SCAPS manual

Marc Burgelman
Koen Decock, Alex Niemegeers, Johan Verschraegen, Stefaan Degrave

University of Gent
Department of Electronics and Information Systems (ELIS)
Campus Ardoyen, Technologiepark 914, Grote Steenweg Noord
9052 Gent-Zwijnaarde
'Belgium'

SOLAR CELLS

elis
dept. electronics & information systems

UNIVERSITEIT
GENT

Table of contents

Chapter 1: About SCAPS

SCAPS is a one dimensional solar cell simulation program developed at the department of Electronics and Information Systems (ELIS) of the University of Gent, Belgium. Several researchers have contributed to it's development: Alex Niemegeers, Marc Burgelman, Koen Decock, Johan Verschraegen, Stefaan Degrave. A description of the program, and the algorithms it uses, is found in the literature [1-6].

The program is freely available to the PV research community (universities and research institutes). It runs on PC under Windows 95, 98, NT, 2000, XP, Vista, Windows 7, and occupies about 50 MB of disk space.

The program can be freely downloaded (but...: don't sell, don't distribute further, refer when you publish results obtained with SCAPS). Please report to Marc Burgelman (<u>Marc.Burgelman@elis.ugent.be</u>) when you have downloaded a SCAPS version (your name and your institution name and address, and the promotor's name for doctorate's students).

Up to now, there was no consistent manual for the program but there was (and is) a collection of add-on manuals describing the novelties in every new version. This manual is based on those previous documents. Also, there are two short and recommendable documents: a text *Getting Started 2011.pdf*, and a presentation *SCAPS introduction.pdf*. These documents are doing exactly what they promise.

SCAPS is originally developed for cell structures of the CuInSe2 and the CdTe family. Several extensions however have improved its capabilities so that it is also applicable to crystalline solar cells (Si and GaAs family) and amorphous cells (a-Si and micromorphous Si). An overview of its main features is given below:

- up to 7 semiconductor layers
- almost all parameters can be graded (i.e. dependent on the ocal composition or on the depth in the cell): E_g, χ, ε, N_C, N_V, v_{thn}, v_{thp}, μ_n, μ_p, N_A, N_D, all traps (defects) N_t
- recombination mechanisms: band-to-band (direct), Auger, SRH-type
- defect levels: in bulk or at interface; their charge state and recombination is accounted for
- defect levels, charge type: no charge (idealisation), monovalent (single donor, acceptor), divalent (double donor, double acceptor, amphoteric), multivalent (user defined)
- defect levels, energetic distributions: single level, uniform, Gauss, tail, or combinations
- defect levels, optical property: direct excitation with light possible (impurity photovoltaic effect, IPV)
- defect levels, metastable transitions between defects
- contacts: work function or flat-band; optical property (reflection of transmission filter) filter
- tunneling: intra-band tunneling (within a conduction band or within a valence band); tunneling to and from interface states
- generation: either from internal calculation or from user supplied $g(x)$ file
- illumination: a variety of standard and other spectra included (AM0, AM1.5D, AM1.5G, AM1.5Gedition2, monochromatic, white,...)

- illumination: from either the *p*-side or the *n*-side; spectrum cut-off and attenuation
- working point for calculations: voltage, frequency, temperature
- the program calculates energy bands, concentrations and currents at a given working point, *J-V* characteristics, ac characteristics (*C* and *G* as function of *V* and/or *f*), spectral response (also with bias light or voltage)
- batch calculations possible; presentation of results and settings as a function of batch parameters
- loading and saving of all settings; start-up of SCAPS in a personalised configuration; a script language including a free user function
- very intuitive user interface
- a script language facility to run SCAPS from a 'script file'; all internal variables can be accessed and plotted via the script.
- a built-in curve fitting facility
- a panel for the interpretation of admittance measurements

Chapter 2: Getting started

2.1 The basics

SCAPS is a Windows-oriented program, developed with LabWindows/CVI of National Instruments. We use here the LW/CVI terminology of a 'Panel' (names used in other softwares are: a window, a page, a pop-up…). SCAPS opens with the 'Action Panel'.
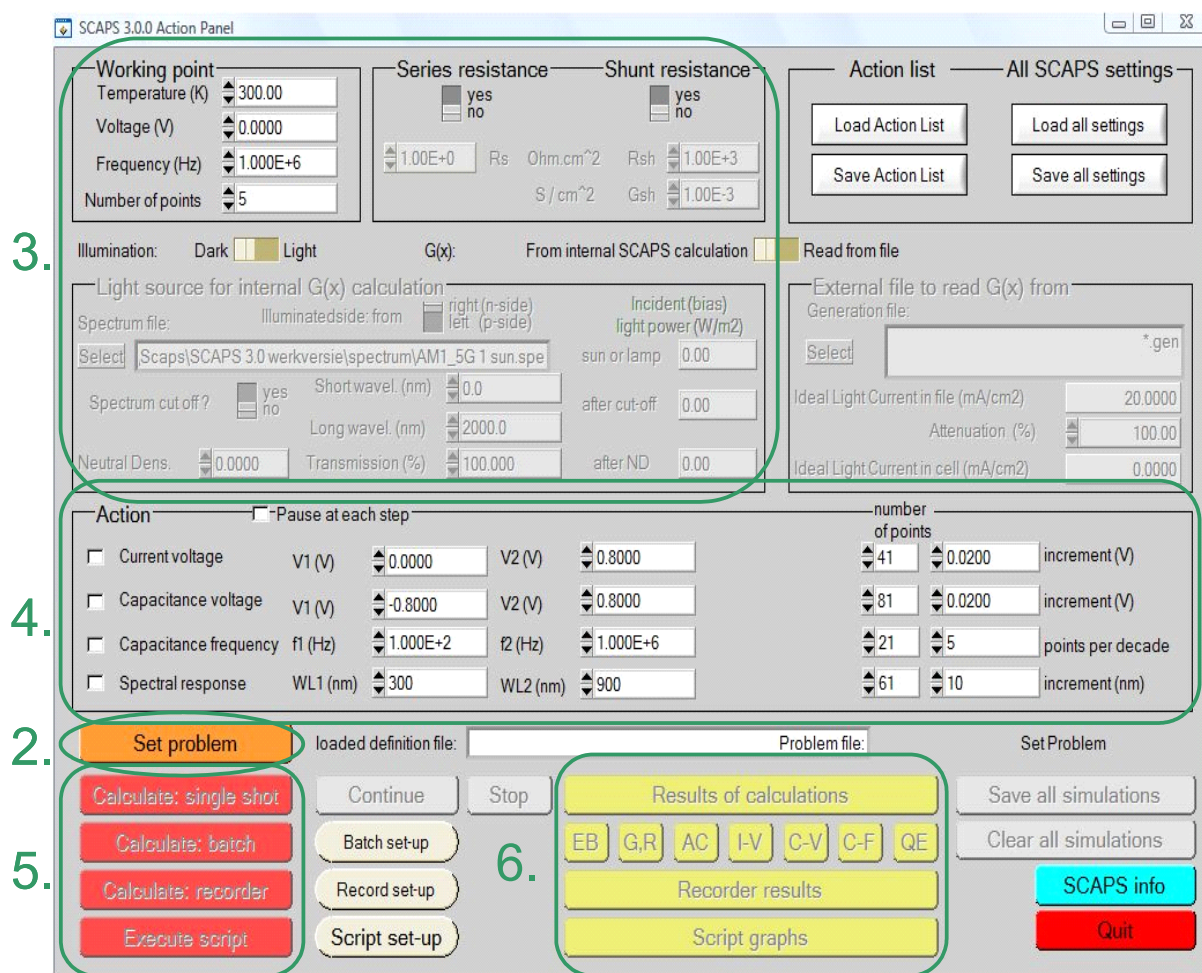
Figure 2.1   The SCAPS start-up panel: the Action panel or main panel. The meaning of the blocks numbered 1 to 6 is explained in the text.

There are dedicated panels for the basic actions:
1. Run SCAPS .
2. Define the problem, thus the geometry, the materials, all properties of your solar cell
3. Indicate the circumstances in which you want to do the simulation, i.e. specify the working point

4. Indicate what you will calculate, i.e. which measurement you will simulate.
5. Start the calculation(s)
6. Display the simulated curves, ... (see section 6)

This is further explained below.

