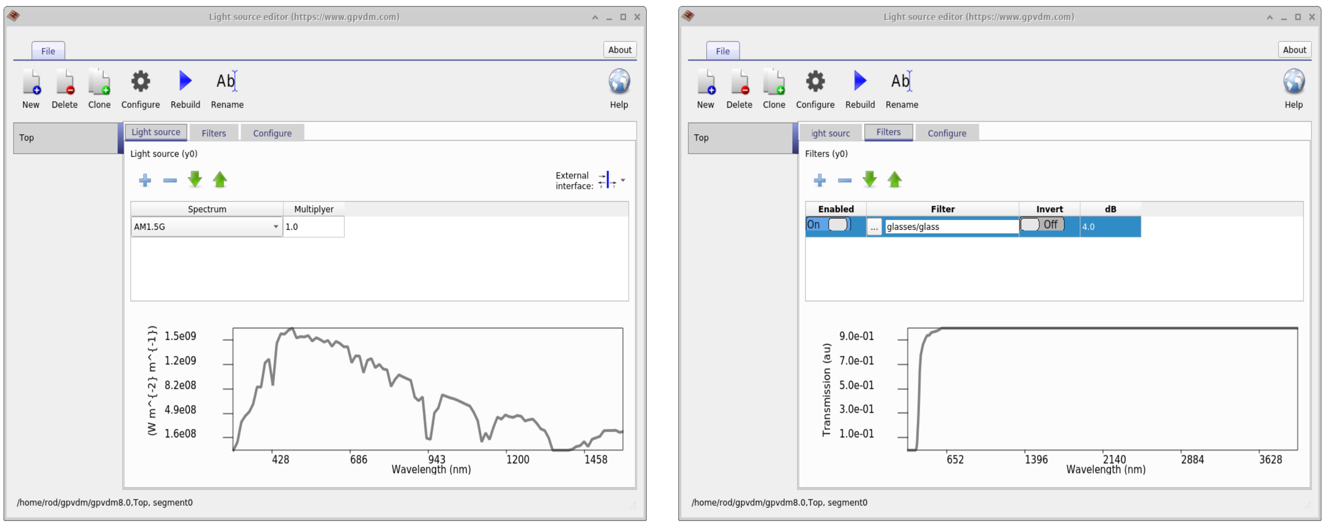
Stop and start wavelengths can also be set in the configure tab.

11.1.1 Local ground view factor

The local ground view factor which is given as [4]

$$F_{ground} = sin^2\left(\frac{\theta_t}{2}\right) = \frac{1 - cos(\theta_t)}{2} \tag{11.1}$$

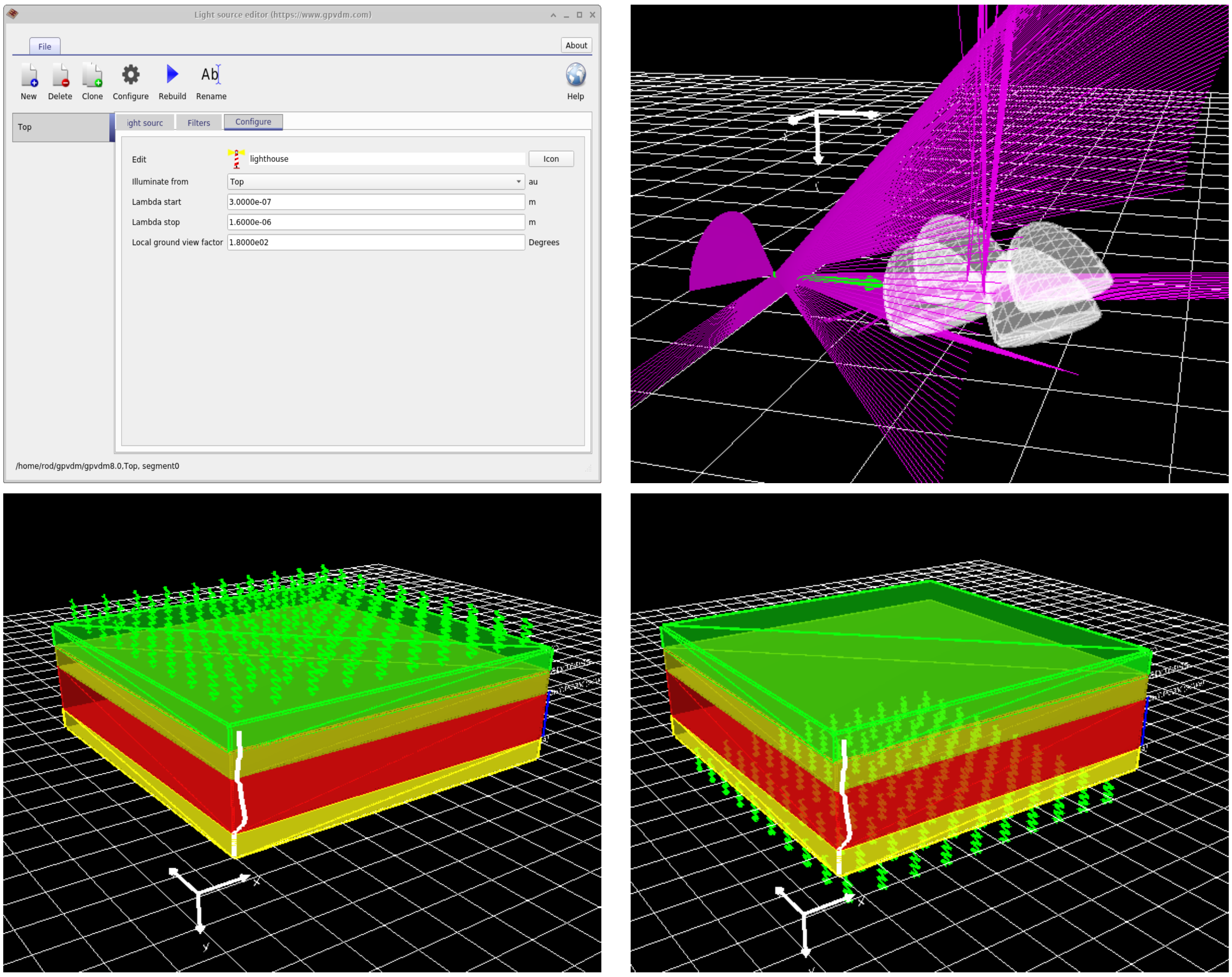
can be set in the configure tab.



Spectrum	Multiplyer
AM1.5G	1.0

Enabled	Filter	Invert	dB
On	... glasses/glass	Off	4.0

Figure 11.2: Left: Building an optical spectrum; Right modifying the light sources with optical filters.



Edit	lighthouse	
Illuminate from	Top	au
Lambda start	3.0000e-07	m
Lambda stop	1.6000e-06	m
Local ground view factor	1.8000e02	Degrees

/home/rod/gpvdm/gpvdm8.0,Top, segment0

Figure 11.3: Top left: The configure panel of the light source editor; Top right: Illuminate from set to "xyz"; bottom left: Illuminate from set to "top"; bottom right: Illuminate from set to "bottom".

11.2 Transfer matrix model

The transfer matrix model is used for simulating external light incident on the device. It assumes light hits the device normal to the surface and the light has reached steady state. This method is good for understanding optical absorption, reflection and transition in structures such as solar cells, optical filters or sensors. In general any structure where the structure can generally be described as 1D. There are other methods which can be used for this such as FDTD, however general speaking the transfer matrix model will be orders of magnitude faster.

11.2.1 The user interface

This simulation can be reached from the file ribbon and selection "Optical simulation". If you click "Run optical simulation" (see 11.4) the distribution of light within the structure will be calculated as a function of position and wavelength. You can see from the top of the figure that there are various simulation modes. The full transfer matrix method is selected by selecting "Transfer matrix", this will do full optical simulation. There are also other simplified simulation modes which allow the user to explore more simple charge carrier generation profiles and answer "what if" questions.

- Transfer matrix: This is a full transfer matrix simulation which takes into account multiple reflections from the interfaces and optical loss within the structure. This is in effect solving the wave equation in 1D and is an accurate (and recommended) optical model to use.

- Exponential profile: This is a very simple optical model which assumes light decays exponentially according to the relation

$$I = I_0 e^{-\alpha x} \tag{11.2}$$

  between layers and and assumes light is transmitted between layers according to the formula:

$$T = 1.0 - \frac{n_1 - n_0}{n_1 + n_0} \tag{11.3}$$

- Flat profile: The flat profile assumes light is constant within layers and only decreases at material interfaces according to equation 11.3.

- From file: This can be used to import generation profiles from a file. It us generally used to import the results of more complex optical simulations such as those from external FDTD solvers.

- Constant value: If you click on the arrow to the right of the simulation button you will be able to set the charge carrier generation rate within each layer by hand.

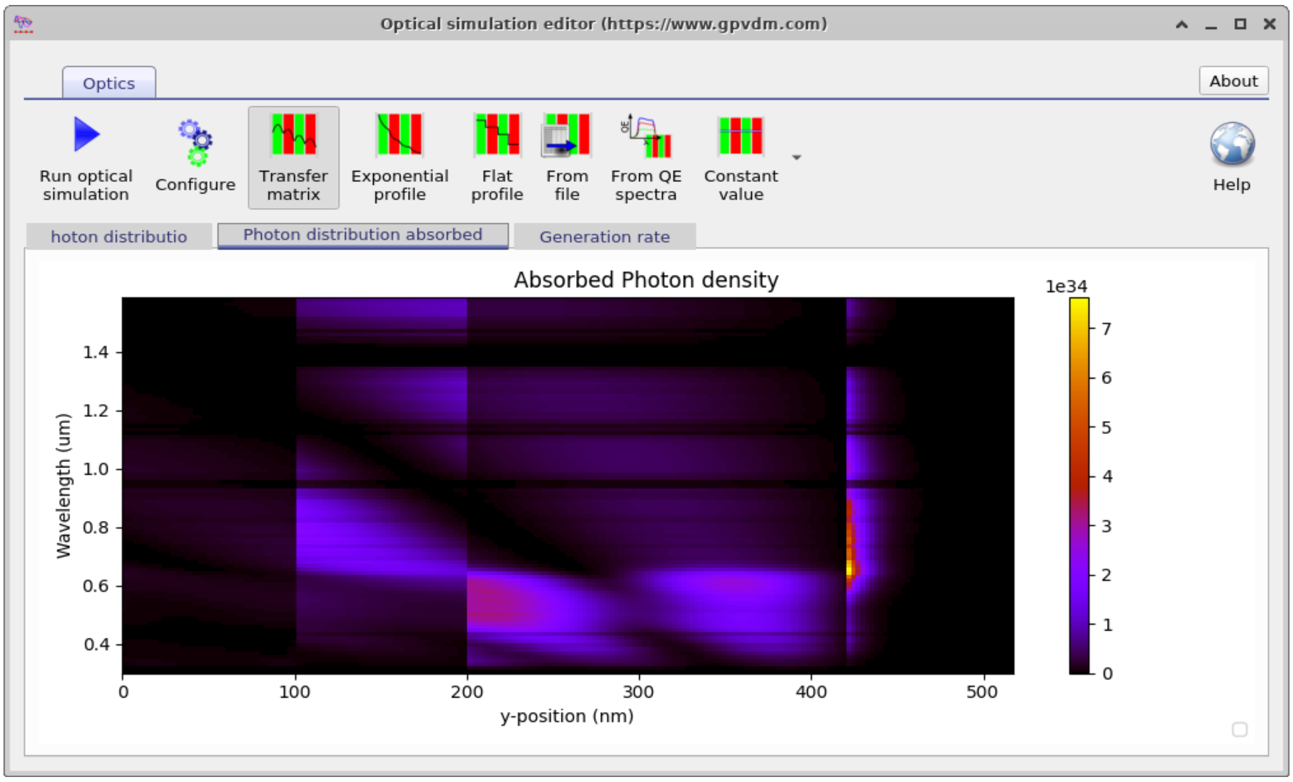
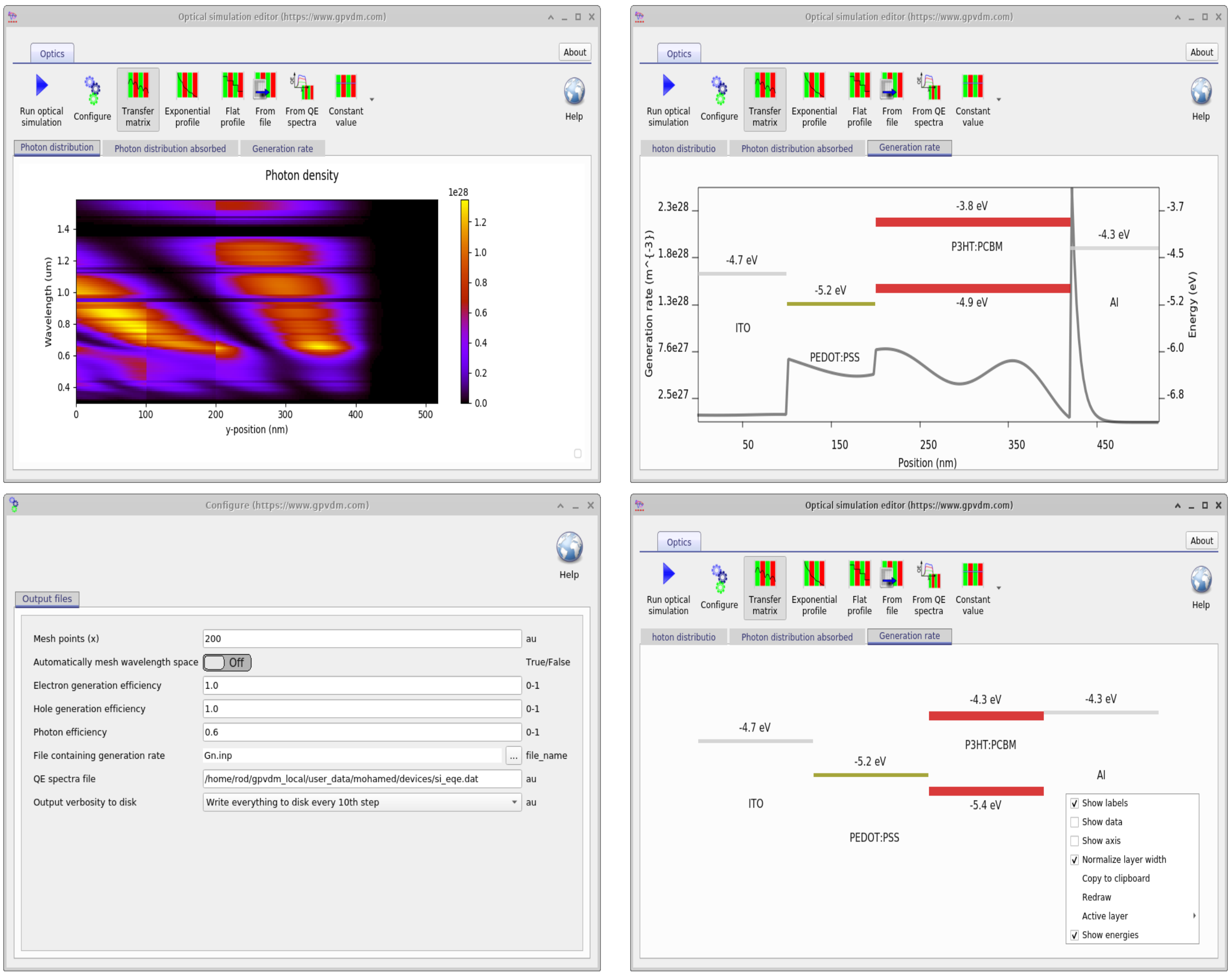


Figure 11.4: The output tab this is just like windows file explorer, you can explore the simulation directory tree.

The optical simulation window has various tabs which can be used to explore how light interacts with the device. These can be seen in figure 11.5. The the top left hand image shows the photon density within the device, the image on the right shows the *total* photon density within the layers of the device. Notice how the reflection of the light of various layers causes interfearance patterns. Bottom left shows the configuration of the optical model, you can set here things like the number of wavelengths which are simulated or the number of x-points which are used to describe the device in position space. Bottom right shows the same figure as in the top right of the figure, except by right clicking and playing with the menu options the figure has been converted into band diagram. This can be useful for generating band diagram figures for papers.



Mesh points (x)	200	au
Automatically mesh wavelength space	Off	True/False
Electron generation efficiency	1.0	0-1
Hole generation efficiency	1.0	0-1
Photon efficiency	0.6	0-1
File containing generation rate	Gn.inp	file_name
QE spectra file	/home/rod/gpvdm_local/user_data/mohamed/devices/si_eqe.dat	au
Output verbosity to disk	Write everything to disk every 10th step	au

Figure 11.5: Various views of the optical simulation window

11.2.2 Theory of the transfer matrix method

On the left of the interface the electric field is given by

$$E_1 = E_1^+ e^{-jk_1 z} + E_1^- e^{jk_1 z} \tag{11.4}$$

and on the right hand side of the interface the electric field is given by

$$E_2 = E_2^+ e^{-jk_2 z} + E_2^- e^{jk_2 z} \tag{11.5}$$

Maxwel's equations give us the relationship between the electric and magnetic fields for a plane wave.

$$\nabla \times E = -j\omega\mu H \tag{11.6}$$

which simplifies to:

$$\frac{\partial E}{\partial z} = -j\omega\mu H \tag{11.7}$$

Applying equation 11.7 to equations 11.4-11.5, we can get the magnetic field on the left of the interface

$$-j\mu\omega H_1^y = -jk_1 E_1^+ e^{-jk_1 z} + jk_1 E_1^- e^{jk_1 z} \tag{11.8}$$

and on the right of the interface

$$-j\mu\omega H_2^y = -jk_2 E_2^+ e^{-jk_2 z} + jk_2 E_2^- e^{jk_2 z}. \tag{11.9}$$

Tidying up gives,

$$H_1^y = \frac{k}{\omega\mu} E_1^+ e^{-jk_1 z} - \frac{k}{\omega\mu} E_1^- e^{jk_1 z} \tag{11.10}$$

$$H_2^y = \frac{k}{\omega\mu} E_2^+ e^{-jk_2 z} - \frac{k}{\omega\mu} E_2^- e^{jk_2 z} \tag{11.11}$$

11.2.3 Refractive index and absorption

$$E(z,t) = Re(E_0 e^{j(-kz+\omega t)}) = Re(E_0 e^{j(\frac{-2\pi(n+j\kappa)}{\lambda}z+\omega t)}) = e^{\frac{2\pi\kappa z}{\lambda}} Re(E_0 e^{\frac{j(-2\pi(n+j\kappa)}{\lambda}z+\omega t}) \tag{11.12}$$

And because the intensity is proportional to the square of the electric field the absorption coefficient becomes

$$e^{-\alpha x} = e^{\frac{2\pi\kappa z}{\lambda}} \tag{11.13}$$

$$\alpha = -\frac{4\pi\kappa}{\lambda_0} \tag{11.14}$$

11.3 Finite Difference Time Domain

Finite Difference Time Domain (FDTD) is a very computationally intensive way to simulate the propagation of electromagnetic radiation. It solves full on Maxwell's equations in time domain making no assumptions. This approach has only been possible in the last few years with the increase in computing power.