2.2 Run SCAPS:

Click the above pictogram on the Desktop, or double-click the file `scaps3200.exe` in the file manager (or any other SCAPS version). SCAPS opens with the Action Panel.

2.3 Define the problem:

Click the button `set problem` in the action panel, and chose `load` in the lower right corner of the panel that opens. Select and open e.g. the file `NUMOS CIGS baseline.def`: that is the example problem file of the practicum session at the NUMOS workshop, Gent, 30 march 2007. This file is supposed to be in the folder `/scaps/def`, where `/scaps/` stands for the directory where you installed SCAPS, and where the SCAPS .exe file resides. If necessary, browse to find this file. In a later stage, you can alter all properties of the cell by clicking `set problem` in the action panel.
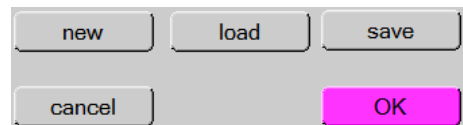
2.4 Define the working point

The working point specifies the parameters which are not varied in a measurement simulation, and which are relevant to that measurement. Thus:

- the temperature T: relevant for all measurements. Note: in SCAPS, only $N_C(T)$, $N_V(T)$, the thermal velocities, the thermal voltage kT and all their derivatives are the only variables which have an explicit temperature dependence; you must input for each T the corresponding materials parameters yourself.

- the voltage V: is discarded in I-V and C-V simulation. It is the dc-bias voltage in C-f simulation and in $QE(\lambda)$ simulation. SCAPS always starts at 0 V, and proceeds at the working point voltage in a number of steps that you also should specify.

- the frequency f: is discarded in I-V, $QE(\lambda)$ and C-f simulation. It is the frequency at which the C-V measurement is simulated.

- the illumination: is used for all measurements. For the $QE(\lambda)$ measurement, it determines the bias light conditions. The basis settings are: dark or light, choice of the illuminated side, choice of the spectrum. A one sun (= 1000 W/m^2) illumination with the 'air mass 1.5, global' spectrum is the default, but you have a large choice of monochromatic light and spectra for your specialized simulations. If you have an optical simulator at your disposal you can immediately load a generation profile as well in stead of using a spectrum.
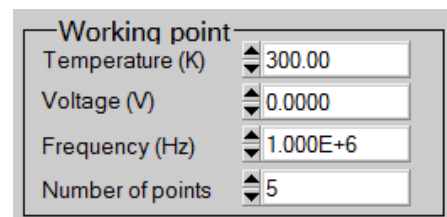
2.5 Select the measurement(s) to simulate

In the action-part of the Action Panel, you can select one or more of the following measurements to simulate: I-V, C-V, C-f and $QE(\lambda)$. Adjust if necessary the start and end values of the argument, and the number of steps. Initially, do one simulation at a time, and use rather coarse steps: your computer and/or the SCAPS program might be less fast than you hope, or your problem could be really tough... A hint: in a C-V simulation, the I-V curve is calculated as well, no need then to specify it separately.

2.6 Start the calculation(s):

Click the button `calculate: single shot` in the action panel. The Energy Bands Panel opens, and the calculations start. At the bottom of the Panel, you see a status line, e.g. "iv from 0.000 to 0.800 Volt: V = 0.550 Volt", showing you how the simulation proceeds. Meanwhile, SCAPS stands you a free movie how the conduction and valence bands, the Fermi levels and the whole caboodle are evolving. When you see the hated divergence message, you're entitled to get into a bad mood, but don't exaggerate. Anyway, you did not loose the *I-V* points already calculated.
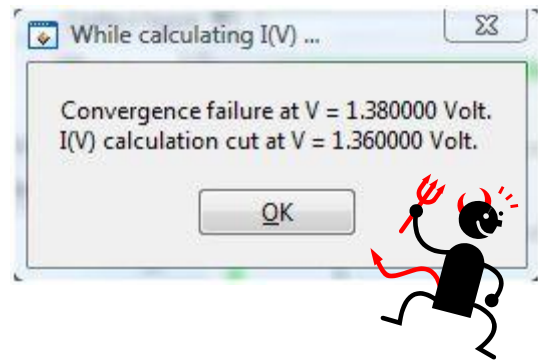
2.7 Display the simulated curves, ...

After the calculation(s), SCAPS switches to the Energy band panel (or the AC-band panel). You can now look at your ease to the band diagrams, carrier densities, current densities,… at the last bias point calculated (stop your calculations earlier, or use the pause button on the Action Panel if you want to look at an intermediate state at ease). You can output the results (buttons `print`, `save graphs`, `show` (then the numbers are shown on screen; cut & paste to e.g. Excel is possible), or `save` (then the numbers are saved to a file). You can switch to one of the specialized output Panels (if you have already simulated at least one corresponding measurement). We only show the example of the IV Panel.
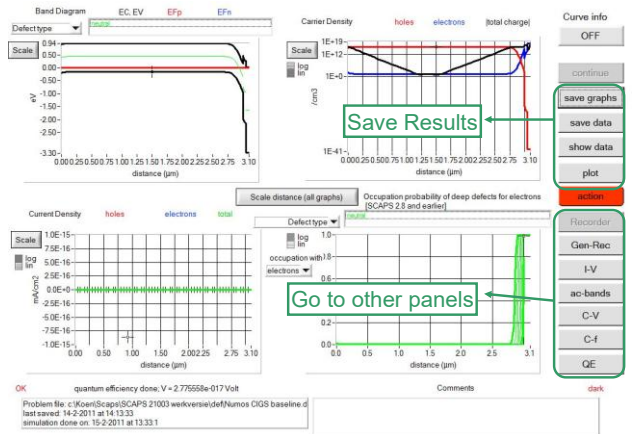
2.8 …e.g. the *I-V* curves

The meaning of the `plot`, `show` or `save` buttons is as for the Energy Bands Panel. Again, you can switch to the other output panels (energy bands, ac, *C-V*, *C-f* and *QE*, if already calculated), and to the Action Panel to do a new calculation, or to stop (important: you can only leave SCAPS from the Action Panel!). Several small remarks:

The color of the last calculated curve is indicated (tip: when the graph gets too crowded, go to the Action Panel and click `clear all simulations` to clear all graphs). The recombination curves are only shown for the last simulation. The color of the legend corresponds to the color of the curve (indicated as 1bis).

If `Curve Info` is switched `ON` and you click the cursor on a curve in a graph, a pop-up panel will appear which gives information about the graph, curve and the point which you clicked.

Here you can display a measurement file (only one measurement at a time!). Select e.g. the file `Numos Ex 1 light.iv` or `Numos Ex 1 dark.iv` which you should find under /SCAPS210/measurements.
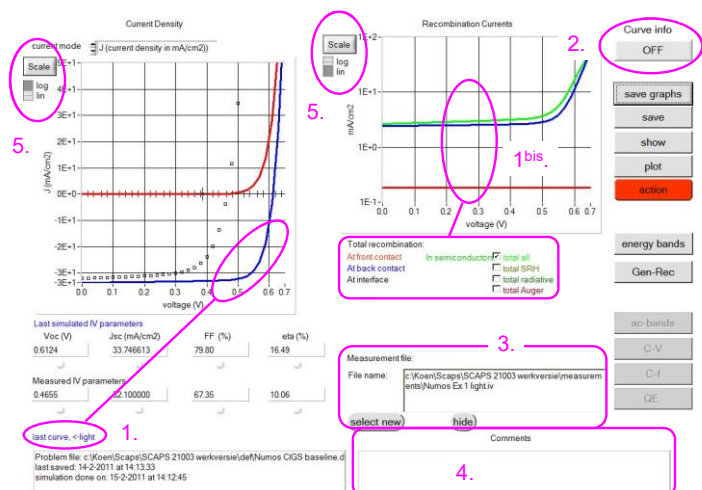
Hint: when you are doing many simulations, be friendly and helpful to yourself, and write some comments in the comment box before printing the Panel: you'll be glad to have done so when the time of writing (an article, your doctorate) comes…

You can change the range and scaling of the axes with the `Scale` button. If you press the CTRL-button and select a rectangular area in a graph, the graph will zoom-in to the selected area. Pressing the CTRL-button and clicking the right mouse button results in zooming out.

2.9 Editing the problem

Go to the Action Panel, click `set problem.` You are now in the Solar Cell Definition Panel. Click on a layer name, and you enter the Layer Properties Panel where you can change all parameters of that layer. Use your intuition and/or read the rest of this manual.

2.10 Speeding up: Batch calculations

When you want to explore the influence of one or a few parameters to the solar cell characteristics, you can take profit of the batch option. When you click `Batch set-up,` a panel opens where you can choose which parameter to vary, over which range, and in which mode (Lin, Log or custom). You can also define more than one parameter, and vary all of them (in a nested way or 'simultaneous'), but be modest to start. A batch calculation is launched when `calculate: batch` is clicked.

2.11 Speeding up: Recorder

In a regular single shot or batch calculation, the detailed panels are only available for the last measurement point. To be able to see them as a function of the batch parameters you can launch a record calculation. You should first select the properties which you want to keep track of by clicking `Record set-up`. Browse through the property-lists, and don't forget to press one of the `insert` buttons to add a property to the recorder list. By clicking `calculate: recorder`, a recorder calculation is launched. Cell parameters are varied according to the `Batch set-up`, and all simulations are performed which are needed to determine the asked properties. This means the selected measurements on the action panel are ignored!

Chapter 3: Solar cell definition

The recommended way to introduce your solar cell structure into SCAPS is to use the graphical user interface. This way you can interactively set all parameters while SCAPS watches over you, so that you don't define impossible or unrealistic situations. This chapter explains which situations can be modelled and how to introduce them in SCAPS.

3.1 Editing a solar cell structure

When clicking the 'Set Problem'-button on the action panel, the 'Solar cell definition'-panel is displayed. This panel allows to create/edit solar cell structures and to save those to or load from definition files. These definition files are standard ASCII-files with extension '*.def' which can be read with e.g. notepad. Even though the format of these files seems self-explaining it is however strongly advised not to alter them manually.

Layer-, contact-, and interface properties can be edited by clicking on the appropriate box as shown in Figure 3.1. In a similar way, layers can be added by clicking 'add layer'
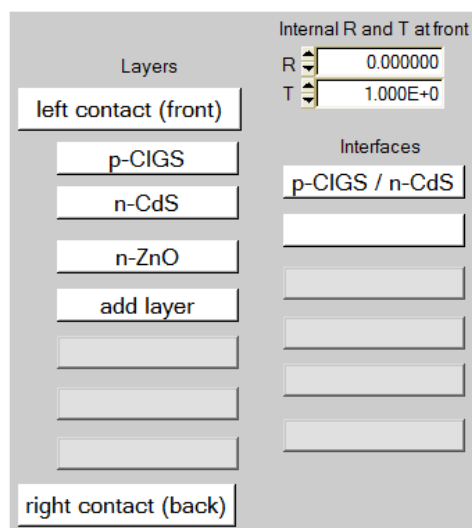
Figure 3.1   Defining a solar cell structure

3.2 Reference conventions for voltage and current

The user can input own reference conventions for the applied voltage V and the current J in the external contacts. When setting a new problem, or editing an existing problem that does not contain any reference data (e.g. an older .def file), the new options in the solar cell definition panel (Figure 3.2 right) are invisible, and the default reference conventions are set. Upon checking the option in the More Numerical Settings Panel (Figure 3.2 left), these options are visible and can be operated right away. When a newer problem is loaded that contains reference information, the checkbox 'allow change of…' is set automatically, and the

three options of Figure 3.2 right are enabled. (As of 2-1-2014, this More Numerical Settings Panel is not yet available to the user; the option "allow change of … references" is always enabled).

Figure 3.2   Setting user reference conventions for voltage and current. Left: checkbox in the More Numerical Settings Panel. Right: new facilities in the Solar Cell Definition Panel.

The three new facilities are:

1. 'apply voltage V to': when 'left' is set, then the right contact is the reference contact, and the voltage V is applied to the left contact; this is the default, and the only possible option in SCAPS<3.2.0. When 'right' is set, the left contact is the reference contact, and the voltage V is applied to the right contact; in an JV curve, this correspond to a reversal of voltage axis compared to the traditional JV curves in SCAPS.

2. 'current reference as a': when 'consumer' is set, then the current reference arrow is set such that $P = J{\times}V$ is the power consumed by the cell, and thus $- J{\times}V$ the power generated by the cell. When 'generator' is set, then the current reference arrow is set such that $P = J{\times}V$ is the power generated by the cell, and thus $- J{\times}V$ the power consumed by the cell. Setting of the current reference arrow thus depends both on the selected voltage reference and on the consumer/generator selection.

3. 'Invert the structure': the solar cell structure is mirrored along the x axis: the leftmost layer becomes the rightmost layer, and so on. This inversion of structure also swaps the interfaces, and all grading information in the layers and the defects. Clicking two times the inversion button brings the original cell back. This inversion only concerns the structure: the illumination side, the voltage and current reference settings all remain unchanged.