Food for thought: Before using FDTD, you should be aware that in many cases when simulating a device, using FDTD is like using a sledge hammer to crack a nut. For example when simulating a standard solar cell, one could use FDTD, this would involve simulating a wave front entering the cell through the top contact in time domain and waiting until the optical field reaches steady state then calculating how much light is being absorbed. This would require thousands of time domain simulation steps per wavelength. However usually one is not interested in the time evolution of light in a solar cell as sunlight varies very slowly indeed, so one is better off not using a steady state method but other methods which assume light has already reached steady state, such as the transfer matrix method discussed above.

Never the less, FDTD is an import method, and can be used to design and understand complex devices.

Related YouTube videos:

Generating photonic crystal structures for FDTD simulation

Tutorial on simulating photonic crystal waveguides using FDTD

11.3.1 Running an FDTD simulation

There is a demo FDTD simulation in the new simulations window. To open this click on new simulation in the file ribbon of the main window. The window in figure 11.6 will appear. Double click on the "Photonic-xtal FDTD" simulation to open it. Save the simulation on your local hard disk, don't save it on a remote disk, USB disk or OneDrive they will be too slow to run the simulation.

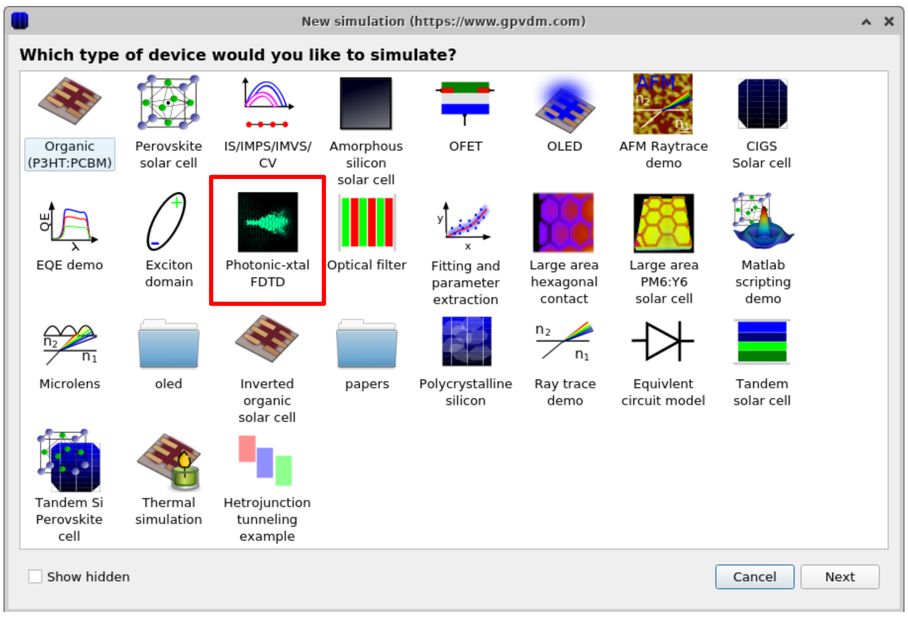
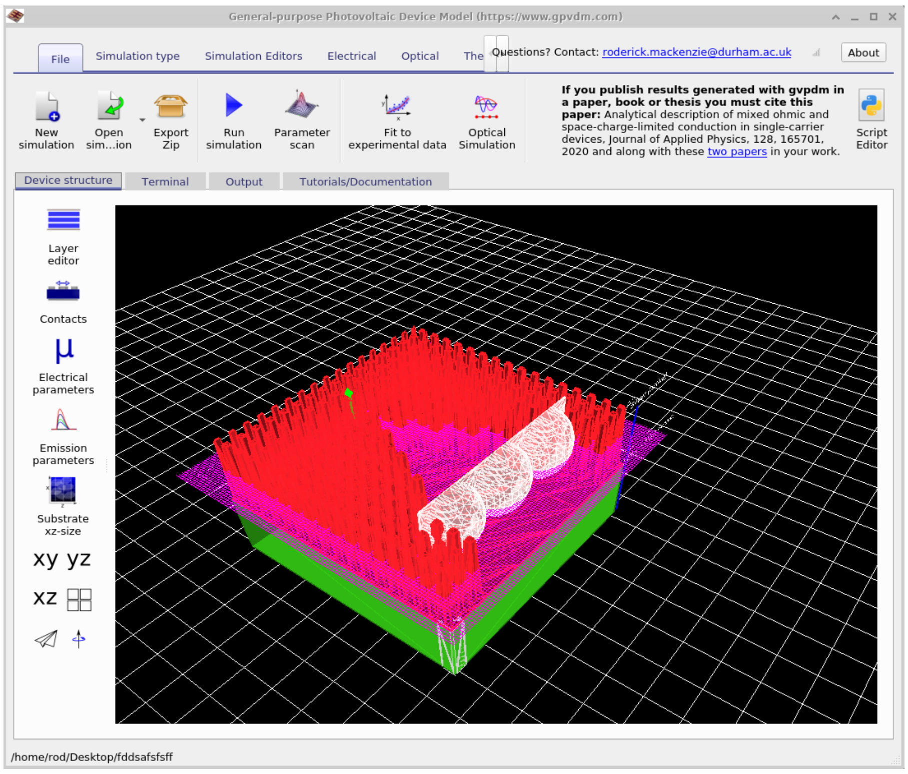


Figure 11.6: The new simulation window

Once you have opened the simulation you should get a window which appears figure 11.7. If you click on the play button the FDTD simulation will run, it will take around 30 seconds to run.



If you publish results generated with gvpdm in a paper, book or thesis you must cite this paper: Analytical description of mixed ohmic and space-charge-limited conduction in single-carrier devices, Journal of Applied Physics, 128, 165701, 2020 and along with these two papers in your work.

Figure 11.7: The initial FDTD simulation window. Use the slider to look at the results in time domain, and the drop down menu to select which field you are going to look at.

After the simulation has run click on the *Output* tab and 11.8 you will see the FDTD *snapshots* folder, this can be seen in figure 11.8. If you double click on this the FDTD snapshots window will appear which is shown in figure 11.9. This window will allow you to step through the simulations. If you click in the files to plot box, you will be able to select which field you plot. You will be able to select the *Ey*, *Ex* or *Ez* fields. In this case select the Ey field. Then use the slider bar to step through the field as a function of time.

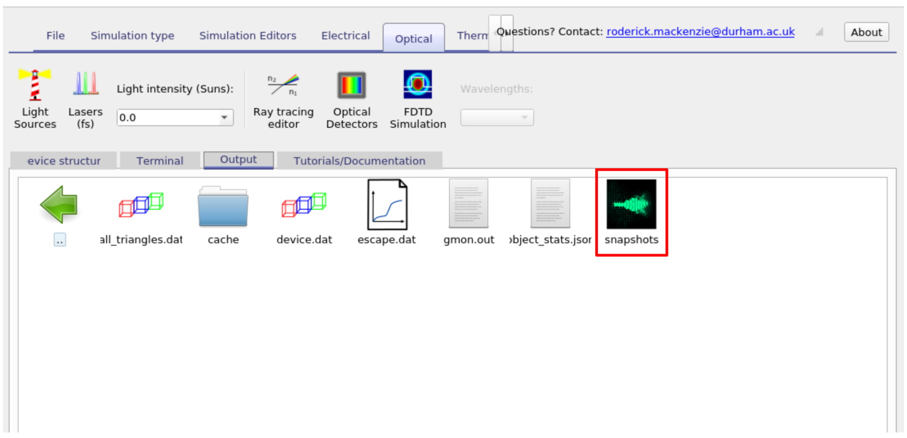


Figure 11.8: The output tab after having run an FDTD simulation, the key output is the Snapshots folder where the fields are stored.

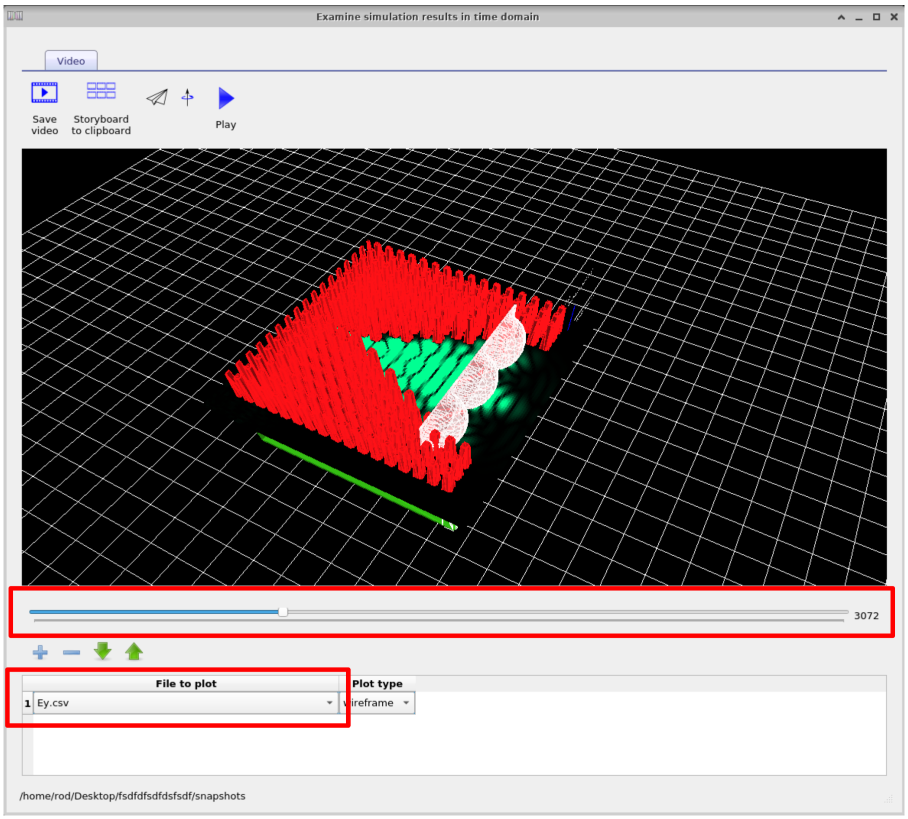

Examine simulation results in time domain
Video
Save video
Storyboard to clipboard
Play
3072
File to plot
Plot type
1
Ey.csv
wireframe
/home/rod/Desktop/fsdfdfsdfdsfsdf/snapshots

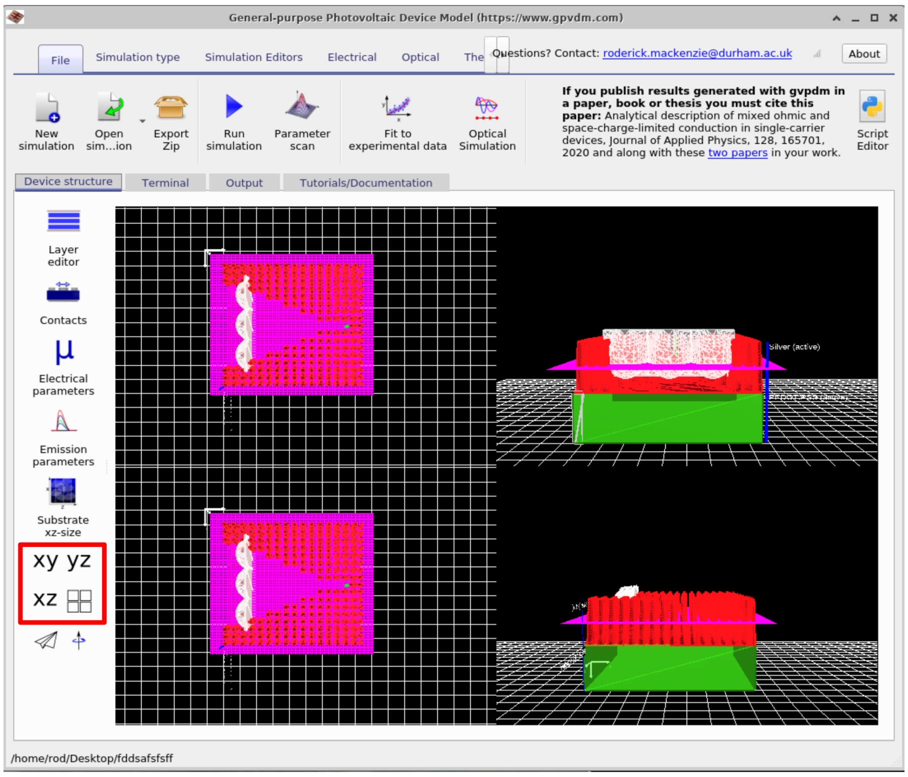
Figure 11.9: The FDTD snapshots window.

11.3.2 Manipulating objects in gpvdm

Now close the snapshot viewer and go back to the main simulation window and select the *Device tab*. On the left of the window, you will see four buttons *xy*, *yz*, *xz* and for little square boxes this can be seen in figure 11.10. Try clicking them to see what happens to the view of the device. After you have had a play select the *xz* option, so that the screen looks like the left hand side of 11.11. If you left click on the lenses, you will notice that you will be able to move them around. Try to move the lenses back in the device so that your device looks more like the right hand side of figure 11.10. If you hold shift down while dragging an object you can rotate it on the spot.

If you right click on the lenses and select *Edit* you will be able to bring up the object Editor. This shows the user all the object properties, this is visible in figure 11.12. Try changing the object type from *convex_lens* to *concave_lens* by clicking the edit button and rerunning the simulation. If you want to add your own shapes to the shape data base see section 10.5. Using this window you can also change the material which is used in the FDTD simulation, the color of the object as well as its position or rotational angle.

The *shape enabled* button enables you to turn off the shape if you don't want it in the simulation. If this shape were also electrically active you could also use this window to configure the electrical parameters.



If you publish results generated with gpvdm in a paper, book or thesis you must cite this paper: Analytical description of mixed ohmic and space-charge-limited conduction in single-carrier devices, Journal of Applied Physics, 128, 165701, 2020 and along with these two papers in your work.

/home/rod/Desktop/fddsafsfsff

Figure 11.10: Changing the object view in gpvdm

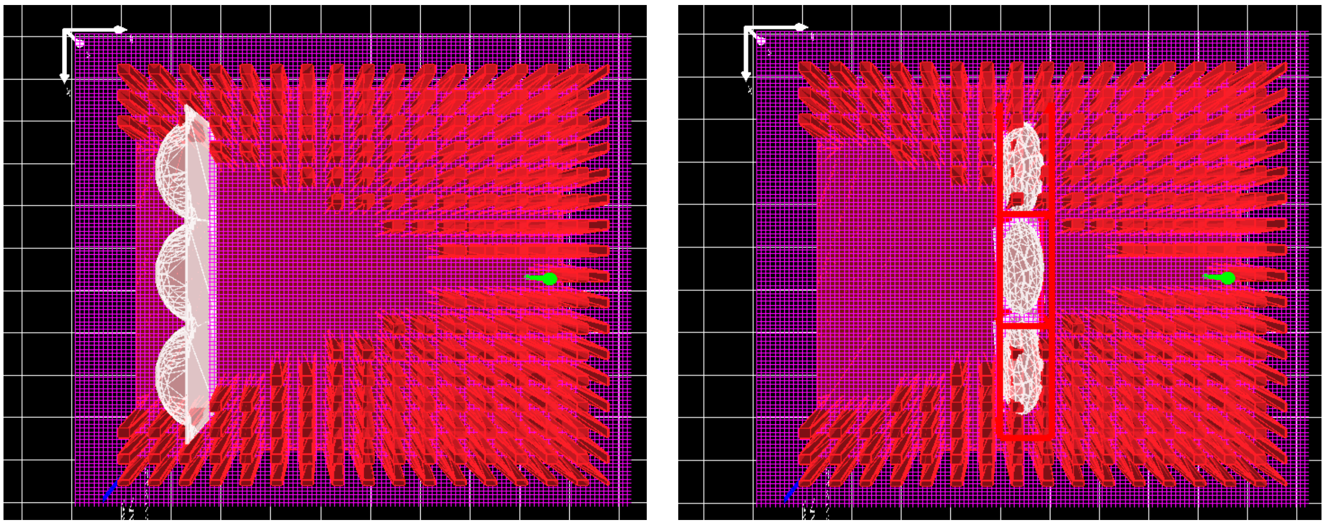
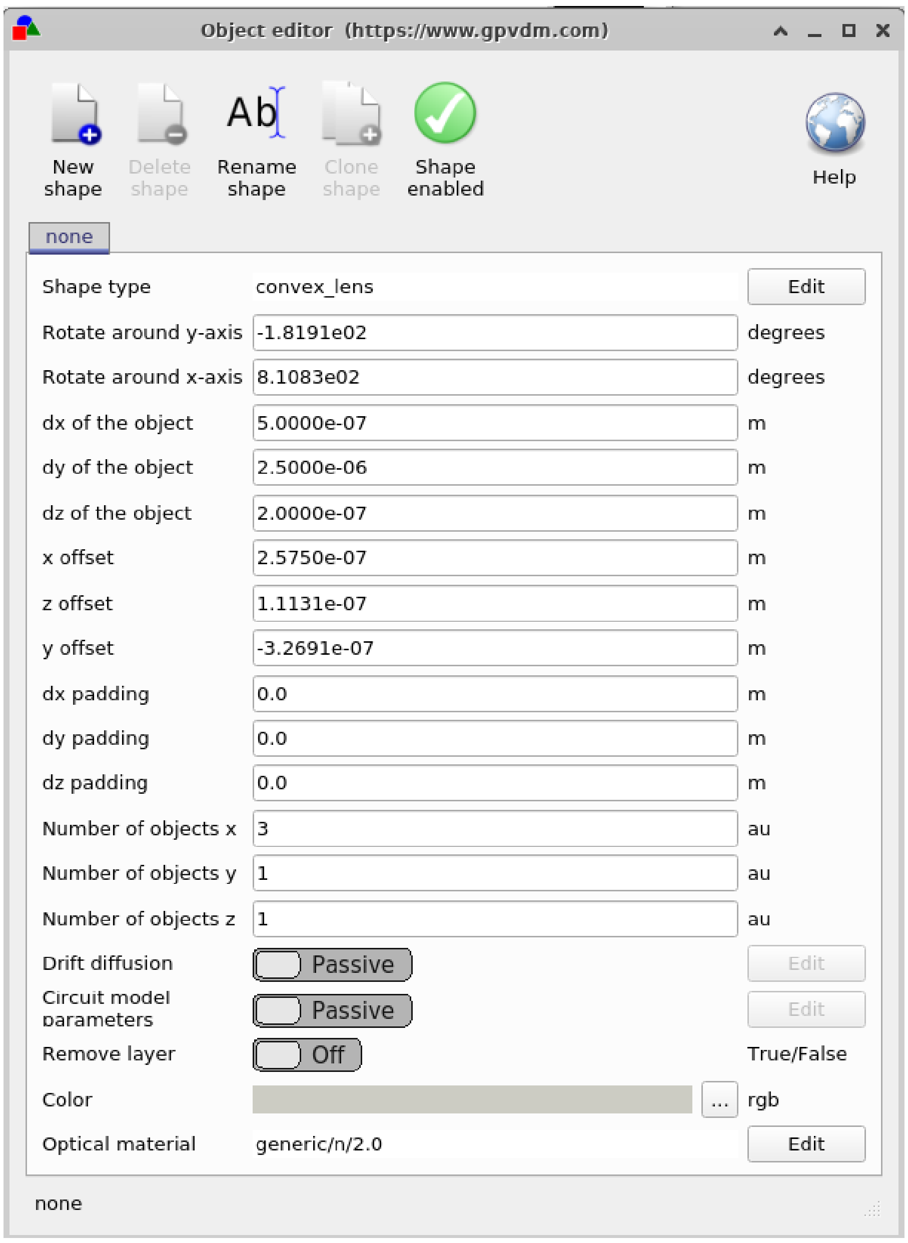


Figure 11.11: An example of moving objects in the simulation window.



Shape type	convex_lens	Edit
Rotate around y-axis	-1.8191e02	degrees
Rotate around x-axis	8.1083e02	degrees
dx of the object	5.0000e-07	m
dy of the object	2.5000e-06	m
dz of the object	2.0000e-07	m
x offset	2.5750e-07	m
z offset	1.1131e-07	m
y offset	-3.2691e-07	m
dx padding	0.0	m
dy padding	0.0	m
dz padding	0.0	m
Number of objects x	3	au
Number of objects y	1	au
Number of objects z	1	au
Drift diffusion	Passive	Edit
Circuit model parameters	Passive	Edit
Remove layer	Off	True/False
Color	...	rgb
Optical material	generic/n/2.0	Edit

none

Figure 11.12: The object viewer. This window is brought up by right clicking on an object and selecting *Edit*.

11.3.3 Manipulating light sources in gpvdm

Also visible in figure 11.11 is the light source given by the green arrow. Try moving this more forward in the device.

11.3.4 Theoretical background

References for this section are [5]. This section of the manual aims to describe the FDTD code in full with verbose derivations to help understanding/pick up errors.

Ampere's law is given as [5]

$$\sigma \boldsymbol{E} + \epsilon \frac{\partial \boldsymbol{E}}{\partial t} = \nabla \times \boldsymbol{H} = \begin{vmatrix} \hat{\boldsymbol{x}} & \hat{\boldsymbol{y}} & \hat{\boldsymbol{z}} \\ \frac{\partial}{\partial x} & \frac{\partial}{\partial y} & \frac{\partial}{\partial z} \\ H_x & H_y & H_z \end{vmatrix} \tag{11.15}$$

which can be expanded as

$$\sigma E_x + \epsilon \frac{\partial E_x}{\partial t} = \frac{\partial H_z}{\partial y} - \frac{\partial H_y}{\partial z} \tag{11.16}$$

$$\sigma E_y + \epsilon \frac{\partial E_y}{\partial t} = -\frac{\partial H_z}{\partial x} + \frac{\partial H_x}{\partial z} \tag{11.17}$$

$$\sigma E_z + \epsilon \frac{\partial E_z}{\partial t} = \frac{\partial H_y}{\partial x} - \frac{\partial H_x}{\partial y} \tag{11.18}$$

For the case $\frac{\partial}{\partial y} = 0$

$$\begin{aligned}
\sigma E_x + \epsilon \frac{\partial E_x}{\partial t} &= -\frac{\partial H_y}{\partial z} \\
\sigma E_y + \epsilon \frac{\partial E_y}{\partial t} &= -\frac{\partial H_z}{\partial x} + \frac{\partial H_x}{\partial z} \\
\sigma E_z + \epsilon \frac{\partial E_z}{\partial t} &= \frac{\partial H_y}{\partial x}
\end{aligned} \tag{11.19}$$

for E_x

$$\begin{aligned}
&\sigma E_x + \epsilon \frac{\partial E_x}{\partial t} = -\frac{\partial H_y}{\partial z} \\
&\sigma \frac{E_x^{t+1}[] + E_x^t[]}{2} + \epsilon \frac{E_x^{t+1}[] - E_x^t[]}{\Delta t} = -\frac{H_y^{t+\frac{1}{2}}[\frac{1}{2}] - H_y^{t+\frac{1}{2}}[-\frac{1}{2}]}{\Delta z} \\
&\sigma \frac{E_x^{t+1}[]}{2} + \epsilon \frac{E_x^{t+1}[]}{\Delta t} = -\frac{H_y^{t+\frac{1}{2}}[\frac{1}{2}] - H_y^{t+\frac{1}{2}}[-\frac{1}{2}]}{\Delta z} - \sigma \frac{E_x^t[]}{2} + \epsilon \frac{E_x^t[]}{\Delta t} \\
&\sigma \frac{E_x^{t+1}[]}{2} + \epsilon \frac{E_x^{t+1}[]}{\Delta t} = -\frac{H_y^{t+\frac{1}{2}}[\frac{1}{2}] - H_y^{t+\frac{1}{2}}[-\frac{1}{2}]}{\Delta z} - \sigma \frac{E_x^t[]}{2} + \epsilon \frac{E_x^t[]}{\Delta t} \\
&\frac{\sigma \Delta t + 2\epsilon}{2\Delta t} E_x^{t+1}[] = -\frac{H_y^{t+\frac{1}{2}}[\frac{1}{2}] - H_y^{t+\frac{1}{2}}[-\frac{1}{2}]}{\Delta z} - \sigma \frac{E_x^t[]}{2} + \epsilon \frac{E_x^t[]}{\Delta t} \\
&E_x^{t+1}[] = \left(-\frac{H_y^{t+\frac{1}{2}}[\frac{1}{2}] - H_y^{t+\frac{1}{2}}[-\frac{1}{2}]}{\Delta z} - \sigma \frac{E_x^t[]}{2} + \epsilon \frac{E_x^t[]}{\Delta t} \right) \frac{2\Delta t}{\sigma \Delta t + 2\epsilon}
\end{aligned} \tag{11.20}$$

for E_y

$$\begin{aligned}
&\sigma E_y + \epsilon \frac{\partial E_y}{\partial t} = -\frac{\partial H_z}{\partial x} + \frac{\partial H_x}{\partial z} \\
&\sigma \frac{E_y^{t+1}[] + E_y^t[]}{2} + \epsilon \frac{E_y^{t+1}[] - E_y^t[]}{\Delta t} = -\frac{H_z^{t+\frac{1}{2}}[\frac{1}{2}] - H_z^{t+\frac{1}{2}}[-\frac{1}{2}]}{\Delta x} + \frac{H_x^{t+\frac{1}{2}}[\frac{1}{2}] - H_x^{t+\frac{1}{2}}[-\frac{1}{2}]}{\Delta z} \\
&\sigma \frac{E_y^{t+1}[]}{2} + \epsilon \frac{E_y^{t+1}[]}{\Delta t} = -\frac{H_z^{t+\frac{1}{2}}[\frac{1}{2}] - H_z^{t+\frac{1}{2}}[-\frac{1}{2}]}{\Delta x} + \frac{H_x^{t+\frac{1}{2}}[\frac{1}{2}] - H_x^{t+\frac{1}{2}}[-\frac{1}{2}]}{\Delta z} - \sigma \frac{E_y^t[]}{2} + \epsilon \frac{E_y^t[]}{\Delta t} \\
&E_y^{t+1}[] = \left(-\frac{H_z^{t+\frac{1}{2}}[\frac{1}{2}] - H_z^{t+\frac{1}{2}}[-\frac{1}{2}]}{\Delta x} + \frac{H_x^{t+\frac{1}{2}}[\frac{1}{2}] - H_x^{t+\frac{1}{2}}[-\frac{1}{2}]}{\Delta z} - \sigma \frac{E_y^t[]}{2} + \epsilon \frac{E_y^t[]}{\Delta t} \right) \frac{2\Delta t}{\sigma \Delta t + 2\epsilon}
\end{aligned} \tag{11.21}$$

for E_z

$$\begin{aligned}
&\sigma E_z + \epsilon\frac{\partial E_z}{\partial t} = \frac{\partial H_y}{\partial x} \\
&\sigma\frac{E_z^{t+1}[] + E_z^t[]}{2} + \epsilon\frac{E_z^{t+1}[] - E_z^t[]}{\Delta t} = \frac{H_y^{t+\frac{1}{2}}[\frac{1}{2}] - H_y^{t+\frac{1}{2}}[-\frac{1}{2}]}{\Delta x} \\
&\sigma\frac{E_z^{t+1}[]}{2} + \epsilon\frac{E_z^{t+1}[]}{\Delta t} = \frac{H_y^{t+\frac{1}{2}}[\frac{1}{2}] - H_y^{t+\frac{1}{2}}[-\frac{1}{2}]}{\Delta x} - \sigma\frac{E_z^t[]}{2} + \epsilon\frac{E_z^t[]}{\Delta t} \\
&E_z^{t+1}[] = \left(\frac{H_y^{t+\frac{1}{2}}[\frac{1}{2}] - H_y^{t+\frac{1}{2}}[-\frac{1}{2}]}{\Delta x} - \sigma\frac{E_z^t[]}{2} + \epsilon\frac{E_z^t[]}{\Delta t}\right)\frac{2\Delta t}{\sigma\Delta t + 2\epsilon}
\end{aligned} \tag{11.22}$$

Faraday's law is given as [5]

$$-\sigma_m\boldsymbol{H} - \mu\frac{\partial \boldsymbol{H}}{\partial t} = \nabla\times\boldsymbol{E} = \begin{vmatrix} \hat{\boldsymbol{x}} & \hat{\boldsymbol{y}} & \hat{\boldsymbol{z}} \\ \frac{\partial}{\partial x} & \frac{\partial}{\partial y} & \frac{\partial}{\partial z} \\ E_x & E_y & E_z \end{vmatrix} \tag{11.23}$$

which can be expanded to give:

$$-\sigma_m H_x - \mu\frac{\partial H_x}{\partial t} = \frac{\partial E_z}{\partial y} - \frac{\partial E_y}{\partial z} \tag{11.24}$$

$$-\sigma_m H_y - \mu\frac{\partial H_y}{\partial t} = -\frac{\partial E_z}{\partial x} + \frac{\partial E_x}{\partial z} \tag{11.25}$$

$$-\sigma_m H_z - \mu\frac{\partial H_z}{\partial t} = \frac{\partial E_y}{\partial x} - \frac{\partial E_x}{\partial y} \tag{11.26}$$

With $\sigma_m = 0$ and $\frac{\partial}{\partial y} = 0$

$$\begin{aligned}
\frac{\partial H_x}{\partial t} &= \frac{1}{\mu}\left(\frac{\partial E_y}{\partial z}\right) \\
\frac{\partial H_y}{\partial t} &= \frac{1}{\mu}\left(\frac{\partial E_z}{\partial x} - \frac{\partial E_x}{\partial z}\right) \\
\frac{\partial H_z}{\partial t} &= -\frac{1}{\mu}\left(\frac{\partial E_y}{\partial x}\right)
\end{aligned} \tag{11.27}$$

which discretizing gives

$$\begin{aligned}
H_x^{t+1} &= \frac{1}{\mu}\left(\frac{E_y^{t+\frac{1}{2}}[\frac{1}{2}] - E_y^{t+\frac{1}{2}}[-\frac{1}{2}]}{\Delta z}\right)\Delta t + H_x^t[] \\
H_y^{t+1} &= \frac{1}{\mu}\left(\frac{E_z^{t+\frac{1}{2}}[\frac{1}{2}] - E_z^{t+\frac{1}{2}}[-\frac{1}{2}]}{\Delta x} - \frac{E_x^{t+\frac{1}{2}}[\frac{1}{2}] - E_x^{t+\frac{1}{2}}[-\frac{1}{2}]}{\Delta z}\right)\Delta t + H_y^t[] \\
H_z^{t+1} &= \frac{1}{\mu}\left(-\frac{E_y^{t+\frac{1}{2}}[\frac{1}{2}] - E_y^{t+\frac{1}{2}}[-\frac{1}{2}]}{\Delta x}\right)\Delta t + H_x^z[]
\end{aligned} \tag{11.28}$$

11.3.5 Ray tracing model

Add text.

Chapter 12

Fitting experimental data

Related YouTube videos:

- Advanced topics in fitting of JV curves to experimental data using gpvdm.
- Fitting transient photocurrent (TPC) and light JV curves using gpvdm
- Fitting the light JV curve of an ultra large area (2.5meter x 1cm) OPV device using gpvdm

Chapter 13

Automating/Scripting the model

There are three main ways to automate the mode. The first is the parameter scan window, see section 13.1. This alows the user vary a paramter in steps. The second way is by using Python scripting, see section 13.2.1 and the third way is through matlab scripting see section 13.2.2.

13.1 The parameter scan window

Sometimes one wishes to systematically vary a simulation parameter, to do this first bring up the parameter scan window, this can be done by clicking on the

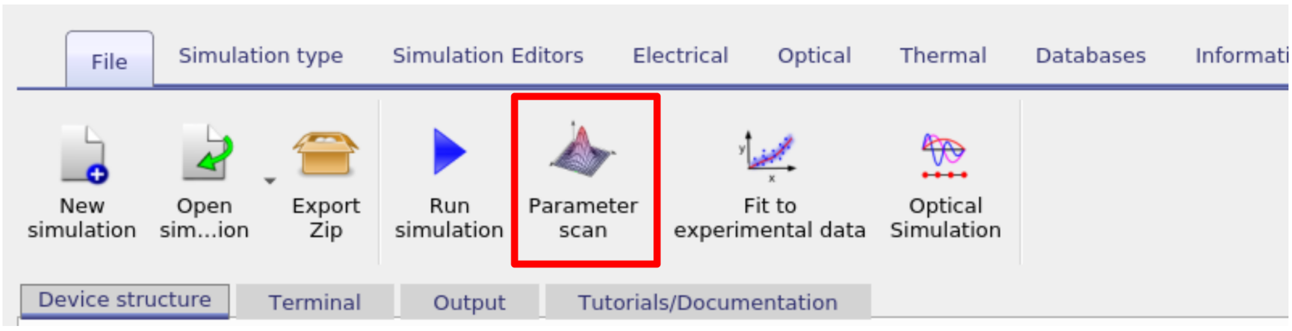


Figure 13.1: Step 1: Select the 'Parameter scan' tool.

Then make a new scan by clicking on the new button (1) then open the new scan by double clicking on the icon representing the scan (2). See figure 13.2. This will bring up the scan window, see figure 13.3.

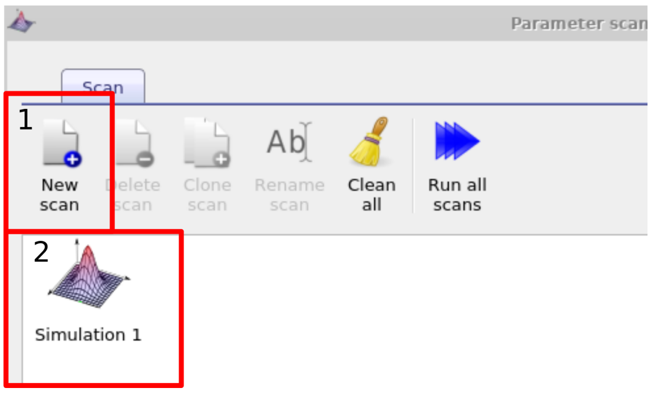


Figure 13.2: Step 2: Make a new parameter scan, then double click on it to open it.

13.1.1 Changing one material parameter

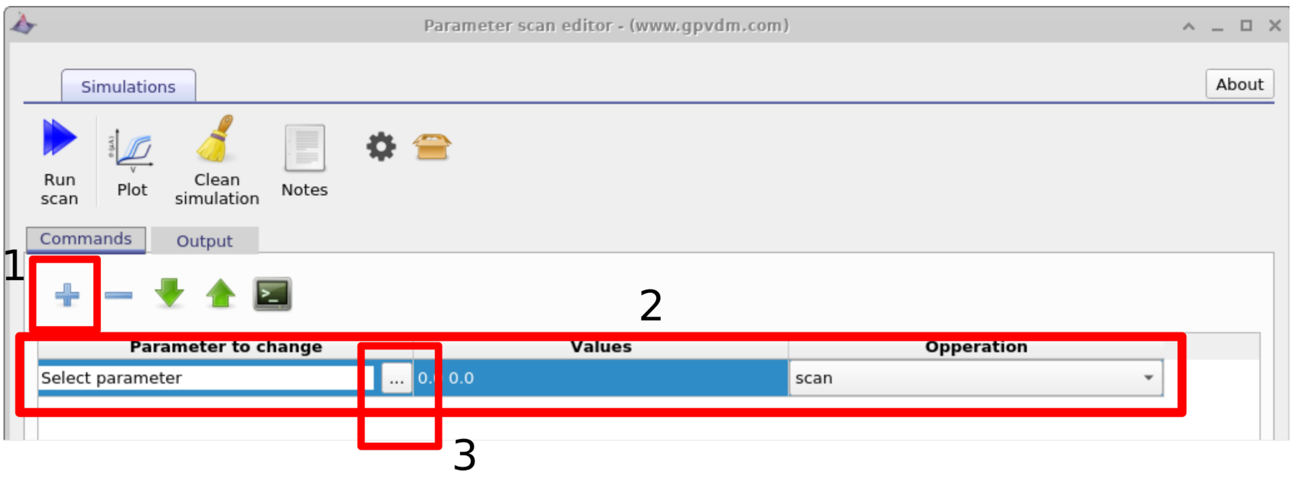
Once the scan window has appeared. Make a new scan line by clicking on the the plus icon 13.3 (1), then select this line so that it is highlighted (2), then click on the three dots (3) to select which parameter you want to scan. In this example we will be selecting the electron mobility of a P3HT:PCBM solar cell. Do this by navigating to epitaxy→ P3HT:PCBM→ Drift diffusion→ Electron mobility y. Highlight the parameter and then click OK. This should then appear in the scan line. The meaning of ”epitaxy→ P3HT:PCBM→ Drift diffusion¿Electron mobility y” will now be explained below:

- epitaxy: All parameters in the .gpvdm file are exposed via the parameter selection window see 13.4. This file is a tree structure, see 9.1. The all parameters which define the device it's self are contained under epitaxy.

- P3HT:PCBM: Under epitaxy each layer of the device is given by its name. The active layer in this device is called P3HT:PCBM.

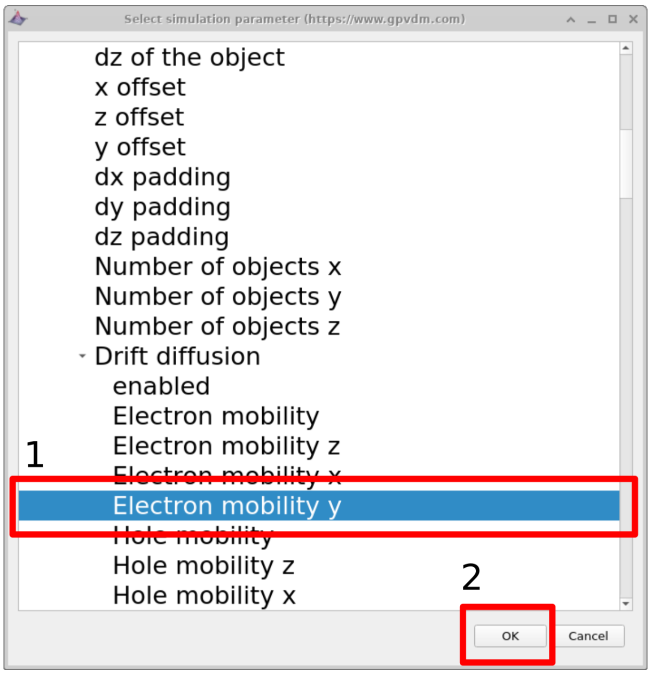
- Drift diffusion: All electrical parameters are stored under drift diffusion.

- Electron mobility y: One can define asymmetric mobilities in the z,x and y direction - this is useful for OFET simulations. However by default the model assumes a symmetric mobility which is the same in all directions. This value is defined by ”Electron mobility y”.