With these 3 settings, one can define 8 different problems, resulting in 4 different aspects of the JV curves. We illustrate this in Figure 3.3 and Figure 3.4, to make the user more familiar with the concepts of voltage, current and power references.
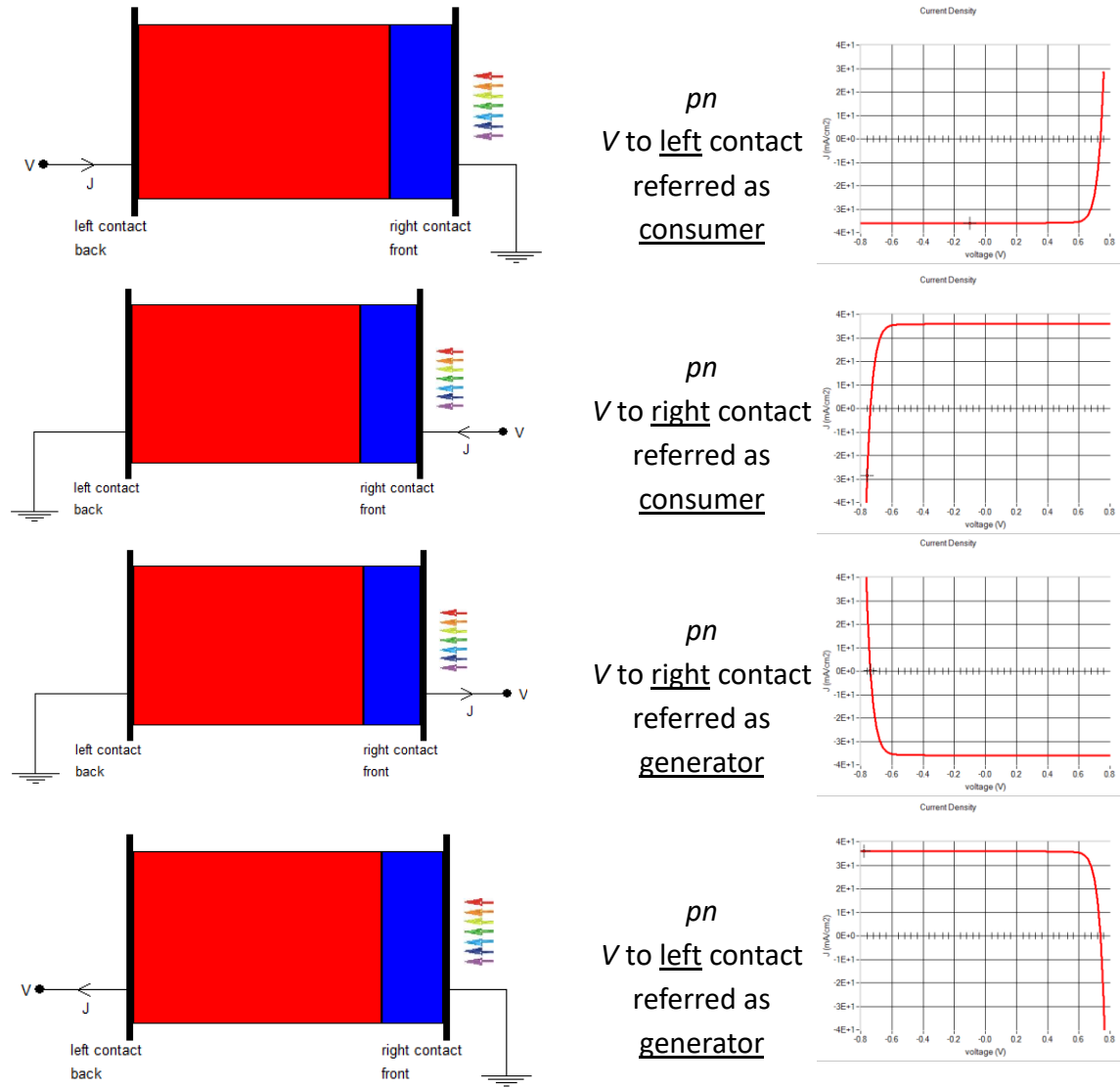
Figure 3.3　　Possible references of *V* and *J* for *pn* structures. The calculations are for the problem file simple pn.def, and the illumination is always from the right.
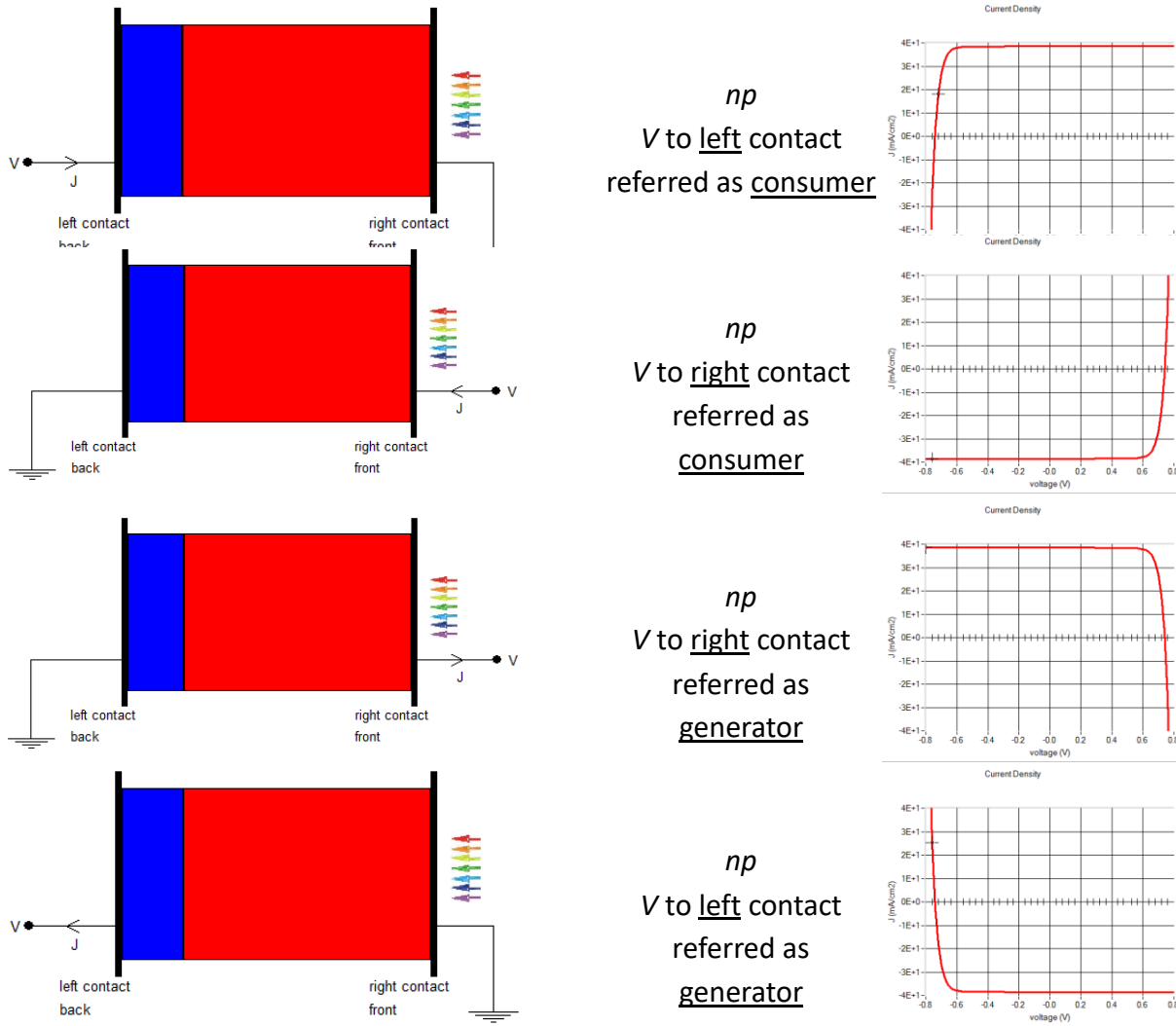
Figure 3.4   Possible references of *V* and *J* for *np* structures. The calculations are for the problem file simple pn.def, and the structure is inverted with the button 'Invert the structure'; the illumination is always from the right.

Internally in SCAPS, only the default reference is used (voltage applied at the left contact, current reference arrow from left to right, resulting in a reference as a consumer. In all output (graphs, show/save tables), the result is shown consistent with the user's choice of reference. Note that the electric field in the SCAPS output is not subject to the user-set *V* and *J* references: it is always referred to the positive *x*-axis, thus from left to right.

3.3 Contacts

The contact properties can be set by either clicking the front or back contact button on the cell definition panel, which opens the 'contact properties panel', Figure 3.5.
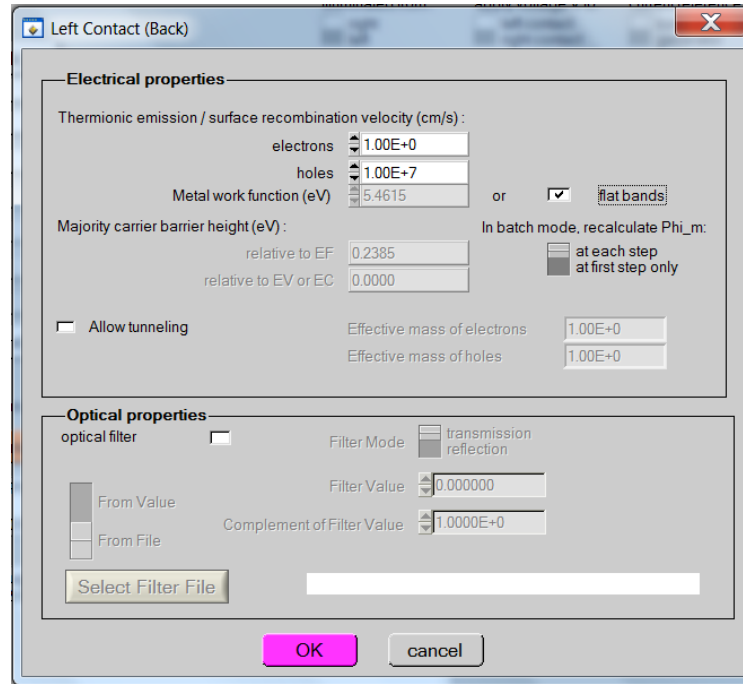
Figure 3.5   Contact properties panel.

The metal work function Φ_m (for majority carriers) can be input by the user. However, the user can also choose the option *"flat bands"*. In this case, SCAPS calculates for every temperature the metal work function Φ_m in such a way that flat-band conditions prevail. In SCAPS versions before 1-1-2014, a simplified algorithm described below was used. When the layer adjacent to the contact is *n*-type Eq.(1) is used, when it is *p*-type Eq.(2) is used, when it can be considered to be intrinsic Eq.(3) is used.

$$\Phi_m = \chi + k_B T \ln\left(\frac{N_C}{N_D - N_A}\right) \tag{1}$$

$$\Phi_m = \chi + E_g - k_B T \ln\left(\frac{N_C}{N_A - N_D}\right) \tag{2}$$

$$\Phi_m = \chi + k_B T \ln\left(\frac{N_C}{n_i}\right) \tag{3}$$

As can be seen, only the shallow doping density is taken into account in order to calculate the flat band metal work function. When there is a considerable amount of charge in defects present near the contact however, it is thus possible that the flat band option will not lead to flat bands ☹.