Parameter to change	Values	Opperation
Select parameter	0.0	scan

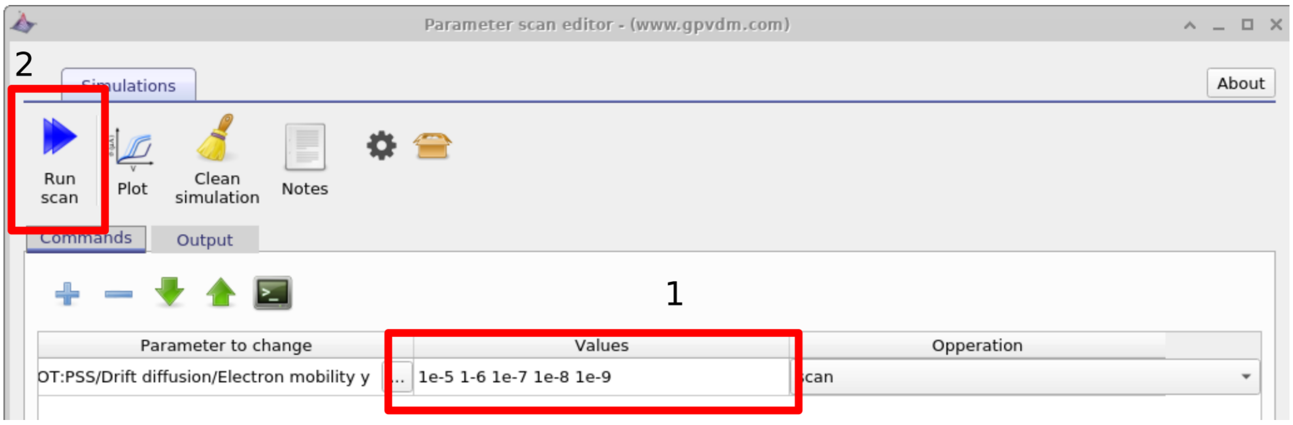
Figure 13.3: Step 3: Add a 'scan line' to the scan.


Select simulation parameter (https://www.gpvdm.com)

dz of the object
x offset
z offset
y offset
dx padding
dy padding
dz padding
Number of objects x
Number of objects y
Number of objects z
Drift diffusion
enabled
Electron mobility
Electron mobility z
Electron mobility x
Electron mobility y
Hole mobility
Hole mobility z
Hole mobility x

Figure 13.4: Step 5: Select the parameter you want to scan in the parameter selection window, in this case we are selecting epitaxy→ P3HT:PCBM→ Drift diffusion→ Electron mobility y.

Next enter the values of mobility which you want to scan over in this case we will be entering ”1e-5 1-6 1e-7 1e-8 1e-9” (see figure 13.5 1) then click ”run scan” (see figure 13.5 2). Gpvdm will run one simulation on each core of your computer until all the simulations are finished.


Parameter scan editor - (www.gpvdm.com)

Parameter to change	Values	Opperation
OT:PSS/Drift diffusion/Electron mobility y	1e-5 1-6 1e-7 1e-8 1e-9	scan

Figure 13.5: Step 6: Enter the input values of mobility (or other values) you want to scan over (1). Then run the simulations.

To view the simulation results click on the ”output” tab this will bring up the simulation outputs, see figure 13.6. You can see that a directory has been created for each variable that we scanned over so 1e-5, 1-6, 1e-7, 1e-8 and 1e-9. If you look inside each directory it will be an exact copy of the base simulation directory. If you double click on the files with multi colored JV curves, see the red box in figure 13.6. Gpvdm will automaticity plot all the curves from each simulation in one graph, see figure 13.7.

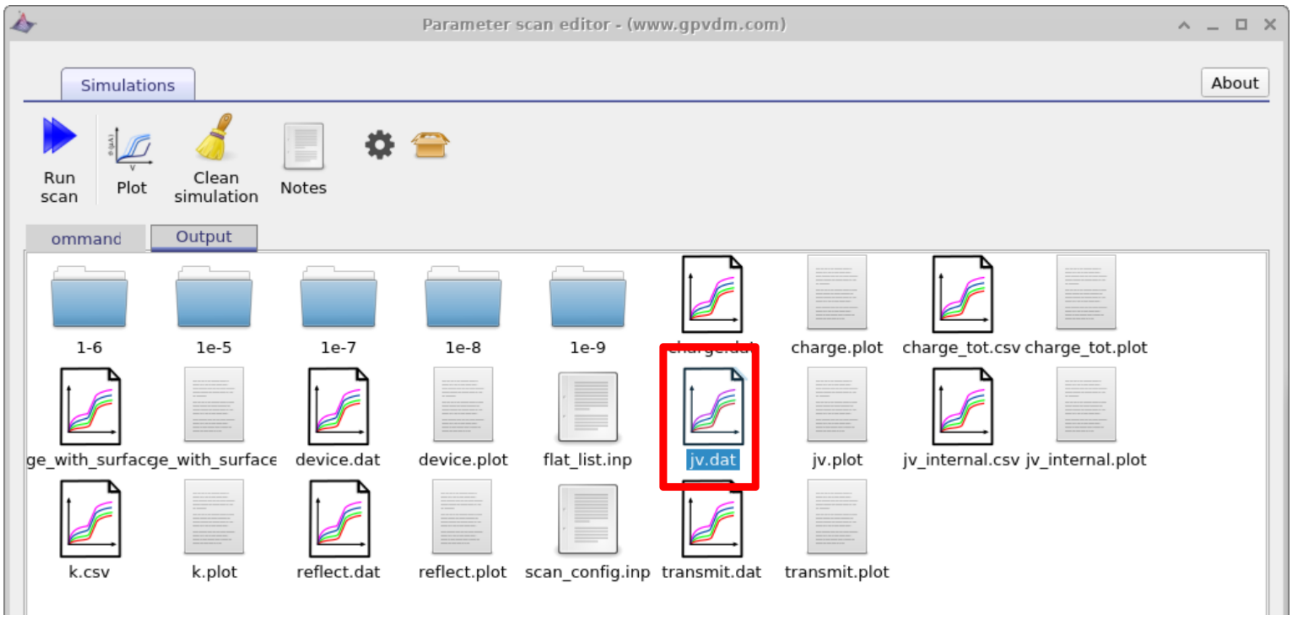


Figure 13.6: Step 7: The output tab.

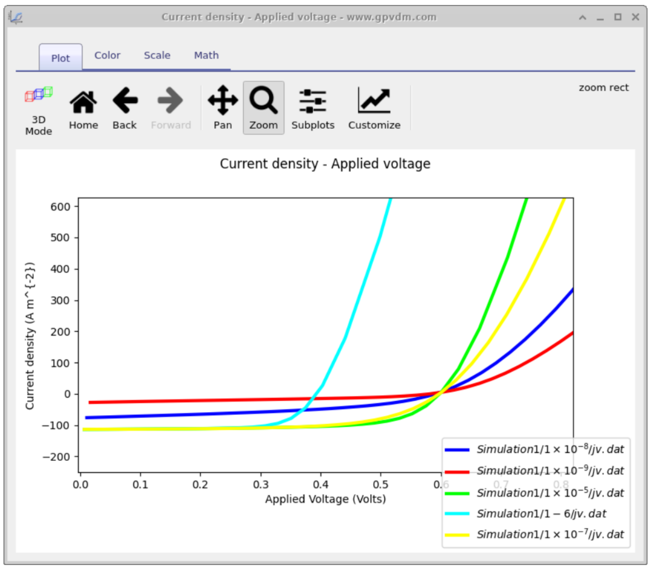


Figure 13.7: Step 8: The result of the mobility scan.

13.1.2 Duplicating parameters - changing the thickness of the active layer

Very often one wants to change a parameter, then set another parameter equal to the parameter which was changed. An example of this is one may want to change electron and hole mobilities together when simulating a device with symmetric mobilities. This can be done using the duplicate function of the scan window as seen in figure 13.8. In this example we tackle a slightly more trick problem than changing mobilities together we are going to change the physical width of the active layer and at the same time adjust the electrical mesh to make it match. As discussed in section 14 the width of the active layer must always match the width of the electrical mesh. When you change the layer width by hand in the layer editor gvpdm updates the width of the electrical mesh for you. But when scripting the model it won't do this update for you. Therefore in the example below we are going to set the width of the active layer by scanning over:

epitaxy→P3HT:PCBM→dy of the object

Then we are going to add another line under and under parameter to scan select

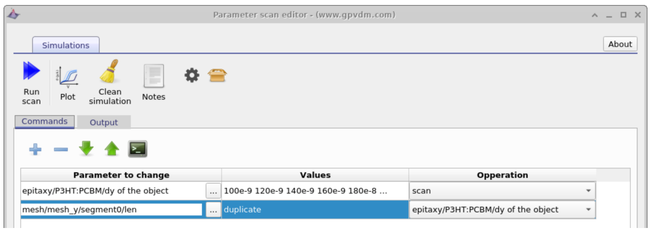
mesh→mesh_y→segment0→len

and set it to

epitaxy→P3HT:PCBM→dy of the object

under the operation dropdown box. You will see the word duplicate appear under values.

If you now run the simulation "epitaxy→P3HT:PCBM→dy of the object" will be changed and "mesh→mesh_y→segment0→len" will follow it.



Parameter to change	Values	Opperation
epitaxy/P3HT:PCBM/dy of the object	100e-9 120e-9 140e-9 160e-9 180e-8 ...	scan
mesh/mesh_y/segment0/len	duplicate	epitaxy/P3HT:PCBM/dy of the object

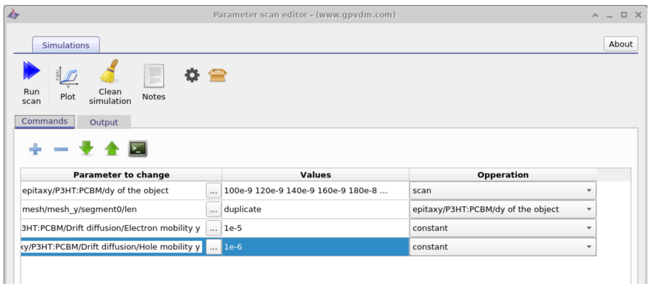
Figure 13.8: Duplicating material paramters.

Side note: Device with multiple active layers

The sum of the active layer thickness (as defined in the layer editor) MUST equal the electrical mesh thickness (more about the mesh in section 14). If for example one had three active layers TiO2 (100 nm)/Perovskite (200 nm)/Spiro (100 nm) with a total width of 400 nm. The total mesh length must be 400 nm as well. Therefore were one want to change the thickness of the perovskite layer as in 13.8 one would have to break the electrical mesh up into three sections and make sure you were updating the mesh segment referring to the perovskite layer alone.

13.1.3 Setting constants

Often when running a parameter scan one wants to set a constant value, this can be done using the "constant" option in the Operations dropdown menu. See figure 13.9



Parameter to change	Values	Opperation
epitaxy/P3HT:PCBM/dy of the object	100e-9 120e-9 140e-9 160e-9 180e-8 ...	scan
mesh/mesh_y/segment0/len	duplicate	epitaxy/P3HT:PCBM/dy of the object
3HT:PCBM/Drift diffusion/Electron mobility y	1e-5	constant
xy/P3HT:PCBM/Drift diffusion/Hole mobility y	1e-6	constant

Figure 13.9: The result of the mobility scan.

13.1.4 The equivalent of loops

Often when scanning over a parameter range one may want to simulate so many parameters that it is not practical to type them in. In this case gpvdm has the equivalent of a loop. So for

Python scripting perovskite solar cell simulation

example if one wanted to change a value from 100 to 400 in steps of 1, one could type

```
[100 400 1]
```

Listing 2: JSON example

13.1.5 Limitations of the scan window

Although the scan window is convenient in that it provides a quick way to scan simulation parameters, it is by nature rather limited in terms of flexibility. If you want to do complex scans were multiple parameters are changed or to programmatically collect data from each simulation then you can use the or matlab interfaces to gpvdm. These are described in the next section.

13.2 Python/MATLAB scripting of gpvdm

Scripting offers a more powerful way to interact with gvpdm. Rather than using the graphical user interface, you can use your favourite programming language to interact with gpvdm. This gives you the option to drive gpvdm in a far more powerful way than can be done using the graphical interface alone. Below I give examples of using MATLAB and python to drive gpvdm, but you can use any language you want which has a json reader/writer. Pearl and Java are two languages which spring to mind.

Before you begin scripting gpvdm, open up the install path of gpvdm, the default gpvdm will be installed to C:\Program files x86 \gpvdm, in there you will see in this directory there are two windows executables, one called *gpvdm.exe*, this is the graphical user interface, and a second .exe, called *gpvdm_core.exe*. You can run *gpvdm_core.exe* from the command line without *gpvdm.exe*. You simply need to navigate to a directory containing a *sim.gpvdm* folder and call *gpvdm_core.exe*, this can be done from the windows command line, matlab, python or any other scripting language. However, before you can do this on windows, you need to add C:\Program files x86 \gpvdm to your windows path so that windows knows where gpvdm is instaled. An example of how to do this on a modern version of windows is given in the link `https://docs.microsoft.com/en-us/previous-versions/office/developer/sharepoint-2010/ee537574(v=office.14)`

Every new version of windows seems to move the configuration options around, so you may have to find instructions for your version of windows.

13.2.1 Python scripting

Related YouTube videos:

There are two ways to interact with .gpvdm files via python, using native python commands or by using the gpvdm class structures, examples of both are given below.

```
import json
import os
import sys

f=open('sim.json')          #open the sim.json file
lines=f.readlines()
f.close()
lines="".join(lines)    #convert the text to a python json object
data = json.loads(lines)

#Edit a value (use firefox as a json viewer
# to help you figure out which value to edit)
# this time we are editing the mobility of layer 1
data['epitaxy']['layer1']['shape_dos']['mue_y']=1.0


#convert the json object back to a string
jstr = json.dumps(data, sort_keys=False, indent='\t')

#write it back to disk
f=open('sim.json',"w")
f.write(jstr)
f.close()

#run the simulation using gpvdm_core
os.system("gpvdm_core.exe")
```

Listing 3: Manipulating a sim.json file with python and running a gpvdm simulation.

The native python way

As described in section 9.1, .gpvdm files are simply json files zipped up in an archive. If you extract the sim.json file form the sim.gpvdm file you can use Python's json reading/writing code to edit the .json config file directly, this is a quick and dirty approach which will work. You can then use the *os.system* call to run *gpvdm_core.exe* to execute gpvdm.

For example were one to want to change the mobility of the 1st device layer to 1.0 and then run a simulation you would use the code listed in listing 3.

If the simulation in sim.json is setup to run a JV curve, then a file called sim_data.dat will be written to the simulation directory containing paramters such as PCE, fill factor, J_{sc} and V_{oc}. This again is a raw json file, to read this file in using python and write out the value of V_oc to a second file use the code given in listing 4.

Using gpvdm's built in classes for reading and writing json

Gpvdm has a set of classes that can read in gpvdm files and write them to disk. The difference between using python's native commands and the gpvmd classes is that, gpvdm will convert the json save files to a hierarchical tree of python classes rather than leaving them as raw json. So for example using Python's native json interpreters one would write:

but using the gpvdm interpreter one would write

```
f=open('sim_info.dat')
lines=f.readlines()
f.close()
lines="".join(lines)
data = json.loads(lines)

f=open('out.dat',"a")
f.write(str(data["Voc"])+"\n");
f.close()
```

Listing 4: Reading in a sim_data.dat file using Python's native json reader.

```
    data['epitaxy']['layer1']['shape_dos']['mue_y']=1.0
```

Listing 5: Reading in a sim_data.dat file using Python's native json reader.

The gpvdm class tree also has embedded functions for searching for objects and alike some of which are described below in listing 7.

Running gpvdm across multiple cores

The scan window by default uses gpvdm's built in job scheduler so that if you want to scan across 10 parameters and have a CPU with multiple cores, the jobs will be spread across all cores. This increases the overall speed of the simulations. You can access this API using the gpvdm_api class, an example of how to do this is given in listing 8.

```
    data.epitaxy.layer[1].shape_dos.mue_y=1.0
```

Listing 6: Reading in a sim_data.dat file using Python's native json reader.

```
#!/usr/bin/env python3
import json
import os
import sys

sys.path.append('c:\Program files x86\gpvdm\modules')
from gpvdm_json import gpvdm_data

data=gpvdm_data()
data.load("sim.json")
data.epitaxy.layer[1].shape_dos.mue_y=1.0
data.save()

os.system("gpvdm_core.exe")
```

Listing 7: Editing sim.json files using gpvdm's built in classes.

13.2.2 MATLAB scripting

As described in section 9.1 gpvdm simulations are stored in .json files zipped up inside a zip archive. Matlab has both a zip decompressor and a json decoder. Therefore it is straight forward to edit and read and edit .gpvdm files in MATLAB. You can then use MATLAB to perform quite complex parameter scans. The example script below in listing 9 demonstrates how to run multiple simulations with mobilities ranging from 1e-7 to 1e-5 $m^2V^{-1}s^{-1}$). The script starts off by unzipping the sim.json file, if you already have extracted your sim.json file from the sim.gpvdm file you don't need these lines. The code then reads in sim.json using the MATLAB json decoder $jsondecode$. A new directory is made which corresponds to the mobility value, the sim.gpvdm file copied into that directory. Then $json\_data.epitaxy.layer0.shape\_dos.mue_y$ is set to the desired value of mobility and the simulation saved using $jsonencode$ and $fopen$, $fprintf$, $fclose$. The *system* call is then used to run *gpvdm_core.exe* to perform the simulation. Out put parameters such as J_{sc} are stored in sim_data.dat again in json format, see section 15.6.1, although this is not done in this simple script.

```
#!/usr/bin/env python3
import os
import sys
sys.path.append('c:\Program files x86\gpvdm\modules')

from gpvdm_api import gpvdm_api

#initialize the API
api=gpvdm_api(verbose=False)

#Use the name of the current script to determine the directory name to make
script_name=os.path.basename(__file__).split(".")[0]

#define the name of the simulation dir
scan_dir=os.path.join(os.getcwd(),script_name)

#make the simulation dir
api.mkdir(scan_dir)                       #make a new scandir

#tell the API where we are going to run the simulation
api.server.server_base_init(scan_dir)

#Loop over electron and hole mobilities.
for mue in [ 1e-5, 1e-6, 1e-7, 1e-8]:
    for muh in [ 1e-5, 1e-6, 1e-7, 1e-8 ]:
        #define the sub sim path
        sim_path=os.path.join(scan_dir,"{:.2e}".format(mue),"{:.2e}".format(muh))
        #make the directory
        api.mkdir(sim_path)

        #clone the current sim dir to the new dir
        api.clone(sim_path,os.getcwd())

        #make edit the newly generated sim.json file
        data=gpvdm_data()
        data.load(os.path.join(sim_path,"sim.json"))
        data.epitaxy.layer[1].shape_dos.mue_y=mue
        data.epitaxy.layer[1].shape_dos.muh_y=muh
        data.save()

        #Add the path to the job list
        api.add_job(path=sim_path)

#run all the jobs over multiple CPUs
api.server.simple_run()

#Generate GNUPLOT compatible files for plotting the results together.
api.build_multiplot(scan_dir,gnuplot=True])
```

Listing 8: Running jobs across multiple CPUs using python

```
if exist("sim.gpvmd", 'file')==false
 sprintf("No sim.gpvdm file found"); %Check if we have a sim.gpvdm file
end

if exist("sim.json", 'file')==false
 unzip("sim.gpvdm")  %if we don't have a sim.json file
                     %try to extract it
end

A = fileread("sim.json");  %Read the json file.
json_data=jsondecode(A);  %Decode the json file

mobility=1e-7  %Start mobility
origonal_path=pwd  %working with the current dir
base_dir="mobility_scan"  %output directory name
while(mobility<1e-5)
    dir_name=sprintf("%e",mobility);
    full_path=fullfile(origonal_path,base_dir,dir_name)    %join paths
    mkdir(full_path)  %make the dir
    cd(full_path)  %cd to the dir

    %Update the json mobility
    json_data.epitaxy.layer0.shape_dos.mue_y=mobility  %Change mobility
                                                       %of layer0

    copyfile(fullfile(origonal_path,"sim.gpvdm"),\\
        fullfile(origonal_path,base_dir,dir_name,"sim.gpvdm"))

    %now write the json file back to disk
    out=jsonencode(json_data);
    json_data
    fid = fopen("sim.json",'w');
    fprintf(fid, '%s', out);
    fclose(fid);

    %run gpvdm - This won't work if you have not added the gpvdm
    %install directory to your windows paths
    system("gpvdm_core.exe")

    %Multiply mobility by 10
    mobility=mobility*10;
end

%Move back to the original dir
cd(origonal_path)
```

Listing 9: An example of how to call gpvdm from MATLAB

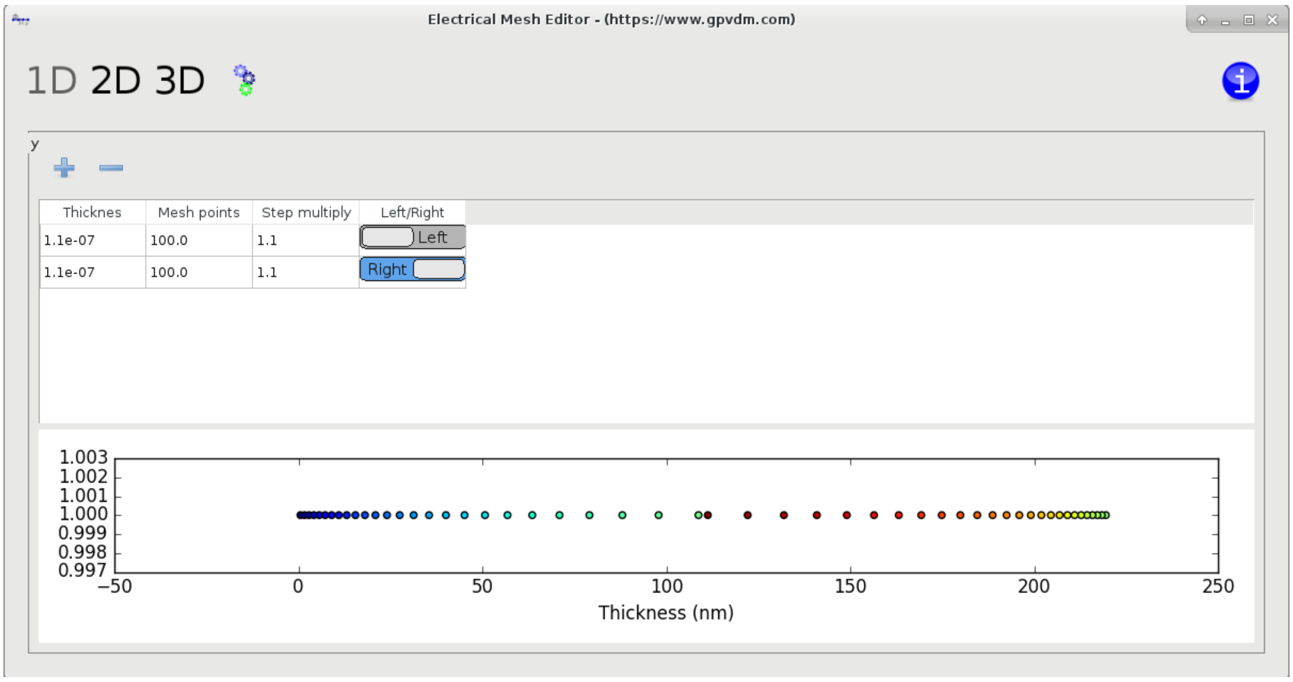
Chapter 14

Meshing

14.1 Editing the electrical mesh/layers

The device structure is split up into layers of different materials. These can be configured in layer editor which is discussed in section 2.1.1. Some of these layers will have the layer type 'active'. An 'active' layer is a layer over which the electrical model will be applied. The electrical model needs a finite difference mesh to to be setup for it to work. Usually, this will be take care of automatically, by gpvdm. However, some users will want fine control over the mesh. This section describes how to do that. The electrical mesh editor is depicted in figure 14.1.

The buttons marked 1D, 2D and 3D at the top of the window can be used to toggle the simulation between 1D, 2D and 3D modes. (Note, if you want to do 2D or 3D simulations you are best off using a default 2D simulation, such as the OFET simulation. This is because to do 2D/3D simulations, a special newton solver configuration will be needed.) The table on the left hand side is used to configure the mesh. The sum of the mesh layer thicknesses must exactly match that of the sum of the active layers. If this is not the case, the model will give an error message. The columns thickness and mesh points, determine the thickness of the mesh layer and the number of points on the mesh layer, if there is a uniform spacing between mesh points. The column, 'step multiply' by how much to grow each step. In this example, the mesh spacing is increased by a factor of 0.1 each step. The toggle button left/right, defines on which side the mesh layer is generated. In this example there are two mesh layers, one starting on the left and one starting on the right. The resulting mesh is plotted in the graph at the bottom of the window. It can be seen that a non-linear mesh has been generated.



Thicknes	Mesh points	Step multiply	Left/Right
1.1e-07	100.0	1.1	Left
1.1e-07	100.0	1.1	Right

Figure 14.1: The electrical mesh editor

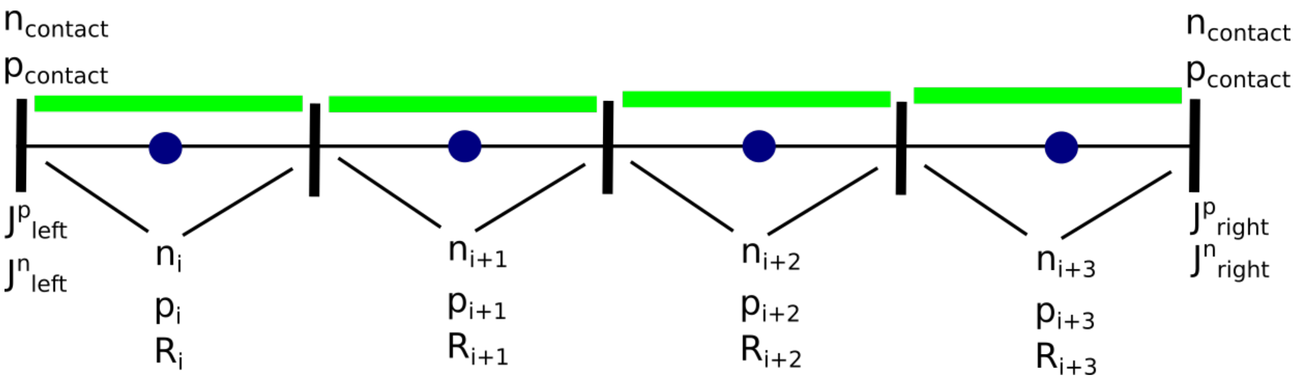


Figure 14.2: A 1D diagram of the mesh

14.2 Should I be simulating in 1D, 2D or 3D?

When deciding if you should perform 1D, 2D or 3D, simulations, consider the dimensionality of your problem. For example if you consider a solar cell, it is only a few micros thick, and there is rapid variation in the structure, charge densities, mobilities, and doping as a function of depth (y). However, the structure will not vary very in the lateral (xz) plane. Therefore, in general to capture all interesting effects present within a solar cell one only needs a 1D model. If one now considers OFETs, there is both vertical an lateral current flow, therefore one can not get away with a 1D model any more, as one must simulate both vertical current flow, and current between the source and the drain, thus one needs a 2D simulation. As the number of dimensions increases, computation speed will decrease, therefore my general advice is to use the minimum number of dimensions possible to solve your problem.

In short try to make your simulation as simple as possible as it will save you time and effort. Generally the following geometries could be used for various types of devices:

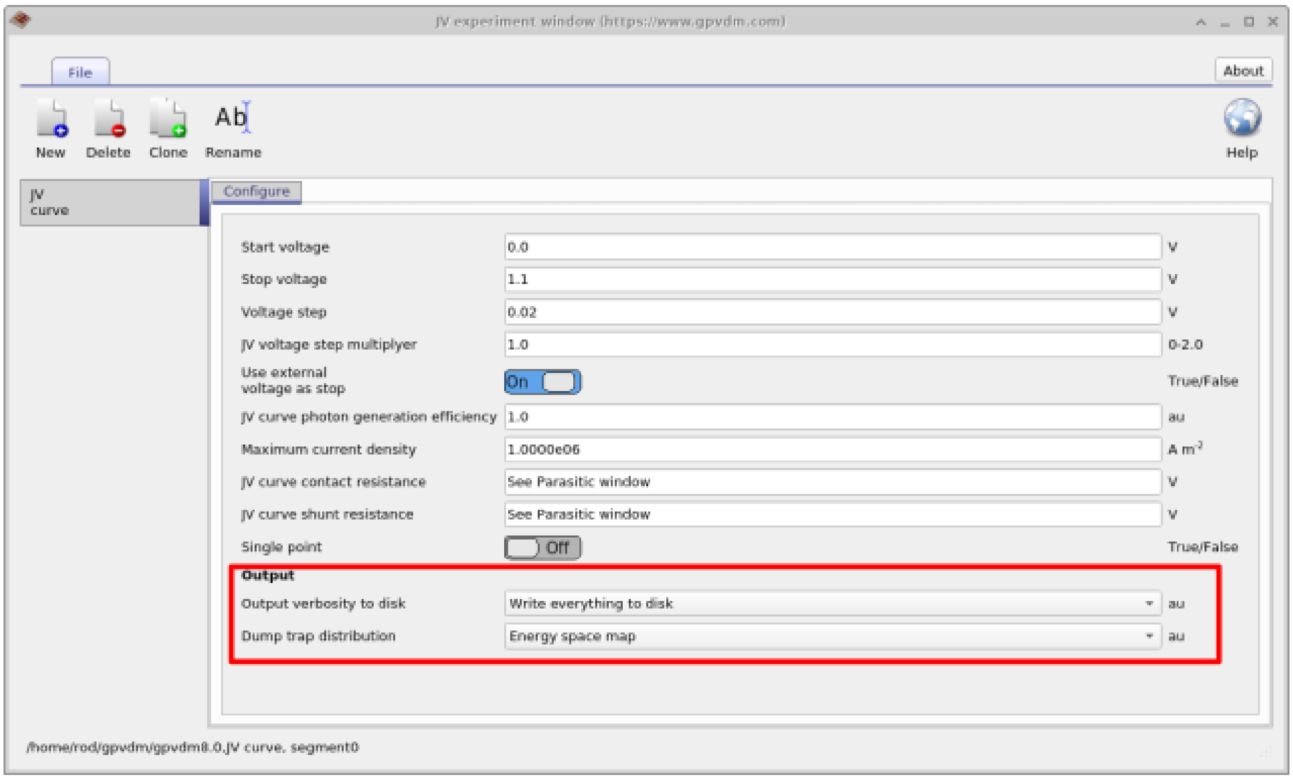
Device type	Number of dimentions
$Solarcells$	1D
$Opticalfilter$	1D
$OFET$	2D

Table 14.1: How many dimensions should I use to simulate my device.

Chapter 15

Output files

In general writing to disk is slow on even the most modern of computers with an SSD. The seek speed of mechanical disks has increased little of their history. Thus often writing the output data to the hard disk is the most time consuming part of any simulation. By default gpvdm writes all output files to disk this is so the new user can get a feel for what output gpvdm can provide. However to speed up simulations you should limit how much data is written to disk. The simulation editor windows (steady state,time domain etc..) offer options to decide how much data you want to dump to disk. This is shown in figure 15.1



Start voltage	0.0	V
Stop voltage	1.1	V
Voltage step	0.02	V
JV voltage step multiplyer	1.0	0-2.0
Use external voltage as stop	On	True/False
JV curve photon generation efficiency	1.0	au
Maximum current density	1.0000e06	A m^{-2}
JV curve contact resistance	See Parasitic window	V
JV curve shunt resistance	See Parasitic window	V
Single point	Off	True/False
Output		
Output verbosity to disk	Write everything to disk	au
Dump trap distribution	Energy space map	au

/home/rod/gpvdm/gpvdm8.0,JV curve, segment0

[H]

Figure 15.1: Selecting which output files are written to disk.

The option "Output verbosity to disk" can be toggled between "None" and "write everything to disk". When "None" is selected nothing is outputted to disk at all - even simulation results are not written. When "write everything to disk" is selected the simulation dumps everything to disk, so JV curves and all internal variables of the solver are written to disk so that the user can examine how carrier densities, fermi-levels, potentials etc.. change during the course of the simulation (see section 15.1). The second option below "Output verbosity to disk" called "dump trap distribution" will write out the distribution of traps in energy and position space.

See section 15.2.

15.1 Snapshots directory - dir

The snapshots directory (see figure 15.2) allows the user to plot all internal solver parameters. For example figure 15.3 where the snapshots tool is being used to plot the conduction band, valance band and quasi Fermi-levels as a function of voltage. The slider can be used to view different voltages.

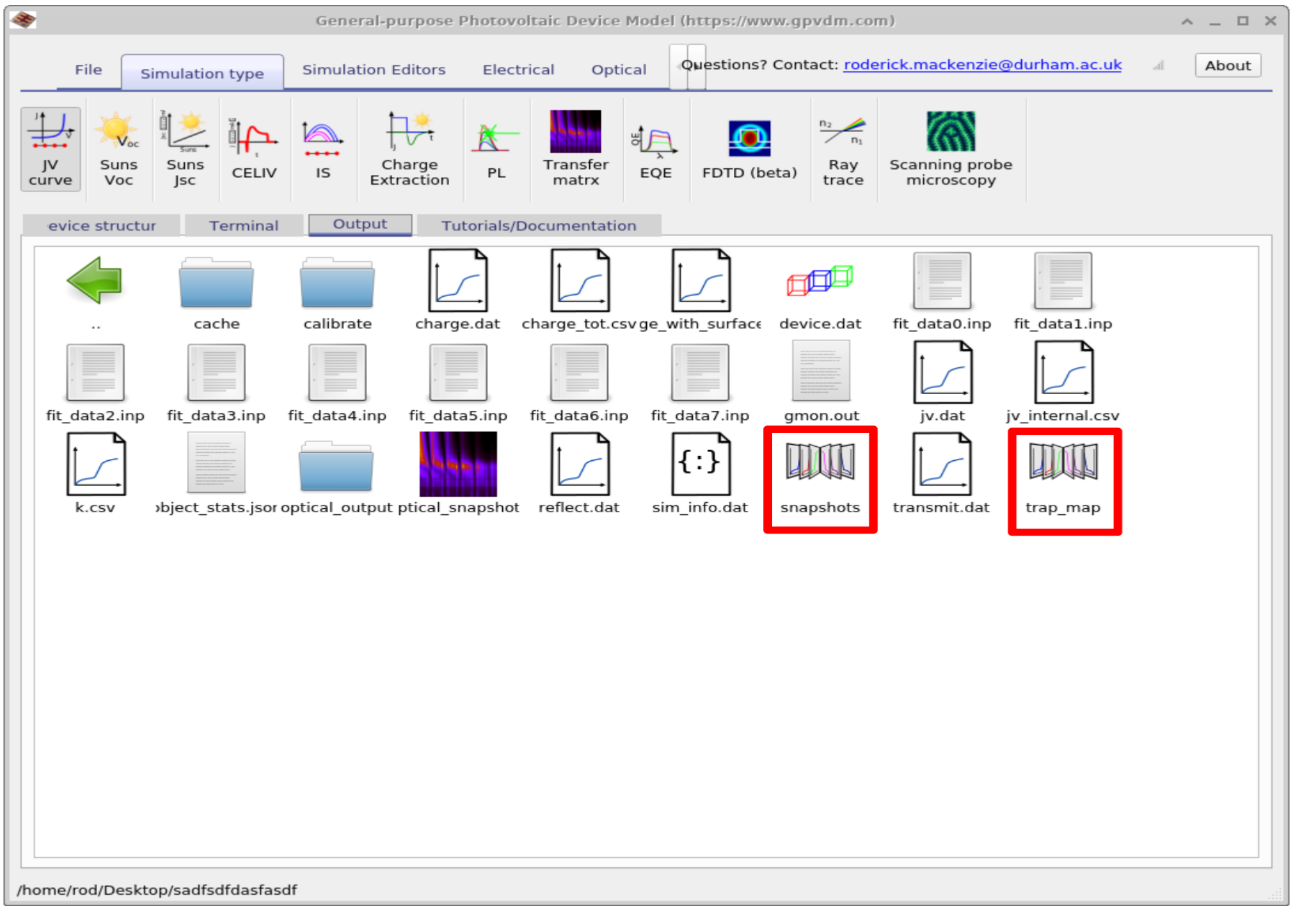
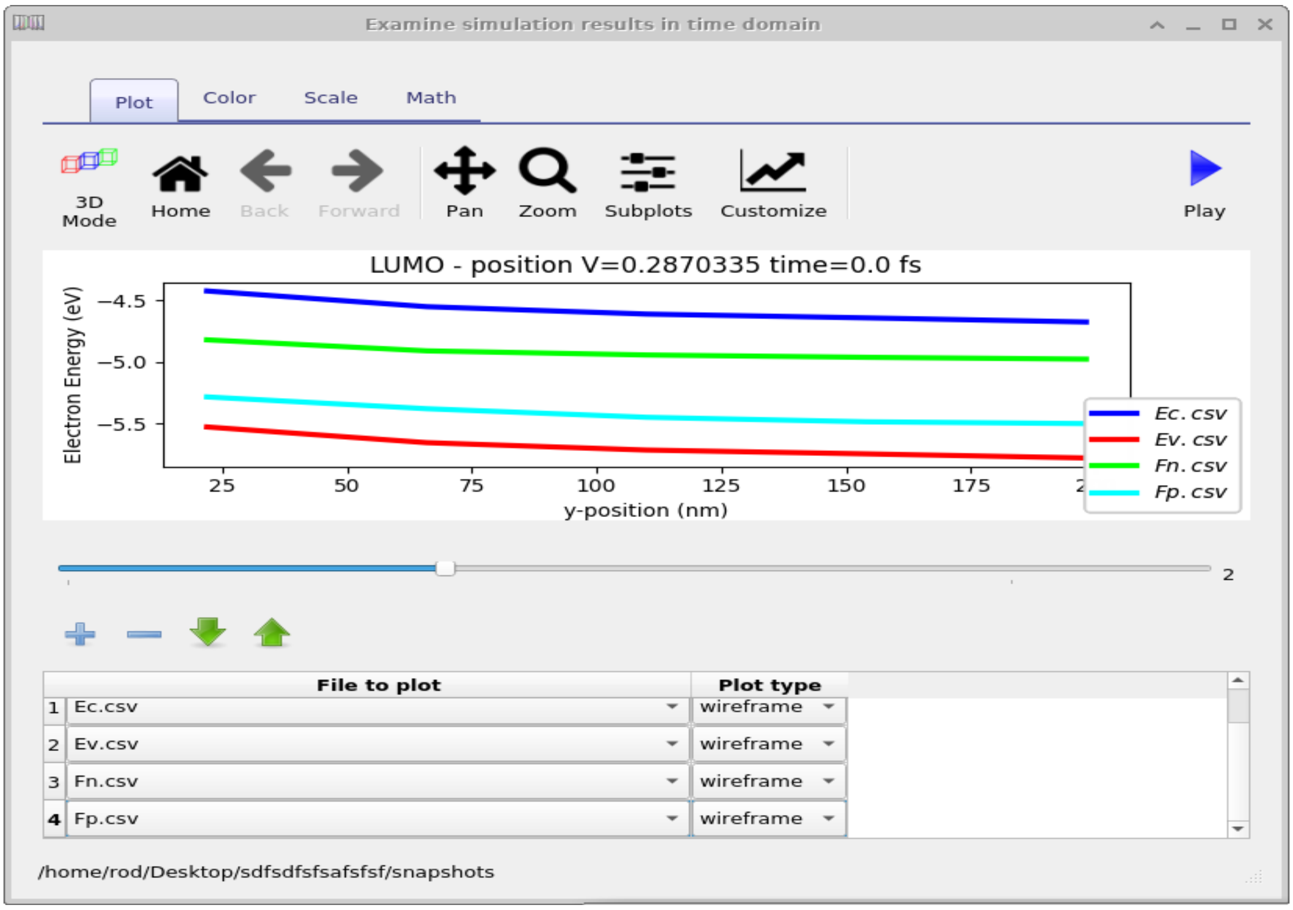