In recent SCAPS versions (after 1-1-2014), also charge in deep defects is considered; an algorithm involving the solution of a non-linear algebraic equation (expressing that total charge = zero) then replaces the simple equations (1) to (3).

When the layer next to the contact is either *n*- or *p*-type (NOT INTRINSIC) a recalculation of the barrier height with respect to the Fermi level and conductance/valence band are calculated and displayed in the contact properties panel. These values however only serve as an indication to the user, they are not used in the simulation.

At the contacts a (wavelength dependent) reflection/transmission $R(\lambda)$ or $T(\lambda)$ can be set, see §4.2.1. These can be set either as a constant value (wavelength independent) or as a filter file. These filter-files are standard ASCII-files with the extension '*.ftr'. Several files are provided with the SCAPS installation, however, the user can easily make more files. If a line in this file can be interpreted as starting with at least

two numeric values, the first value is interpreted as the wavelength (in nm) and the second as the reflection (in %). All other lines are ignored and treated as comment. Often a SCAPS simulation will need $R(\lambda)$ and $T(\lambda)$ at a wavelength outside the range specified in the filter file: this is e.g. the case if an $R(\lambda)$ file was specified between $\lambda = 300$ nm and $\lambda = 1100$ nm, and a simulation was asked under illumination with the default spectrum AM1_5G 1 sun.spe that is specified between 305 nm and 4045 nm. Then extrapolation will be used for the spectrum wavelengths 1100 nm $< \lambda < 4045$ nm. For the SCAPS extrapolation rules, see Section 3.5.5.3.

3.4 Layer thickness

Until SCAPS 3.3.00, January 2014, the input values of layer thickness was always in units of μm (micrometer), the thickness range was from 1 nm = 0.001 μm to 10 cm = 10^5 μm, and thickness was always displayed with three decimal digits shown. This was annoying when one had input $d = 0.0025$ μm (thus 2.5 nm): SCAPS was calculating with 2.5 nm, but the thickness display showed 0.002 μm (thus 2 nm); and input of a thickness lower than 1 nm was automatically set to 0.001 μm = 1 nm.

Since SCAPS 3.3.00, August 2014, the allowed input range of thickness is extended to 0.01 nm = 10^{-5} μm at the thin side to 1 m = 10^6 μm at the thick side. The number of decimal digits is still limited to 3, but the user can select several units for the thickness display: Å (1 Ångström = 0.1 nm), nm, μm, mm and cm. When the unit selected is not the traditional micrometer, the value and unit are displayed in magenta colour. When the display unit was set e.g. to μm, and one had input 0.00245, the display would show the value rounded off to 3 digits: 0.002 μm; but when one selects nm as display unit, 2.45 nm will be shown (see Figure 3.6). When a new problem is loaded from file, thickness will be displayed in nm if $d < 10$ nm; in μm if 10 nm $\leq d \leq 1000$ μm; and in mm if $d > 1$ mm.
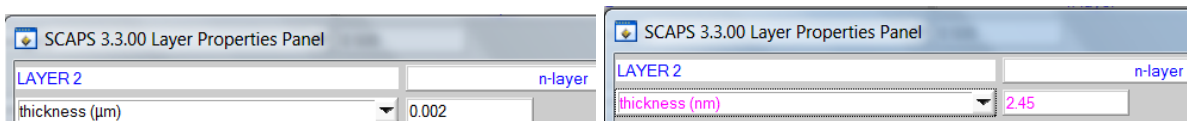
Figure 3.6   Display of layer thickness, when $d = 0.00245$ μm was input: once with μm as display unit (in black, left), and once with nm as display unit (coloured, right).

3.5 (Graded) semiconductor layers

All parameters of a semiconductor layer can be edited by clicking 'add layer' or on the appropriate layer button in the 'Solar cell definition panel'. All properties can be graded, as will be discussed below. However, some remarks not related to grading should be made first.

3.5.1 Adding, duplicating, splitting, removing layers from the cell structure

By right-clicking a layer button in the 'Solar cell definition panel', a panel opens where you can remove this layer, or duplicate it, or split it, see Figure 3.7.
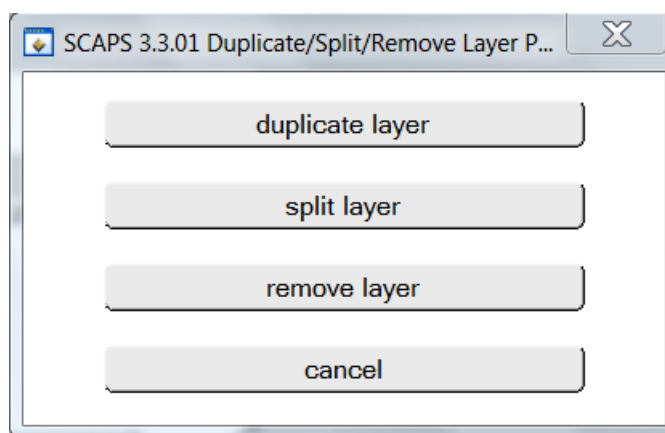
Figure 3.7   Panel to remove a layer, or to duplicate or split it.

With 'duplicate' an identical layer is inserted after (= to the right of) the layer you right-clicked; in particular, the inserted layer has the same thickness as the original layer. The split option is there from SCAPS 3.3.01, 4-3-2015 on. With split, the thickness and the grading properties (see below) of the original layer are changed. The original layer is called the 'left split layer', and the inserted layer is called the 'right split layer'. The splitting action conserves the thickness of the original layer. Upon clicking 'split' in the panel of Figure 3.7, the Split Layer Panel of Figure 3.8 opens. There, you can set the thickness of the left split layer or of the right split layer, as absolute thickness in μm or nm, or as relative thickness (a fraction of the original thickness). The name of the layer and its duplicate are changed, e.g. in the example of Figure 3.8 with original layer name 'p-layer', the left split layer will be named 'p-layer left' and the right split layer 'p-layer right'.
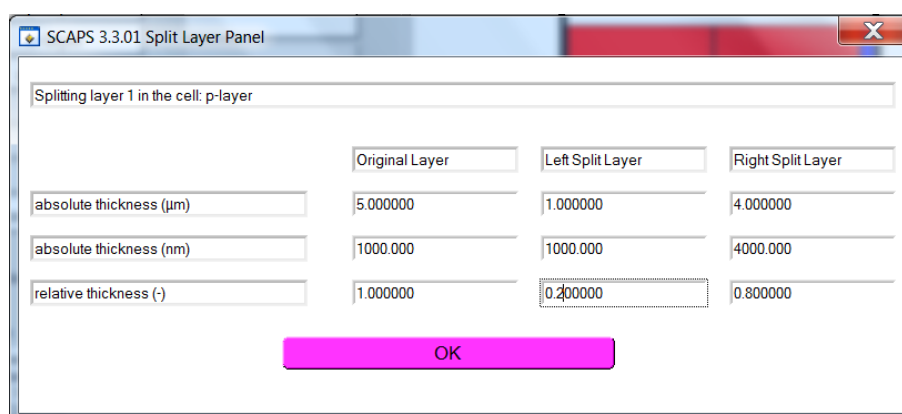
Figure 3.8   The Split Layer Panel allows to split a layer whilst conserving the total thickness and the grading properties of the original layer.

Also, the grading properties of the split layers are adapted so that the overall grading of all properties is conserved (advancing to the terminology explained in 3.5.3). This works perfect for most grading types. For 'exponential grading', a set of transcendental equations has to be solved, we hope that this will work fine for all values you would try, but there is no guarantee. For 'Beta-function grading', it is mathematically not possible to find Beta-function solutions for the grading in both split layers. SCAPS will set the Beta-function a and b of the split layers in a very rough approximation, that will be unsatisfactory in many cases, especially when $a > 1$ and $b > 1$. It is up to the user to check the results, and judge if they are acceptable or not.

For 'grading from file', two new grading files are created for the split layers, and their names are the original grading filename preceded by 'leftsplit_' or 'rightsplit_'. Exception: (SCAPS 3.3.03, february 2016) When the grading file has the *range: shared over adjacent layers* keyword, the grading file is used without modification in the two split layers; no two files with different names are created.

When there is no space to duplicate or split a layer, i.e. when the maximum number of layers is already in use (actually 7), the duplicate and split buttons are disabled (and have an appropriate label). Also, you cannot remove a unique layer from a cell structure; when you try, the remove button will be disabled and have an appropriate label.

3.5.2 Temperature dependence of parameters

The density of states in the conduction/valence band are temperature dependent according to (4). T_0 is the default temperature (at which the parameter value should be defined in SCAPS and equals 300 K). Likewise the thermal velocity is temperature dependent (5). All other parameters are assumed to be temperature independent. The diffusion coefficient $D=\mu kT/q$ which is used in the calculations is temperature dependent (contrary to what was mentioned in the very first SCAPS manual (version 2.0))

$$N_C(T) = N_C(T_0)\left(\frac{T}{T_0}\right)^{1.5}$$

$$N_V(T) = N_V(T_0)\left(\frac{T}{T_0}\right)^{1.5} \tag{4}$$

$$v_{th}(T) = v_{th}(T_0)\left(\frac{T}{T_0}\right)^{0.5} \tag{5}$$

Should you want to give a temperature dependence to other parameters, e.g. $E_g(T)$, $\mu_n(T)$, $\mu_p(T)$, …, it is entirely your task. The way to do so is:

In the Batch Set-up, define the temperature T as a batch parameter. Define for example E_g as a next, 'simultaneous' batch parameter, and set its value from an $E_g(T)$ list (should the temperatures T in your list not form a regular series, you should set the T values also from a list). You should take care to save both lists and maintain them together.

3.5.3 Grading: general approach

All layer-parameters can be graded. The principles of the algorithms used to simulate graded solar cell structures have been presented in [3]. To give a suitable and materials oriented description of the grading of the various materials parameters, SCAPS derives all parameters consistently from the composition grading of a layer. Each layer is assumed to have composition $A_{1-y}B_y$. The user defines the properties of the pure compounds A (e.g. A = CuInSe$_2$) and B (e.g. B = CuInS$_2$), and the composition grading $y(x)$ over the thickness of the layer: thus defining the composition values y at the left and right side of the layer, and by specifying some grading law in between. All materials properties P are then derived from the local composition parameter $y(x)$, that is, $P[y(x)]$ is evaluated. Several grading laws are implemented in SCAPS and offered by the user interface: uniform, linear, logarithmic, parabolic (two laws), power law, exponential, effective medium, from file and a Beta function. These grading laws can be used to set the composition grading $y(x)$ over a layer, as well as to set the composition dependence $P(y)$ of a property. These grading laws and their parameters can be set on the Grading panel, see Figure 3.9. The grading lays used are summarized in Table 3.1.

Table 3.1   Available basic grading laws. SCAPS takes care of possible numerical problem, that could occur for very small or very large arguments; hence these laws are not always strictly followed but are more complicated under specific circumstances.

Name	$P(y) = \cdots$	remarks
Uniform	$P_A = P_B$	

Linear

$$\frac{P_A\left(y_B - y\right) + P_B\left(y - y_A\right)}{y_B - y_A}$$

Parabolic

$$\frac{P_A\left(y_B - y\right) + P_B\left(y - y_A\right)}{y_B - y_A} - b\frac{\left(y_B - y\right)\left(y - y_A\right)}{\left(y_B - y_A\right)^2}$$

b: bowing factor

Parabolic2

$$y < y_0 : P_0 + \left(P_A - P_0\right)\left(\frac{y - y_0}{y_A - y_0}\right)^2$$

$$y > y_0 : P_0 + \left(P_B - P_0\right)\left(\frac{y - y_0}{y_B - y_0}\right)^2$$

There are two parabolas, one to each side of the point $[y_0, P_0]$ which can be given in the user interface.

For extrapolation a fourth order equation is used

Logarithmic

$$P_A^{\frac{y_B - y}{y_B - y_A}} \times P_B^{\frac{y - y_A}{y_B - y_A}}$$

Exponential

$$P_0 + \left(P_A - P_0\right)\frac{\sinh\left(\dfrac{y_B - y}{L_A}\right)}{\sinh\left(\dfrac{y_B - y_A}{L_A}\right)} + \left(P_B - P_0\right)\frac{\sinh\left(\dfrac{y - y_A}{L_B}\right)}{\sinh\left(\dfrac{y_B - y_A}{L_B}\right)}$$

P_0: background value
$L_{A,B}$: characteristic lengths

Beta function

$$P_A + \left(P_A - P_B\right)\beta_{a,b}\left(\frac{y - y_A}{y_B - y_A}\right)$$

$\beta_{a,b}(x)$ is the incomplete beta-function ($I_x(a,b)$ in [7] on p.226)

Power law

$$\left(P_A^{1/m} \times \frac{y_B - y}{y_B - y_A} + P_B^{1/m} \times \frac{y - y_A}{y_B - y_A}\right)^m$$

m: power

Effective medium

$$\left(y_B - y\right)\frac{P_A - P}{P_A + 2P} + \left(y - y_A\right)\frac{P_B - P}{P_B + 2P} = 0 \text{ , thus:}$$

$$P = \frac{b + \sqrt{b^2 + 8P_A P_B}}{4} \text{ with}$$

$$b = \frac{\left(y - y_A\right)\left(2P_B - P_A\right) + \left(y_B - y\right)\left(2P_A - P_B\right)}{y_B - y_A}$$

Not available for composition.
It is the Bruggeman equation.

Two diffusion-type grading laws were implemented in SCAPS 3.3.06, december 2017. They are only available for graded densities, dependent on position x (thus not: dependent on composition y), thus for shallow doping densities $N_D(x)$, $N_A(x)$ and for defect densities $N_t(x)$. In both laws, the input parameter x_{diff} represents a diffusion depth, and is given by $x_{\text{diff}} = 2\sqrt{D\left(T_{\text{diff}}\right)t_{\text{diff}}}$.

Diffusion: Gauß law

$$N\left(x\right) = \frac{N_{\text{tot}}}{\dfrac{\sqrt{\pi}}{2}x_{\text{diff}}}\exp\left(-\frac{\left(x - x_{\text{proj}}\right)^2}{x_{\text{diff}}^2}\right)$$

For dopants or impurities diffusing in a layer from a limited source, that contains a total amount of impurities N_{tot} (cm^{-2}). The parameter x_{proj} is the 'projected range'. It describes the penetration of the (maximum position of) the ion beam in an ion implantation process. For the diffusion depth x_{diff}, there is a sign convention:

- if diffusion is from the

left side of the layer into the right side, a negative value for the diffusion depth x_{diff} should be input.

- if diffusion is from the right side of the layer into the left side, a positive value for the diffusion depth x_{diff} should be input.

Diffusion: complementary error function law

$$N(x) = N_s \operatorname{erfc}\left(\frac{x}{x_{\text{diff}}}\right)$$

When impurities diffuse from a source that keeps the surface density constant at N_s (cm^{-3}; either $N_s = N_{\text{left}}$ or $N_s = N_{\text{right}}$). The same convention for the sign of x_{diff} holds.
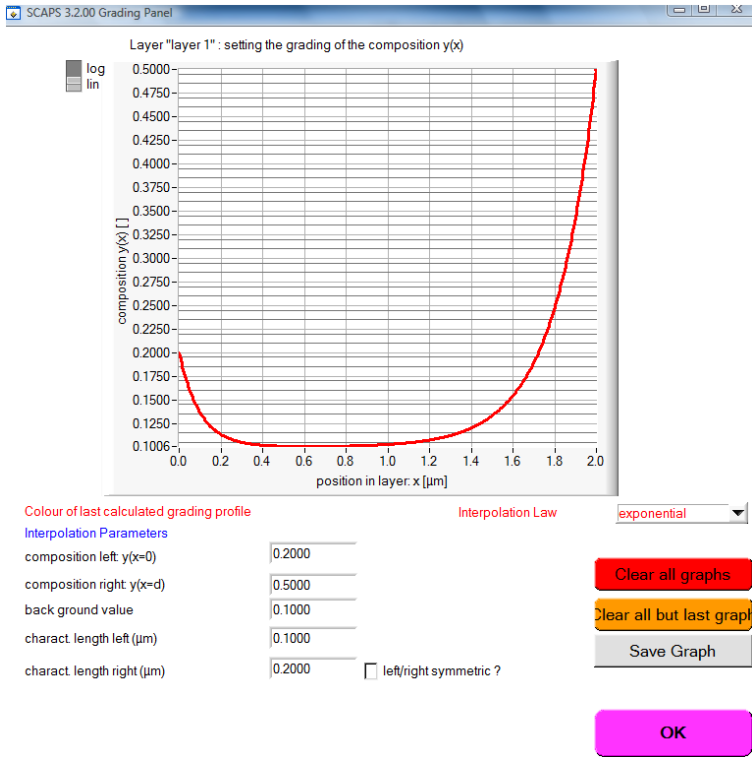
Figure 3.9   The Grading panel, in this example an exponential composition grading is set.