Figure 15.2: The file viewer showing the snapshots and trap map directory



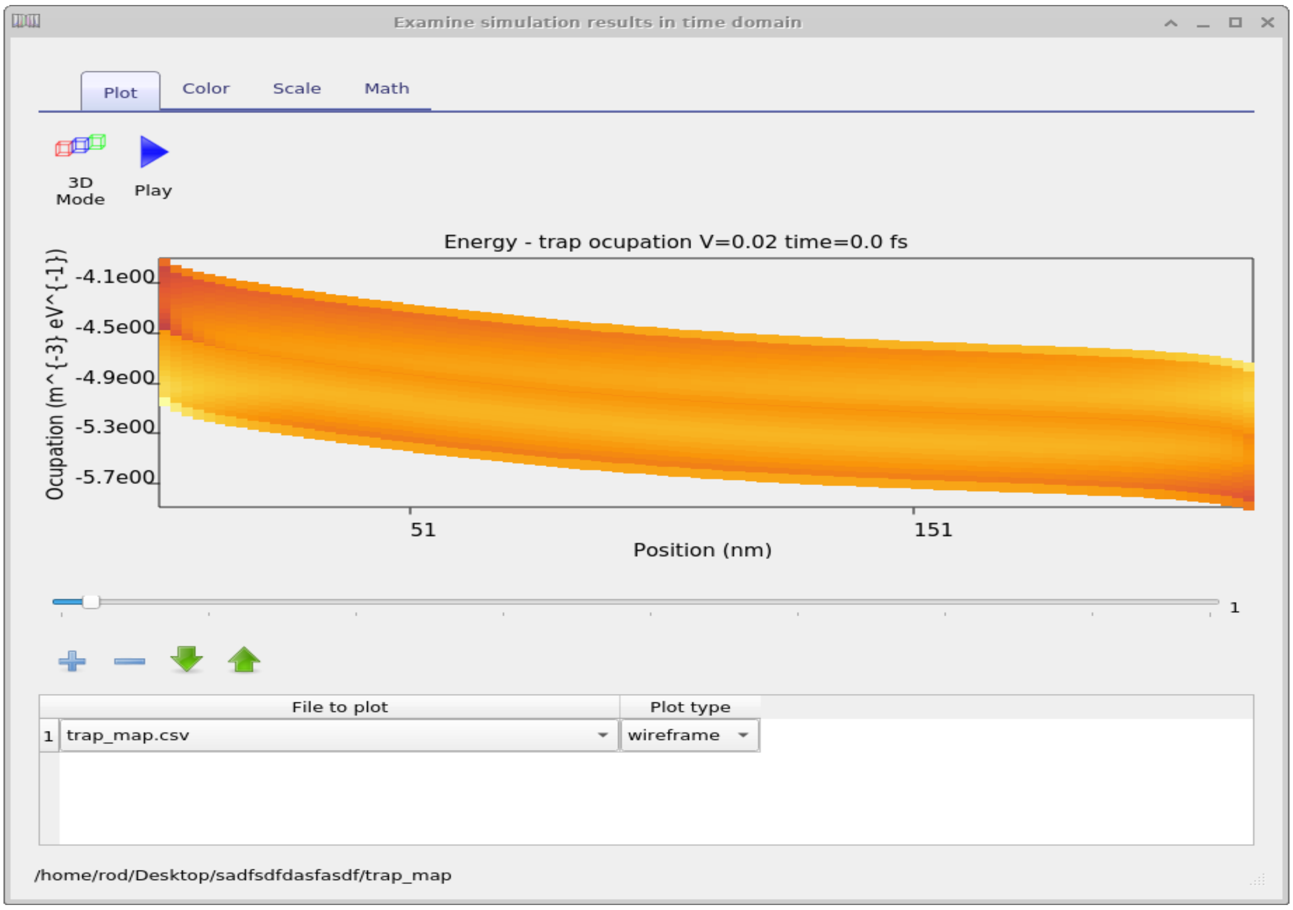
	File to plot	Plot type
1	Ec.csv	wireframe
2	Ev.csv	wireframe
3	Fn.csv	wireframe
4	Fp.csv	wireframe

/home/rod/Desktop/sdfsdfsfsafsfsf/snapshots

Figure 15.3: Using the snapshots tool to view the conduction band, valance band and quasi Fermi-levels

15.2 Trap_map directory - dir

The trap map directory contains the distribution and density of carriers in the traps as a function of position and energetic depth. An example is given in figure 15.4


Examine simulation results in time domain
Plot
Color
Scale
Math
3D Mode
Play
Energy - trap ocupation V=0.02 time=0.0 fs
Ocupation (m^{-3} eV^{-1})
-4.1e00
-4.5e00
-4.9e00
-5.3e00
-5.7e00
51
151
Position (nm)
1

File to plot	Plot type
1 trap_map.csv	wireframe

/home/rod/Desktop/sadfsdfdasfasdf/trap_map

Figure 15.4: Plotting the position and energy dependence of carriers using the trap map tool

15.3 Optical snapshots - dir

Contains results of the optical simulations.

15.4 Cache - dir

Getting a computer to do math is on the whole a slow thing to do. It's much faster to precalculate results then store the answers in a look up table. This can speed up calculations significantly. The cahce dir stores the results of such precalculations, you can delete if you want it gpvdm will just remake it when it runs.

15.5 Equilibrium directory

Before the solver starts any simulation it solves the device equations in the dark with 0V applied bias. The result of this calculation are placed in this directory. The practical reason for doing this is that Newton's method only works if you give it a reasonable starting guess for any given problem. Thus to start the solver, we guess the carrier densities at 0V in the dark, we then use Newton's method to calculate the exact carrier density profiles at 0V in the dark (results are stored in the equilibrium directory), then from this point we can work our way to other solutions say at +1V in the light.[6]

15.6 sim_info.dat

This is a json file containging all key simulation metrics such as J_{sc}, V_{oc}, and example sim_info.dat file is given below:

15.6.1 Steady state electrical simulation

In steady state electrical simulations such as performing a JV scan the sim_info.dat outputs the following parameters.

Symbol	JSON token	Meaning	Units	Equ.	Ref
FF	ff	Fill factor	au		
PCE	pce	PCE	percent		
P_{max}	P_max	Power at Pmax			
V_{oc}	V_oc	V_{oc}			
voc_R	voc_R	Recombination rate at P_{max}			
jv_{voc}	jv_voc				
jv_{pmax}	jv_pmax				
voc_{nt}	voc_nt	Trapped electron carrier densiyt at V_oc			
voc_{pt}	voc_pt	Trapped hole carrier density at V_oc			
voc_{nf}	voc_nf	Free electron carrier densiyt at V_oc			
voc_{pf}	voc_pf	Free hole carrier density at V_oc			
J_{sc}	J_sc	J_{sc}	Am^{-2}		
jv_{jsc}	jv_jsc	Average charge density at J_{sc}	m^{-3}		
jv_{vbi}	jv_vbi	Built in voltage	V		
jv_{gen}	jv_gen	Average generation rate			
voc_{np}	voc_np				
j_{pmax}	j_pmax	Current at P_{max}	Am^{-2}		
v_{pmax}	v_pmax	Voltage at P_{max}	V		

Symbol	JSON token	Meaning	Units	Equ.	Ref
μ_{jsc}	mu_jsc	Avg. mobility at J_{sc}	$m^2V^{-1}s^{-1}$		
μ_{jsc}^{geom}	mu_geom_jsc	Geom. avg. mobility @ J_{sc}	$m^2V^{-1}s^{-1}$		
$\mu_{jsc}^{geom\_micro}$	mu_geom_micro_jsc	Geom. avg. mobility @ J_{sc}	$m^2V^{-1}s^{-1}$		
μ_{voc}	mu_voc	Average mobility @ V_{oc}	$m^2V^{-1}s^{-1}$		
μ_{voc}^{geom}	mu_geom_voc	Geom. avg. mobility @ V_{oc}	$m^2V^{-1}s^{-1}$	$\sqrt{\langle\mu_e\rangle\langle\mu_h\rangle}$	
$\mu_{voc}^{geom\_avg}$	mu_geom_micro_voc	Geom. avg. mobility @ V_{oc}	$m^2V^{-1}s^{-1}$	$\langle\sqrt{mu_e\mu_h}\rangle$	
μ_{pmax}^{e}	mu_e_pmax	Avg. electron mobility @ P_{max}	$m^2V^{-1}s^{-1}$		
μ_{pmax}^{h}	mu_h_pmax	Avg. hole mobility @ P_{max}	$m^2V^{-1}s^{-1}$		
μ_{pmax}^{geom}	mu_geom_pmax	Geom. avg. mobility @ P_{max}	$m^2V^{-1}s^{-1}$	$\sqrt{\langle\mu_e\rangle\langle\mu_h\rangle}$	
$\mu_{pmax}^{geom\_micro}$	mu_geom_micro_pmax	Geom. avg. mobility @ P_{max}	$m^2V^{-1}s^{-1}$	$\langle\sqrt{mu_e\mu_h}\rangle$	
μ^{pmax}	mu_pmax	Avg. mobility @ P_{max}	$m^2V^{-1}s^{-1}$		

Symbol	JSON token	Meaning	Units	Equ.	Ref
τ_{voc}	*tau_voc*	Recom. time at V_{oc}	s	$R = (n - n0)/\tau$	[7]
τ_{pmax}	*tau_pmax*	Recom. time at P_{max}	s	$R = (n - n0)/\tau$	[7]
τ_{voc}^{all}	*tau_all_voc*	Recomb. time at V_{oc}	s	$R = (n)/\tau$	[7]
τ_{pmax}^{all}	*tau_all_pmax*	Recomb. time at P_{max}	s	$R = (n)/\tau$	[7]
$theta_{srh}$	*theta_srh*	θ_{SRH} Collection coefficient at P_{max} y	au		p.100 5.2a[8],[9]
$theta_{srh}$	*theta_srh*	θ_{SRH} Collection coefficient at P_{max}	au		p.100 5.2a[8],[9]

15.6.2 Optical simulation

JSON token	Meaning	Units	Ref
J_{photo}	Photo current density Am^{-2}		
I_{photo}	Photo current A		

15.7 File formats

Almost all input and output files associated with gpvdm are human readable, meaning that they are straight up text files. All output files can be directly plotted in gnuplot/excel as can the input files. Output files are currently called .dat, but they are simply text files. All configuration files are in json format so can be edit directly or by using the python json library.

15.7.1 .dat files

This type of file is a straight text file which can be imported into excel or any other plotting program. It contains two columns of data x and y. There is also a preamble in the file containing information such as units etc.. Gpvdm is moving from .dat files to .csv files.

15.7.2 .csv files

This is a straight csv file as you would expect which can be imported into any text editor. The first line of the file is a json string containing information such as units etc. You can ignore this. The second line of the file describes the x/y data in a human readable form then the rest of the file contains the data.

15.7.3 Binary .csv files - files which are not human readable

In some cases it is not practical to dump text files. Examples are when dealing with 3D structures. In this case gpvdm will dump the same json header as used in the csv file but then dump a series of C floats representing the data.

Chapter 16

Theory of drift diffusion modelling

16.1 Outline

Gpvdm's electrical model is a 1D/2D drift-diffusion model (like many others) however the special thing about gpvdm which makes it very good for disordered materials (Think organics, perovskites and a-Si) is that it goes to the trouble of explicitly solving the Shockley-Read-Hall equations as a function of energy and position space. This enables one to model effects such as mobility/recombination rates changing as a function of carrier population and enables one to correctly model transients as one does not have to assume all the carriers in the trap states have reached equilibrium. Things such as ToF transients, CELIV transients etc.. can be modelled with ease. Of course can be used for more ordered materials as well, you then just need to turn the traps off.

16.2 Summary of model inputs

A device is comprised of a series of layers (upto 10 layers), all these layers will interact with light. Usually only one or two of these layers are electrically active, meaning the transport of electrons and holes must be modeled in detail. Each electrically active layer with in the device has a set of electrical input parameters which define, charge transport, recombination and trapping. (see table) If a device has more than one electrically active layer, then multiple sets of these parameters must be defined. It should be noted that for organic materials (unlike inorganic) there is no standard set of material parameters for any given material. The exact parameters will depend a lot on the fabrication conditions. All layers in the device will also need a refractive index spectrum to be defined, this includes the real and imaginary refractive index as a function of wavelength (typically 300-1000 nm).

16.3 Electrostatic potential

The conduction band/valance band (or LUMO/HOMO in organic semiconductor speak) are defined as

$$E_{LUMO} = -\chi - q\phi \tag{16.1}$$

$$E_{HOMO} = -\chi - E_g - q\phi \tag{16.2}$$

To obtain the internal potential distribution within the device Poisson's equation is solved,

$$\nabla \cdot \epsilon_0 \epsilon_r \nabla = q(n_f + n_t - p_f - p_t - N_{ad} + -N_{ion} + a), \tag{16.3}$$

where n_f, n_t are the carrier densities of free and trapped electrons; p_f and p_t are the carrier densities of the free and trapped holes; and N_{ad} is the doping density. N_{ion} is the background density of perovskite ions and a is the density of mobile ions.

16.4 Free charge carrier statistics

For free carriers the model can either use Maxwell-Boltzmann statistics i.e.

$$n_l = N_c exp\left(\frac{F_n - E_c}{kT}\right) \tag{16.4}$$

$$p_l = N_v exp\left(\frac{E_v - F_p}{kT}\right) \tag{16.5}$$

or full Fermi-dirac statistics i.e.

$$n_{free}(E_f, T) = \int_{E_{min}}^{\infty} \rho(E) f(E, E_f, T) dE \tag{16.6}$$

$$p_{free}(E_f, T) = \int_{E_{min}}^{\infty} \rho(E) f(E, E_f, T) dE \tag{16.7}$$

where

$$f(E) = \frac{1}{1 + e^{E - E_f/kT}} \tag{16.8}$$

When using FD statistics free carriers are assumed to move in a parabolic band:

$$\rho(E)_{3D} = \frac{\sqrt{E}}{4\pi^2}\left(\frac{2m^*}{\hbar^2}\right)^{3/2} \tag{16.9}$$

The average energy of the carriers is defined as

$$\bar{W}(E_f, T) = \frac{\int_{E_{min}}^{\infty} E\rho(E) f(E, E_f, T) dE}{\int_{E_{min}}^{\infty} \rho(E) f(E, E_f, T) dE} \tag{16.10}$$

16.5 Carrier trapping and Shockley-Read-Hall recombination

The model provides two methods to account for carrier trapping and recombination via trap states. The first by equation 16.11, this assumes that the trapped carrier distribution has reached equilibrium. It also assumes there are relatively few trapped charge carriers compared the the number of free carriers, and thus the trapped charges do not significantly change the electrostatic potential. These assumptions are valid when the material is very ordered (i.e. GaAs) or at a push in steady state for some moderately disordered material systems. However if you wish to simulate transient or frequency domain experiments, then you can no longer use 16.11. Instead, one must use a non-equilibrium SRH approach which does not assume trapped carriers have reached equilibrium. Unlike many other models, gpvdm has such a non-equilibrium SRH model built in this is described in section 16.5.2. In fact, it is turned on by default so when using gpvdm you have to go out of your way to turn on equation 16.11.

To understand the importance of such a dynamic solver, consider the following example: You are performing a transient photocurrent experiment (TPC). You photo-excite your device with a laser, carriers very quickly become trapped during the first 1-2μs after photoexcitation, as time passes, the carriers gradually de-trap from deeper and deeper trap states and produce the long photocurrent transient [10]. These transients can often extend out to over 1 second after photo-excitation. Current at the start of the transient originates from shallow traps while current at the end of the transient originates from carriers from very deep trap levels. To simulate this one has to be able to account for the gradual emptying of trap states firstly starting at the shallow traps, then progressing to deeper and deeper trap states. Were one to assume all trap states were in equilibrium one would not be able to simulate this process.

So in summary, although many others have used 16.11 to model disordered devices in time DON'T you results won't make sense. If you want to simulate anything but steady state in an ordered device turn ON the non-equilibrium solver.

16.5.1 Equilibrium Shockley-Read-Hall recombination

For some very ordered material systems where there are not many trap states it is enough to describe SRH trap states using the equation:

$$R^{SRH} = \frac{np - n_0 * p_0}{\tau_p(n + n_1) + \tau_n(p + p_1)} \tag{16.11}$$

where R_{SRH} is the rate of SRH recombination, n, p are the density of free charge carriers n_0, p_0, are the equilibrium density of charge carriers, $\tau_{n,p}$ are the SRH life times and n_1 and p_1 are the trapped electron and hole densities when the Fermi-level matches the trap state energy. This can be turned on in the electrical parameter editor.

16.5.2 Non-equilibrium carrier trapping and recombination using Shockley-Read-Hall trap states

To describe charge becoming trapping into trap states and recombination associated with those states the model uses Shockley-Read-Hall (SRH) theory. A 0D depiction of this SRH recombination and trapping is shown in figure 16.1, the free electron and hole carrier distributions are labeled as n free and p free respectively. The trapped carrier populations are denoted with n trap and p trap , they are depicted with filled red and blue boxes. SRH theory describes the rates at which electrons and holes become captured and escape from the carrier traps. If one considers a single electron trap, the change in population of this trap can be described by four carrier capture and escape rates as depicted in figure 16.1. The rate rec describes the rate at which electrons become captured into the electron trap, r_{ee} is the rate which electrons can escape from the trap back to the free electron population, r_{hc} is the rate at which free holes get trapped and r_{he} is the rate at which holes escape back to the free hole population. Recombination is described by holes becoming captured into electron space slice through our 1D traps. Analogous processes are also defined for the hole traps.