3.5.4 Composition grading

Composition grading $y(x)$ is the basic grading of the layer and has extra possible grading laws: the definition of uniform is somewhat more complicated than for parameter grading and a the grading can be loaded from a file. The composition grading can be set by clicking the 'Layer composition grading type', see Figure 3.10, which displays the 'Grading Panel'

Figure 3.10   Setting the composition grading $y(x)$. Select a grading law for the composition (in this example exponential). The values $y_{left}=0.2$ and $y_{right}=0.5$ are just indications here. You can only set them on the grading panel which appears when selecting a grading law.

3.5.4.2 Uniform composition grading

There are three possible definitions of 'uniform':

- 'uniform pure A ($y = 0$)'. The composition in this layer is $y = 0$ for all positions x. You see only the column of the materials properties of the pure material A ($y = 0$), with no button available to set a grading of these parameters. All parameters p get the value p ($y = 0$). Position grading of the doping and defect density is still possible, e.g. $N_A(x)$,…

- 'uniform pure B ($y = 1$)'. The composition in this layer is $y = 1$ for all positions x. You see only the column of the materials properties of the pure material B ($y = 1$), with no button available to set a grading of these parameters. All parameters p get the value p ($y = 1$). Position grading of the doping and defect density is still possible, e.g. $N_A(x)$,…

- 'uniform y, $0 < y < 1$'. The composition in this layer is $y =$ constant for all positions x, and you can set this constant composition in the grading panel. You see both columns of the materials properties of the pure material A ($y = 0$) and B ($y = 1$), and you can set a grading of each of these parameters, to give them the uniform value p (y).

   Even though it is possible to set a (position dependent) grading of doping and defect densities when the composition grading of the layer is either uniform A or uniform B, it is strongly advised to use the uniform y-option.

3.5.4.3 Composition grading from file

The composition grading profile can be loaded from a file. The rules and conventions used are the same as for specifying parameter grading 'from file', and are described further in section 3.5.5.2.

3.5.5 Parameter grading

3.5.5.1 Position dependent parameter grading

Materials physics impose that we should implement also position dependent grading (and not only composition dependent grading) as an option for the shallow doping densities N_D and N_A, and for the defect densities N_t.

ND grading dependent on composition y: ND (y)	▼		□ Allow Tunneling	
shallow donor density ND(y) (1/cm3)	1.000E+14	1.000E+16	logarithmic	▼
NA grading dependent on position x: NA (x)	▼			
shallow acceptor density NA(x) (1/cm3)	1.000E+15	1.000E+14	exponential	▼

Figure 3.11   Here, the shallow donor density is given a grading as a function of composition y: $N_D(y)$. The two values displayed are the values of the pure A material (10^{14} cm^{-3}) and the pure B material (10^{16} cm^{-3}). The shallow acceptor density $N_A(x)$ is given a grading as a function of position x: $N_A(x)$. The two values displayed are the values at the left side (10^{15} cm^{-3}) and at the right side (10^{14} cm^{-3}). Notice the use of the colour code (red for position, blue for composition grading).

3.5.5.2   Grading 'from file'

All graded profiles (composition grading $y(x)$ and parameter grading $P(y)$ or $P(x)$) can be specified 'from file'. The (ASCII-text) files containing these profiles should be saved in the folder 'grading', and the default extension is .grd. The file which contains the grading profile should consist of two columns of numerical data, the first number representing the position in the layer in µm, the second representing the composition (usually a number between 0 and 1). All lines which cannot be interpreted as two numerical data are treated as comment. Make sure that the data are in the first column are in ascending order!

When the grading file is used to specify grading of a parameter as a function of composition y, thus $P(y)$, the first column is interpreted as the composition. Normally, only the values between $y = 0$ and $y = 1$ are used (see remark in the next section). If necessary, SCAPS will extrapolate the file data to obtain the property for the pure A material ($y = 0$) and for the pure B material ($y = 1$).

When the grading file is used to specify grading of a parameter as a function of position x, thus $P(x)$, the first column is by default interpreted as the position x in µm. Note that this position-dependent grading is only possible for the composition $y(x)$, for the doping densities $N_A(x)$ and $N_D(x)$ and for the defect densities $N_t(x)$. If the layer thickness exceeds the range of the first column, extrapolation is used. However, you can also force SCAPS to interpret the first column as the relative position x/d, that is the position relative to the layer thickness. In that case, only the data between 0 and 1 are used. To specify whether the first column is the absolute or the relative position, scaled to the thickness, insert a line in the grading file, starting with:

`position: absolute` (the default), or:

`position: relative`

To get the property of the x or y values required by the program, interpolation between the file input values is used. This interpolation can be linear, or 'logarithmic' (that is, the property is first 'plotted' in a logarithmic plot, and is then linearly interpolated). To specify which interpolation mode to use, insert a line in the grading file, starting with:

`interpolation: linear` (this is the default for y, E_g, χ, ε, μ, m_{eff}), or:

`interpolation: logarithmic` (this is the default for N_C, N_V, N_D, N_A, N_t, v_{th}, C_{rad}, C_{Auger})

Addendum SCAPS 3.3.03, february 2016. There is an option to apply one and the same grading file over several layers. You can specify this by inserting a line in the grading file, starting with:

`range: this layer only` (the default), or:

`range: shared over adjacent layers`

The grading file is checked for the occurrence of 'range: this' or 'range: shared' only. When the shared option is set, the grading file applies over all layers of a group of adjacent layers:

- o   it can only be applied to properties that are graded as a function of position x, not as a function of composition y. These are: $y(x)$, $N_A((x)$ and $N_D(x)$. As of now, not $N_t(x)$.
- o   the layers in the group should be adjacent to each other
- o   'grading from file' must be set in all layers of the group

- o the same grading file (exactly the same name, thus the same file) must be set in all layers of the group
- o this grading from file must be set for the same property: all $y(x)$, or all $N_A(x)$, …

The rules for extrapolation, should it be necessary, are described in the next paragraph. They apply to all interpolation of file input data in SCAPS, in particular also to absorption $\alpha(\lambda)$ files.

3.5.5.3 Extrapolation conventions in SCAPS

Extrapolations can be necessary in several SCAPS input files:

- o Files specyfying the wavelength dependence of an input property: absorption files $\alpha(\lambda)$; spectrum files Spec(λ); filter files for reflection $R(\lambda)$ or transmission $T(\lambda)$; optical capture cross sections for the IPV effect, $\sigma_{n,\text{opt}}(\lambda)$ and $\sigma_{p,\text{opt}}(\lambda)$. When these data are needed outside the λ range specified in the file, extrapolation is used: linear extrapolation for Spec(λ), and filter files $R(\lambda)$ or $T(\lambda)$. And logarithmic extrapolation for $\alpha(\lambda)$, $\sigma_{n,\text{opt}}(\lambda)$ and $\sigma_{p,\text{opt}}(\lambda)$ (that is, first plot α or σ on a logarithmic scale, and then extrapolate linearly).
- o Files specifying the position dependence of an input property: generation files $G(x)$, composition $y(x)$, general grading files Property(x) or Property(y) (where property might be E_g, χ, N_C, …, N_A, N_D, N_t, …). When these data are needed outside the position x range (or composition range y) specified in the file, extrapolation is used: linear extrapolation for y, E_g, χ, ε, μ_n, μ_p. And logarithmic extrapolation for G, N_C, $v_{th,n}$, $v_{th,p}$, C_r (direct recombination), C_n and C_p (Auger recombination), N_V, N_A, N_D, N_t.