For each trap level the carrier balance 16.12 is solved, giving each trap level an independent quasi-Fermi level. Each point in position space can be allocated between 10 and 160 independent trap states. The rates of each process r_{ec}, r_{ee}, r_{hc}, and r_{he} are give in table 16.1.

$$\frac{\delta n_t}{\partial t} = r_{ec} - r_{ee} - r_{hc} + r_{he} \tag{16.12}$$

The escape probabilities are given by:

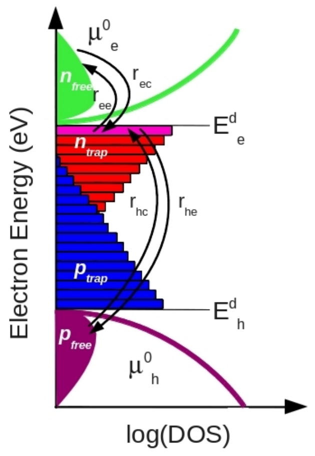


Figure 16.1: Trap filling in both energy and position space as the solar cell is taken from a negative bias Carrier trapping, de-trapping, and recombination

Mechanism	Symbol	Description
Electron capture rate	r_{ec}	$nv_{th}\sigma_n N_t(1-f)$
Electron escape rate	r_{ee}	$e_n N_t f$
Hole capture rate	r_{hc}	$pv_{th}\sigma_p N_t f$
Hole escape rate	r_{he}	$e_p N_t(1-f)$

Table 16.1: Shockley-Read-Hall trap capture and emission rates, where f is the fermi-Dirac occupation function and N_t is the trap density of a single carrier trap.

$$e_n = v_{th}\sigma_n N_c exp\left(\frac{E_t - E_c}{kT}\right) \tag{16.13}$$

and

$$e_p = v_{th}\sigma_p N_v exp\left(\frac{E_v - E_t}{kT}\right) \tag{16.14}$$

where $\sigma_{n,p}$ are the trap cross sections, v_{th} is the thermal emission velocity of the carriers, and $N_{c,v}$ are the effective density of states for free electrons or holes. The distribution of trapped states (DoS) is defined between the mobility edges as

$$\rho^{e/h}(E) = N^{e/h} exp(E/E_u^{e/h}) \tag{16.15}$$

where , $N_{e/h}$ is the density of trap states at the LUMO or HOMO band edge in states/eV and where $E_U^{e/h}$ is slope energy of the density of states.

The value of N_t for any given trap level is calculated by averaging the DoS function over the energy (ΔE) which a trap occupies:

$$N_t(E) = \frac{\int_{E-\Delta E/2}^{E+\Delta E/2} \rho^e E dE}{\Delta E} \tag{16.16}$$

The occupation function is given by the equation,

$$f(E_t, F_t) = \frac{1}{e^{\frac{E_t - F_t}{kT}} + 1} \tag{16.17}$$

Where, E_t is the trap level, and F_t is the Fermi-Level of the trap. The carrier escape rates for electrons and holes are given by

16.5.3 Free-to-free carrier recombination

A free-carrier-to-free-carrier recombination (bi-molecular) pathway is also included. However, most organic solar cells have a great deal of trap states and an ideality factor greater than 1.0 suggesting that free-to-free recombination is not the dominant mechanism. Free-to-free recombination is described using equation 16.18

$$R_{free} = k_r(n_f p_f - n_0 p_0) \tag{16.18}$$

16.5.4 Auger recombination

Auger recombination is as

$$R^{AU} = (C_n^{AU} n + C_p^{AU} p)(np - n_0 p_0) \tag{16.19}$$

where C_n^{AU} and C_p^{AU} are the Auger coefficient of electrons and holes in $m^6 s^{-1}$. This can be set in the electrical paramter editor.

16.6 Charge carrier transport

To describe charge carrier transport, the bi-polar drift-diffusion equations are solved in position space for electrons,

$$\boldsymbol{J_n} = q\mu_e n_f \nabla E_c + qD_n \nabla n_f, \tag{16.20}$$

and holes,

$$\boldsymbol{J_p} = q\mu_h p_f \nabla E_v - qD_p \nabla p_f. \tag{16.21}$$

Conservation of charge carriers is forced by solving the charge carrier continuity equations for both electrons,

$$\nabla \boldsymbol{J_n} = q(R - G + \frac{\partial n}{\partial t}), \tag{16.22}$$

and holes

$$\nabla \boldsymbol{J_p} = -q(R - G + \frac{\partial p}{\partial t}). \tag{16.23}$$

where R and G are the net recombination and generation rates per unit volume respectively.

16.7 Perovskite mobile ion solver

The mobile ion solver is implemented after the work of Calado [11]

$$\boldsymbol{J_a} = q\mu_a a_f \nabla E_v - qD_a \nabla a_f. \tag{16.24}$$

$$\nabla \boldsymbol{J_a} = -q\frac{\partial a}{\partial t}. \tag{16.25}$$

16.8 Semiconductor interfaces

16.8.1 Tunnelling through heterojunctions

Tunnelling of holes through hetrojunction interfaces are is give by

$$\boldsymbol{J_p} = qT_h((p_1 - p_1^{eq}) - (p_0 - p_0^{eq})), \tag{16.26}$$

and for electrons

$$\boldsymbol{J_n} = -qT_e((n_1 - n_1^{eq}) - (n_0 - n_0^{eq})). \tag{16.27}$$

Where T_h and T_e represent the rate constants of the tunnelling. This can be configured in the interfaces editor.

16.8.2 Doping on the interface

Using the interface editor, layers of doping measuring one mesh point thick can be added to either side of the interface. This is useful for OFET simulations where interface charge is important to the turn on voltage.

16.9 Configuring the electrical solver

Behind gpvdm are a series of non-linear solvers that solve the electrical equations in a highly efficient way. These can be configured by going to the electrical tab. There you will see the Drift diffusion button, to the left of that is an arrow. If you click on this it will bring up a window which allows you to configure the "Newton solver". The options are described below.

Related YouTube videos:

How to optimize simulations in gpvdm so they run faster

- Max Electrical iterations (first step): The maximum number of steps the solver can after it's cold started onto a new problem. This is usually at 0V in the dark. The solver usually takes more steps on it's first go.
- Electrical clamp (first step): This is a number by which the maximum newton step is clamped to. 0.1 will make the solver very stable but very slow, 4.0 will make the solver very fast but unstable. A recommended value of 1.0 is suggested for normal problems. If you are solving for high doping or other unusual conditions it can be worth reducing the step. Likewise if you want the solver to be fast and you know the problem is easy set the value to 2.0 or higher. For the first step, I would consider setting this value to be slightly lower than for the subsequent steps.
- Desired solver error (first step): This is the desired error, smaller is more accurate and slower. I would generally not accept answers above $1x10^{-5}$
- Max Electrical iterations: Maximum number of electrical iterations on all but the first step.
- Electrical clamp: Electrical clamp (first step): This is a number by which the maximum newton step is clamped to. 0.1 will make the solver very stable but very slow, 4.0 will make the solver very fast but unstable. A recommended value of 1.0 is suggested for normal problems. If you are solving for high doping or other unusual conditions it can be worth reducing the step. Likewise if you want the solver to be fast and you know the problem is easy set the value to 2.0 or higher.
- Desired solver error: This is the desired error, smaller is more accurate and slower. I would generally not accept answers above $1x10^{-5}$
- Newton solver clever exit: If the solver starts bouncing in the noise then assume we can't get a better answer and quit.
- Newton minimum iterations: Don't allow the solver to quit before doing this number of steps. Often the error in the first few steps of the solution can be below "Desired solver error", thus the solver can quit before finding the true answer.
- Solve Kirchhoff's current law in Newton solver: Solve Kirchhoff's current law in the main Newton Jacobian.
- Matrix solver: This selects the matrix solver to use.
- Newton solver to use:
  - none: No electrical solver is selected, this is used when only solving optical or thermal problems.

  - newton: The standard 1D Newton solver.
  - newton_2D: The standard 2D Newton solver.
  - newton_norm: The standard 1D Newton solver but with Slotboom normalization. This is handy when solving systems with large difference in density between minority and majority carrier density.
  - poisson_2d: A 2D Poisson solver with no drift diffusion equations.

- Complex matrix solver:
- Slotboom T0: Slotboom variable for the newton_norm solver.
- Slotboom D0: Slotboom variable for the newton_norm solver.
- Slotboom n0: Slotboom variable for the newton_norm solver.
- Use newton cache (experimental): Cache large problems to disk - experimental.
- Quit on convergence problem: Quit on convergence problem. Quite often
- Quit on inverted Fermi-level:
- Solver output verbosity:

16.9.1 Solver stability

Avoiding very big and very small numbers

Try opening up MATLAB (Octave if you are on Linux) and typing in the following equation $((1e-1+1e1)-1e1)/1e-1$. Before pressing enter, try to evaluate it in your head. the $1e1$ and the $-1e1$ cancel leaving $\frac{1e-1}{1e-1}$ which equates to 1. Now try replacing the powers to 1 with to the 19, so type in $((1e-19+1e19)-1e19)/1e-19$, again evaluate this in your head. Again , $1e19$ and the $-1e19$ cancel leaving $\frac{1e-19}{1e-19}$ which equates to 1 Now let the computer evaluate the expression. In fact this time the computer does not give you 1 but gives you 0. Double check that you typed it in correctly... you did so what is happening. Why is the computer giving me an answer which is 100% wrong. The answer is easy, computers have a limited precision. This means that they can only store a limited number of decimal places. On a modern PC it's about 15 decimal places. After this the computer starts ignoring the numbers. So when we added $(1e-19+1e19)$ the computer could not keep track of the decimal places so it assumed that the answer was exactly $1.000000000000000e19$ and not $1.0000000000000000001e19$, then when we subtracted $-1e19$ from the answer the computer gave us zero instead of $1e-19$. The $1e-19$ was lost in the precision.

All computers are affected by this no matter how powerful they are, this has important implications when solving device equations. If you have too big a spread of numbers in your simulation (matrix/Jacobian) the computer won't be able to solve it easily. So if you have very low values of mobility say $1e-19$ and very big values say $1e5$ the computer wills start to have problems solving the electrical problem. There fore generally try to reduce the spread of parameters in you model. This is important when simulating insulators.

Avoid zeros

Zeros are bad because they cause divide by zero errors. So don't have zero mobilities, carrier cross sections, tail slopes or densities of states. It's fine to have zero recombination constants though.

Very big steps in the band gap

Big steps in the band gap will produce very small and very large carrier densities - see *Avoiding very big and very small numbers* above.

16.9.2 Simulating disordered devices without traps

This section needs to be rewritten, to more generally talk about recombination and not just Langevin recombination. For a more complete view watch the video below

Related YouTube videos:

Please stop simulating disordered semiconductors without trap states.

In my view Langevin recombination is in general a really bad way to describe recombination in OPV devices. This is because the mechanism assumes Brownian motion of electrons and holes and that charge carriers of opposite polarity will recombine when they get close enough to fall into each others electrostatic field. This picture assumes the charge carriers are free and completely neglects the influence of trap states. I therefore think Langeving recombination should be avoided in OPVs. But in dx.doi.org/10.1021/jp200234m you used Langevin recombination - why?: In this paper I allowed the mobility in the Langevin expression to vary as a function of carrier density i.e.

$$R_{free} = qk_r \frac{(\alpha\mu_e(n) + \beta\mu_h(n))n_{tot}p_{tot}}{2\epsilon_0\epsilon_r} \tag{16.28}$$

I then by defining a mobility edge and assuming any carrier below the mobility edge could not move and any carrier above it could. I could define the averaged electron/hole mobility as:

$$\mu_e(n) = \frac{\mu_e^0 n_{free}}{n_{free} + n_{trap}} \tag{16.29}$$

and

$$\mu_h(n) = \frac{\mu_h^0 p_{free}}{p_{free} + p_{trap}} \tag{16.30}$$

and if one assumes the density of free charge carriers is much smaller than the density of trapped charge carriers one can arrive at

$$R(n,p) = qk_r \frac{(\alpha\mu_e^0 n_{free}p_{trap} + \beta\mu_h p_{free}n_{trap})}{2\epsilon_0\epsilon_r} \tag{16.31}$$

Thus by making the mobility carrier density dependent we arrive at an expression for Langeving recombination that's dependent upon the density of free and trapped carriers (i.e. $n_{free}p_{trap}$ and $p_{free}n_{trap}$) This is in principle the same as SRH recombination (i.e. a process involving free electrons (holes) recombining with trapped holes (electrons)). This was a nice simple approach and it worked quite well in the steady state. However, to make this all work I had to assume all electrons (holes) at any given position in space had a single quasi-Fermi level, which meant they were all in equilibrium with each other. For this to be true, all electrons (holes) would have to be able to exchange energy with all other electrons (holes) at that position in space and have an infinite charge carrier thermalization velocity. This seemed like an OK assumption in steady state when electrons (holes) had time to exchange energy, however once we start thinking about things happening in time domain, it becomes harder to justify because there are so many trap states in the device it is unlikely that charge carriers will be able to act as one equilibrated gas with one quasi-Fermi level. On the other hand the SRH mechanism does not make this assumption, so it is probably a better description of recombination/trapping. I would also add that I have never found a situation in OPV device modeling where SRH recombination was unable to describe the device in question. Conclusion: SRH is better than Langevin.

16.10 Calculating the built in potential

The first step to performing a device simulation, is to calculate the built in potential of the device. To do this we must know the following things:

- The majority carrier concentrations on the contacts n and p.
- The effective densities of states N_{LUMO} and N_{HOMO}.
- The effective band gap E_g

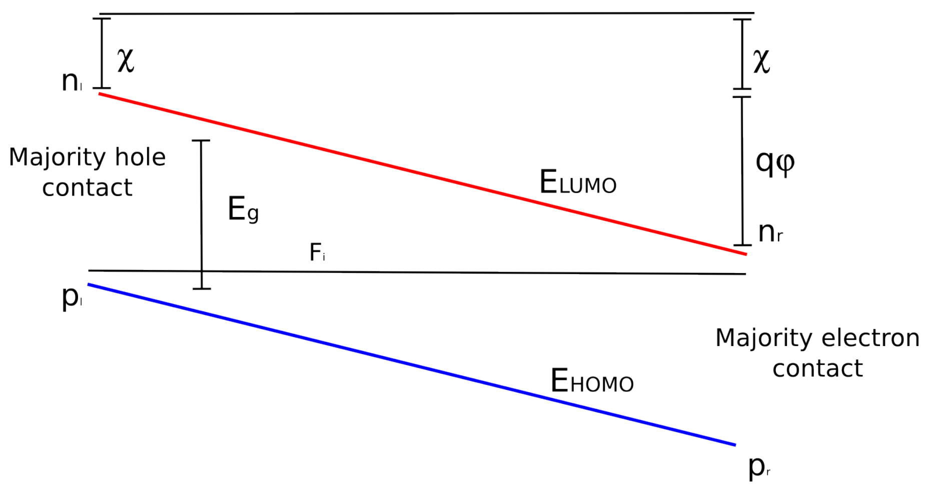


Figure 16.2: Band structure of device in equilibrium.

The left hand side of the device is given a reference potential of 0 V. See figure 16.2. We can then write the energy of the LUMO and HOMO on the left hand side of the device as:

$$E_{LUMO} = -\chi \tag{16.32}$$

$$E_{HOMO} = -\chi - E_g \tag{16.33}$$

For the left hand side of the device, we can use Maxwell-Boltzmann statistics to calculate the equilibrium Fermi-level (F_i).

$$p_l = N_v exp\left(\frac{E_{HOMO} - F_p}{kT}\right) \tag{16.34}$$

We can then calculate the minority carrier concentration on the left hand side using F_i

$$n_l = N_c exp\left(\frac{F_n - E_{LUMO}}{kT}\right) \tag{16.35}$$

The Fermi-level must be flat across the entire device because it is in equilibrium. However we know there is a built in potential, we can therefore write the potential of the conduction and valance band on the right hand side of the device in terms of *phi* to take account of the built in potential.

$$E_{LUMO} = -\chi - q\phi \tag{16.36}$$

$$E_{HOMO} = -\chi - E_g - q\phi \tag{16.37}$$

we can now calculate the potential using

$$n_r = N_c exp\left(\frac{F_n - E_{LUMO}}{kT}\right) \tag{16.38}$$

equation 16.36.

The minority concentration on the right hand side can now also be calculated using.

$$p_r = N_v exp\left(\frac{E_v - F_{HOMO}}{kT}\right) \tag{16.39}$$

The result of this calculation is that we now know the built in potential and minority carrier concentrations on both sides of the device. Note, infinite recombination velocity on the contacts is assumed. I have not included finite recombination velocities in the model simply because they would add four more fitting parameters and in my experience I have never needed to use them to fit any experimental data I have come across.

Once this calculation has been performed, we can estimate the potential profile between the left and right hand side of the device, using a linear approximation. From this the charge carrier densities across the device can be guessed. The guess for potential and carrier densities, is then used to prime the main Newton solver. Where the real value are calculated. The Newton solver is described in the next section.

16.10.1 Average free carrier mobility

In this model there are two types of electrons (holes), free electrons (holes) and trapped electrons (holes). Free electrons (holes) have a finite mobility of μ_e^0 (μ_h^0) and trapped electrons (holes) can not move at all and have a mobility of zero. To calculate the average mobility we take the ratio of free to trapped carriers and multiply it by the free carrier mobility.:

$$\mu_e(n) = \frac{\mu_e^0 n_{free}}{n_{free} + n_{trap}} \tag{16.40}$$

Thus if all carriers were free, the average mobility would be μ_e^0 and if all carriers were trapped the average mobility would be 0. It should be noted that only μ_e^0 (μ_h^0) are used in the model for computation and $\mu_e(n)$ is an output parameter.

The value of μ_e^0 (μ_h^0) is an input parameter to the model. This can be edited in the electrical parameter editor. The value of $\mu_e(n)$, and $\mu_h(p)$ are output parameters from the model. The value of $\mu_e(n)$, and $\mu_h(p)$ change as a function of position, within the device, as the number of both free and trapped charge carriers change as a function of position. The values of $\mu_e(x)$, and $\mu_h(x)$ can be found in *mu_n_ft.dat* and *mu_p_ft.dat* within the *snapshots* directory. The spatially averaged value of mobility, as a function of time or voltage can be found in the files *dynamic_mue.dat* or *dynamic_muh.dat* within the dynamic directory.

Were one to try to measure mobility using a technique such as CELIV or ToF, one would expect to get a value closer to $\mu_e(n)$ or $\mu_h(p)$ rather than closer to μ_e^0 or μ_h^0. It should be noted however, that measuring mobility in disordered materials is a difficult thing to do, and one will get a different experimental value of mobility depending upon which experimental measurement method one uses, furthermore, mobility will change depending upon the charge density profile within the device, and thus upon the applied voltage and light intensity. To better understand this, try for example doing a CELIV simulation, and plotting $\mu_e(n)$ as a function of time (Voltage). You will see that mobility reduces as the negative voltage ramp is applied, this is because carriers are being sucked out of the device. Then try extracting the mobility from the transient using the CELIV equation for extracting mobility. Firstly, the CELIV equation will give you one value of mobility, which is a simplification of reality as the value really changes during the application of the voltage ramp. Secondly, the value you get from the equation will almost certainly not match either μ_e^0 or any value of $\mu_e(n)$. This simply highlights, the difficult of measuring *a* value of mobility for a disordered semiconductor and that really when we quote a value of mobility for a disordered material, it really only makes sense to quote a value measured under the conditions a material will be used. For example, for a solar cell, values of $\mu_e(n)$ and $\mu_h(n)$, would be most useful to know under 1 Sun at the P_{max} point on a JV curve.

16.11

There are three options for thermal simulation in gpvdm; 1) A constant temperature through the device. This is recommended for most simulation and is set at 300K by default; 2) a lattice thermal solver 16.11.1, this solves the heat equation throughout the device taking into account self heating. This is useful for simulating devices which get hot through their operation; 3) A hydrodynamic thermal 16.11.2 solver which does not assume the electron, hole and lattice temperatures are equal. This is useful for simulating heat flow over heterojunctions or where carriers do not have time to relax to the lattice temperature.

The drift diffusion equations given in 16.20 and 16.24 are only valid in isothermal conditions. The full transport equations as derived from the BTE [12] are given by

$$\mathbf{J}_n = \mu_e n \nabla E_c + \frac{2}{3}\mu_e n \nabla \bar{W} + \frac{2}{3}\bar{W}\mu_e \nabla n - \mu_e n \bar{W}\frac{\nabla m_e^*}{m_e^*} \tag{16.41}$$

$$\mathbf{J}_p = \mu_h p \nabla E_v - \frac{2}{3}\mu_h p \nabla \bar{W} - \frac{2}{3}\bar{W}\mu_h \nabla p + \mu_p p \bar{W}\frac{\nabla m_h^*}{m_h^*} \tag{16.42}$$

where $\bar{W}$ is the average kinetic energy of the free carriers as given by 16.10. If the average energy is assumed to be 3/2kT, 16.41 and 16.42, return to the standard drift diffusion equations. Note the full form of these equations is required when not using MB statistics.

The thermal model can be configured in the thermal ribbon 16.3. Usually the thermal model is turned off and a constant temperature (300K) is assumed across the device. If you wish to adjust this temperature click on the "Set temperature icon". The thermal model can be turned on by clicking on the candle to the on the far left of the thermal ribbon, so that a flame appears. Various heating sources can be enabled or disabled by depressing the buttons to the right of the ribbon. Boundary conditions can be set in the "Boundary Conditions" window, thermal constants of the material layers can be changed in the "Thermal parameters window".

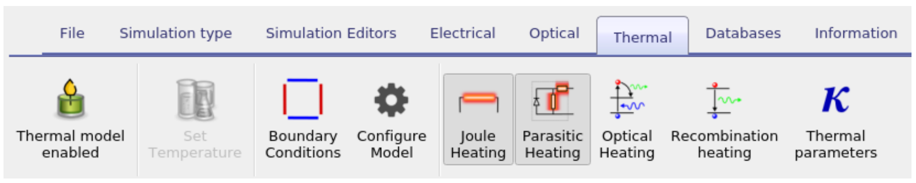


Figure 16.3: Thermal

16.11.1 Lattice thermal model

When solving only the lattice heat equation heat transfer and generation is given by

$$0 = \nabla \kappa_l \nabla T_L + H_j + H_r + H_{optical} + H_{shunt} \tag{16.43}$$

where joule heating (H_j) is give by

$$H_j = J_n \frac{\nabla E_c}{q} + J_h \frac{\nabla E_h}{q}, \tag{16.44}$$

recombination heating (H_r) is given by,

$$H_r = R(E_c - E_v) \tag{16.45}$$

optical absorption heating is given by,

$$H_{optical} \tag{16.46}$$

and heating due to the shunt resistance is given by

$$H_{shunt} = \frac{J_{shunt} V_{applied}}{d}. \tag{16.47}$$

The thickness of the device is given by d. Note shunt heating is only in there to conserve energy conservation.

16.11.2 Energy balance - hydrodynamic transport model

If you turn on the electrical and hole thermal model, then the heat source term will be replaced by

$$H = \frac{3k_b}{2}\left(n(\frac{T_n - T_l}{\tau_e}) + p(\frac{T_p - T_l}{\tau_h}) \right) + R(E_c - E_v) \tag{16.48}$$

and the energy transport equation for electrons

$$S_n = -\kappa_n \frac{dT_n}{dx} - \frac{5}{2}\frac{k_b T_n}{q} J_n \tag{16.49}$$

and holes,

$$S_p = -\kappa_p \frac{dT_p}{dx} + \frac{5}{2}\frac{k_b T_p}{q} J_p \tag{16.50}$$

will be solved.
The energy balance equations will also be solved for electrons,

$$\frac{dS_n}{dx} = \frac{1}{q}\frac{dE_c}{dx} J_n - \frac{3k_b}{2}\left(RT_n + n(\frac{T_n - T_l}{\tau_e}) \right) \tag{16.51}$$

and for holes

$$\frac{dS_p}{dx} = \frac{1}{q}\frac{dE_v}{dx} J_p - \frac{3k_b}{2}\left(RT_p + n(\frac{T_p - T_l}{\tau_e}) \right) \tag{16.52}$$

The thermal conductivity of the electron gas is given by

$$\kappa_n = \left(\frac{5}{2} + c_n \right) \frac{{k_b}^2}{q} T_n \mu_n n \tag{16.53}$$

and for holes as,

$$\kappa_p = \left(\frac{5}{2} + c_p \right) \frac{{k_b}^2}{q} T_p \mu_p p \tag{16.54}$$

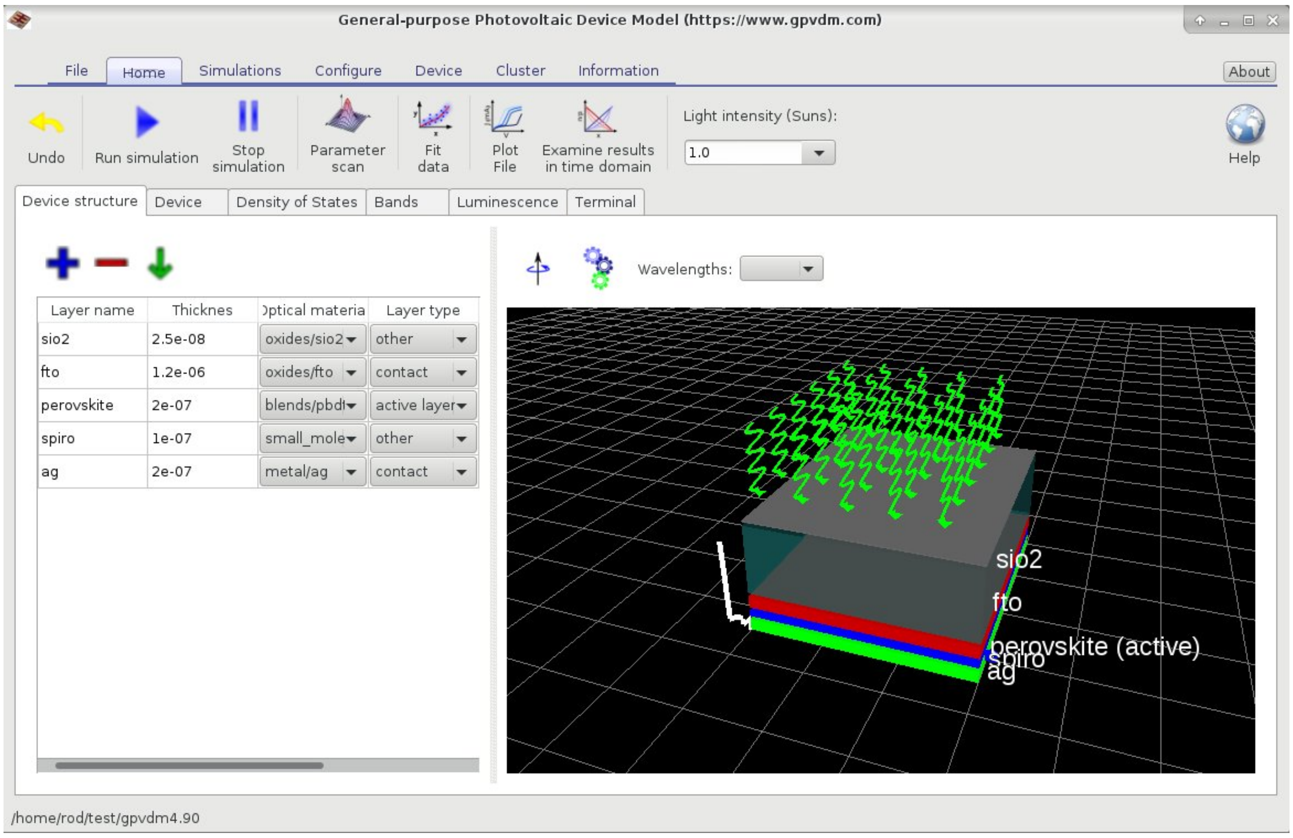
Chapter 17

Troubleshooting

17.1 Windows gives warms me the software is unsigned

17.2 Why don't I get a 3D view of the device

If your simulation window looks like figure 17.2 and not like figure 17.1. It means either you do not have any 3D acceleration hardware on your computer, or you do not have the drivers for it installed. If you have an ATI/Nvidia/Intel graphics card check that the drivers are installed. Currently, not having working 3D hardware will not affect your ability to perform simulations. This is not a gpvdm bug it's a driver/hardware issue on your computer.

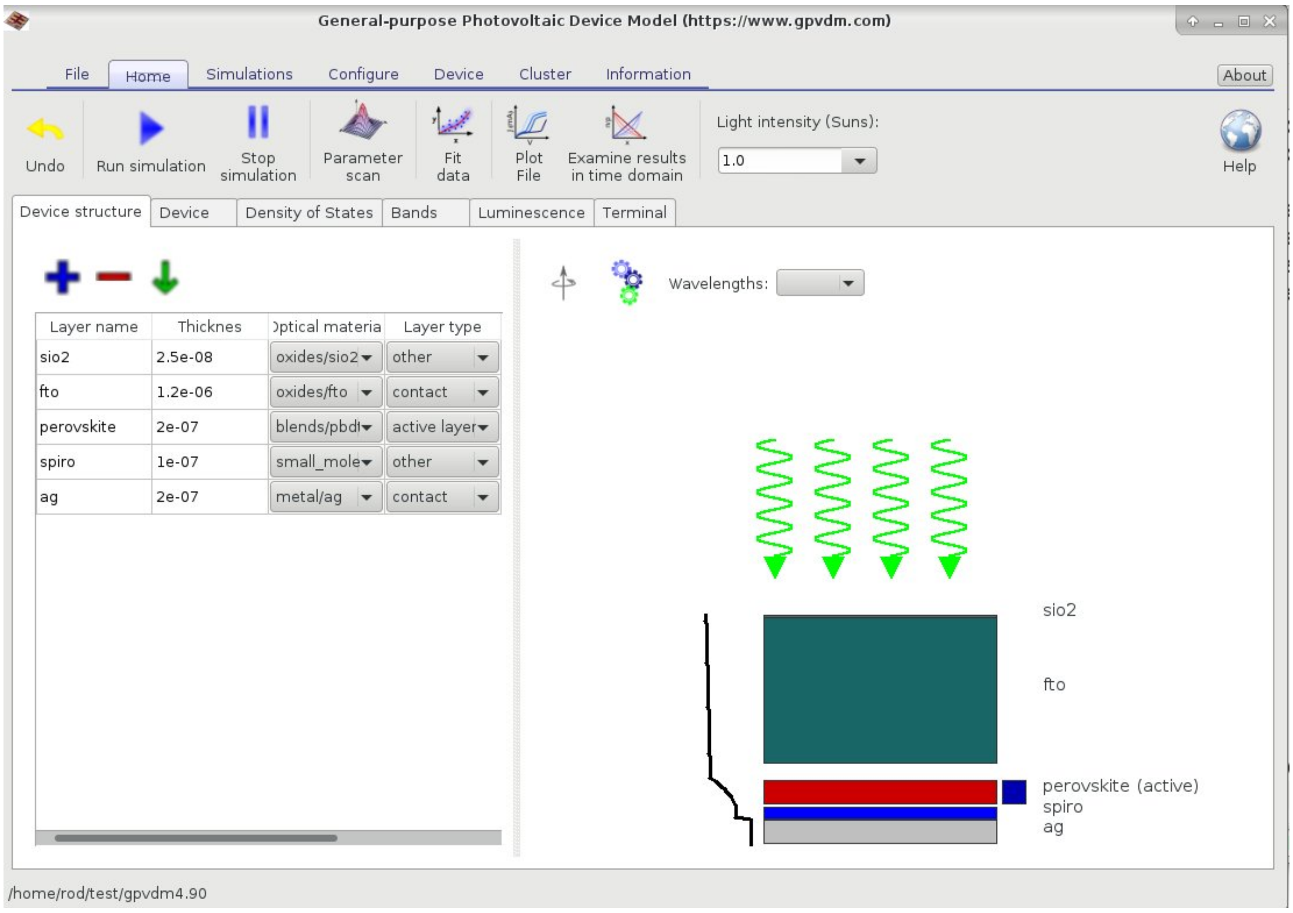


Light intensity (Suns): 1.0

Layer name	Thicknes	Optical materia	Layer type
sio2	2.5e-08	oxides/sio2	other
fto	1.2e-06	oxides/fto	contact
perovskite	2e-07	blends/pbdt	active layer
spiro	1e-07	small_mole	other
ag	2e-07	metal/ag	contact

/home/rod/test/gpvdm4.90

Figure 17.1: gpvdm with working 3D acceleration hardware.



Light intensity (Suns): 1.0

Layer name	Thicknes	Optical materia	Layer type
sio2	2.5e-08	oxides/sio2	other
fto	1.2e-06	oxides/fto	contact
perovskite	2e-07	blends/pbd	active layer
spiro	1e-07	small_mole	other
ag	2e-07	metal/ag	contact

Figure 17.2: gpvdm with no 3D acceleration hardware.

Chapter 18

Text past this point needs to be rewritten

Should I trust the results of gpvdm?

Yes! The model it's self has been verified against experiment [there are over 20 publications doing this, in steady state, time domain (us-fs time scales), and fx-domain]. The basic drift-diffusion solver was cross checked and compared against other drift diffusion models, and the accuracy compared down to 6-9 dp. While the optical model has been compared to analytical solutions of Maxwell's equations. The SRH model has also been compared against analytical models. If the answers you are getting out of gpvdm are odd, then I would suggest to take a look at the input parameters. If your efficienceis are high, try increasing the number of trap states, the recombination cross sections or reducing the e/h mobilites. Finally, I would also recommend always running the latest version, and keeping an eye on the twitter stream for bug announcements.

18.0.1 Can I use the model to simulate my exotic* material system/contacts?

The short answer is yes. The model is an effective medium model, meaning that it does not simulate the details of the medium, rather it approximates the medium with a set of electrical parameters. For example, when simulating an organic solar cell, it does not simulate every detail of the BHJ, rather it just assumes an effective mobility, density of states, recombination cross sections, trapping cross sections and so on... So if you can find electrical parameters to aproximate your material system (or guess them), there is nothing stopping you using gpvdm to simulate any exotic device/material. The same goes for the contacts, the model simulates the contacts simply as a charge density. So if you have fancy graphene contacts which inject lots of charge, use a high majority carrier density on the contacts. Where as if you have some dirty old ITO contacts may be drop the majority carrier density a bit.

18.1 Excited states

This is a new feautre and is not yet finished. I am writing the equations as I implment them in the solver.

$$\frac{dN_S}{dt} = \frac{1}{4}R_{free} + \frac{1}{4}\kappa_{TT}N_T^2 - (\kappa_S + \kappa_{ISC})N_s - (\frac{7}{4}\kappa_{SS}N_S - \kappa_{ST}N_T)N_S \tag{18.1}$$

$$\frac{dN_t}{dt} = \frac{3}{4}R_{free} + \kappa_{ISC}N_S + \frac{3}{4}\kappa_{SS}N_S^2 \tag{18.2}$$

$$\frac{dN_{SD}}{dt} = \frac{\kappa_{FRET}}{N_{DOP}}(N_{DOP} - N_{SD} - N_{TD})N_s + \frac{1}{4}\kappa_{TTD}N_{TD}^2 - (\frac{7}{4}\kappa_{SSD}N_{SD} + \kappa_{STD}N_{TD})N_{SD} \tag{18.3}$$

Chapter 19

Legal

19.1 License

Gpvdm comprises of three independent components, gpvdm gui, gpvdm core and gpvdm data. In general everything is under the MIT license except the Python GUI which I have released under GPL v2.0. Details can be found here.

19.2 Copyright of the manual

This manual is released under CC-BY license.

19.3 Data privacy statement

In some versions of gpvdm it will ask you to register before using it. In these versions it asks for your name, title, company that you work for and what you intend on using gpvdm for. This data is then transmitted to the gpvdm server where it is securely stored. The reason I ask for this information is to be able generate accurate usage information. Having accurate information helps when requesting grants from funding bodies. It's much easier to ask a funding body for money if you can prove you actually have users and your software is a benefit to society. Periodicity gpvdm will also contact the gpvdm server to see if there are any software updates. By using gpvdm you agree for the above to happen.

Bibliography

[1] Z. Liu, Z. Deng, S. J. Davis, C. Giron, and P. Ciais. *Nature Reviews Earth & Environment*, Mar 2022.

[2] S. Manabe and R. T. Wetherald. *Journal of Atmospheric Sciences*, 1967, 24 3 241 – 259.

[3] S. Solak, S. Shishegaran, A. C. Hübler, and R. C. MacKenzie. *Solar RRL*, 2021, 5 12 2100787.

[4] F. NÉRY and J. Matos. *Proceedings of the 14th European Colloquium on Theoretical and Quantitative Geography*, page 23.

[5] J. Schneider. *Understanding the Finite-Difference Time-Domain Method.* 2010.

[6] T. Zhan, X. Shi, Y. Dai, X. Liu, and J. Zi. *Journal of Physics: Condensed Matter*, 2013, 25 21 215301.

[7] B. C. O'Regan, J. R. Durrant, P. M. Sommeling, and N. J. Bakker. *The Journal of Physical Chemistry C*, 2007, 111 37 14001–14010.

[8] P. Kaienburg. 2019.

[9] P. Kaienburg, U. Rau, and T. Kirchartz. *Phys. Rev. Applied*, Aug 2016, 6 024001.

[10] R. C. MacKenzie, C. G. Shuttle, G. F. Dibb, N. Treat, E. von Hauff, M. J. Robb, C. J. Hawker, M. L. Chabinyc, and J. Nelson. *The Journal of Physical Chemistry C*, 2013, 117 24 12407–12414.

[11] P. Calado, A. M. Telford, D. Bryant, X. Li, J. Nelson, B. C. O'Regan, and P. R. Barnes. *Nature communications*, 2016, 7 1 1–10.

[12] E. M. Azoff. *Solid State Electronics*, September 1987, 30 9 913–917.

Use the power of device simulation to understand your experimental data from thin film devices such as Organic Solar cells, sensors, OFET, OLEDs, Perovskite solar cells, and many more. Unlike may other models gpvdm is purpose built from the ground up for simulating thin film devices made from disordered materials. Downloaded more than 25,000 times with over 200 papers published using the model, gpvdm has become one of they key device modelling tools used by the scientific community developing the next generation of opto-electronic devices.

This book written by the author of gpvdm explains will take you from a standing start to being able to confidently simulate your own devices. Learn how to simulate, Organic solar cells, Organic Field Effect Transistors, Perovskite solar cells, Organic LEDs, Large area printed devices and many more..