To prevent that extrapolated data shoot out to infinity, SCAPS applies a few precautions when logarithmic extrapolation is asked. Therefore, the first two points and the last two points in a file table are examined. The basic rule is that an extrapolated property is not allowed to increase when going away from either side of the specified interval. Additionally, when an end point is zero, all values before or after this end point are set to zero. These rules have been adapted and refined somewhat in the evolution of the SCAPS versions. Therefore, slightly different results can be obtained with different versions; a main cause is a different extrapolation of $\alpha(\lambda)$ files for long wavelengths. Figure 3.12 below illustrates these extrapolation conventions. Tip: you can make sure that values outside the specified range are extrapolated to zero by adding manually a zero end point to your data file.

Notice that a property can get a negative value by extrapolation when linear interpolation is used. We allow this because we want to support users who run simulations with a negative composition $y < 0$. For example, a user could define Ga$_{0.75}$Al$_{0.25}$As as the 'pure A' material, and Ga$_{0.25}$Al$_{0.75}$As as the 'pure B' material. Then $y = -0.5$ corresponds to GaAs and $y = 1.5$ corresponds to AlAs. There is little chance that a user would like to do so with the well documented Ga-Al-As system, but e.g. with the still largely unknown CZTS system, we cannot foresee for which compositions $\alpha(\lambda)$ data would be available to a user, and for which compositions she or he would like to simulate…
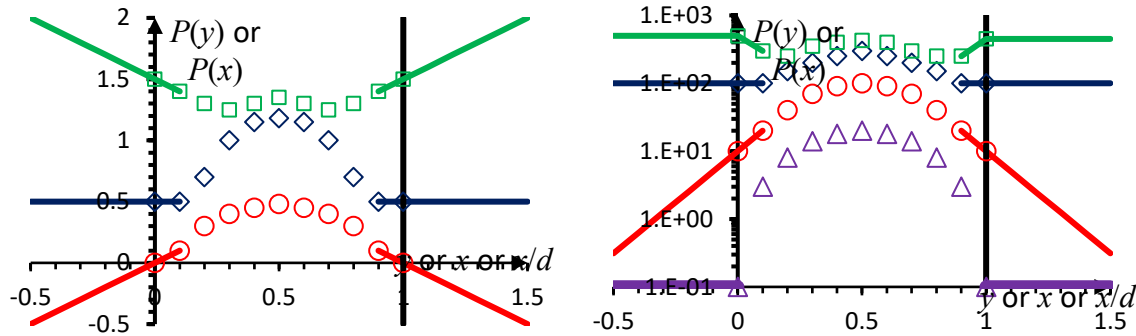
Figure 3.12 Conventions for extrapolation of file input data in SCAPS. The definition interval is between the two vertical thick lines. The data specified in the file are shown with open symbols; the SCAPS extrapolation with solid lines of the same colour. Left: 'Linear interpolation'. Extrapolation is also linear, and notice that the extrapolated values can be negative. Right: 'Logarithmic interpolation'. Extrapolation is also logarithmic, but with the restriction that P cannot rise. For the lowest curve, $P = 0$ at the edges $y = 0$ and $y = 1$ of the specified abscissa range, and the extrapolation yields $P = 0$ outside the specified abscissa range; these values are shown here as $P = 0.1$, just to make them appear in the figure, but internally they are set rigourously to $P = 0$.

==Note from SCAPS 3.3.07 of december 2019:==

Up to december 2019 there was a SCAPS requirement that the input files were strictly ordered to increasing values of the independent variable (the wavelength λ, or the position x, or the composition y); and that no duplicate values (two or more values with exactly the same λ or x or y) occurred. From december 2019, SCAPS eliminates double occurrences (by retaining the first occurrence, and discarding all next occurrences), and then orders the data to increasing λ or x or y (internally, the input files are not changed). This will make SCAPS to run with your input file, which is a clear advantage to earlier SCAPS versions. However, the user should realise that double occurrences or ordering mistakes should warn a user against a possibly incorrect or unintended input file. Also, what SCAPS is now doing with these deficiencies (automatic sorting, and removing all but the first occurrences of duplicate points) might not be what the user intended. Therefore, a warning is given to the user (see below).

Also, all these input files (with the exception of grading files, see the above remarks about $y < 0$ or $y > 1$) are now given a value check: SCAPS forces the values to be ≥ 0 before extrapolation; and the extrapolated values of the $R(\lambda)$ and $T(\lambda)$ files are forced in the range $0 \leq R,T \leq 1$. Again, while SCAPS will now run with every $R(\lambda)$ or $T(\lambda)$ file, and sends a warning to the user. We advise the user to add one or more extra points at the short and long λ end of these files, to take over the control of the extrapolation, instead of leaving it to SCAPS.

These new checks and warnings (thus: duplicates, sorting errors, range errors) are performed:

o   <u>not immediately</u> while loading the .spe, .gen, .abs, .opt and .ftr files from the command line, or from a .def or .scaps file: after this load action, you are still in the Action Panel, and have the occasion to change all these files to your taste.

o   but after pressing the OK button in the Action Panel: then your choice of input files is definitive

o   the range checking for the filter files (R and or T) however is done at each calculation: only then, the wavelength range is known, and SCAPS can know whether or not extrapolation will lead to extrapolated values outside the range [0, 1]. In cases you will be given more than one warning, as one click to the calculate button can launch several calculations, e.g.: the work-point calculation under illumination (then the λ range is that of the spectrum file, in a typical case) <u>and</u> a QE simulation (then the λ range of the QE setting is merged with the λ range of the work point (if under illumination), and can thus differ from the previous λ range).

- these warnings are by default given to the screen, and the user should acknowledge the warning (thus, press OK) to proceed. The warnings can also be directed to a log file, without interruption for the user (this is the default setting while running a script). See Figure 7.5.
- if you are annoyed with these range-warnings, there is only one good solution: edit the filter file, and add extra points, so that extrapolation in whatever λ range you will consider will not shoot out of [0, 100 %].
- and there is a worse, but faster solution: you can silence all inter/extrapolation warnings in the Numerical Panel (Figure 3.13). This setting is saved in the .scaps files (but not in the .def files!), and will be loaded from these. Older .def files will not have this setting saved, and the default setting will be used: 'do give warnings'.
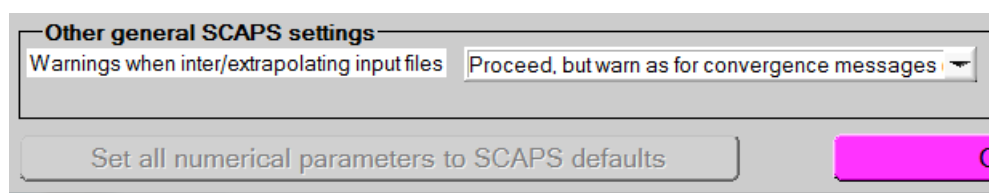
Figure 3.13   SCAPS 3.3.07, december 2019: new option at the bottom left of the Numerical Panel. The default setting is shown. The other setting is "Do not warn and proceed".

3.5.6 The optical absorption constant $\alpha(\lambda)$ or $\alpha(h\nu)$ of a layer

3.5.6.1   The optical absorption constant α: 'from file'

SCAPS absorption files are ASCII-files (simple text files) with the extension '*.abs', and they reside by default in the [SCAPS]\absorption subdirectory. If a line in this file can be interpreted as starting with at least two numeric values, the first value is interpreted as the wavelength (in nm) and the second as α (in 1/m). Note: as in all SCAPS input files, SI units are required, with the exception of nm for wavelength λ and μm for position *x*. All other lines of an .abs input file are ignored and treated as comment. The interpolation and extrapolation rules of section 3.5.5.3 apply, and α-interpolation is always 'logarithmic', not 'linear'.

A limited library of absorption files which are present in the SCAPS distribution; you will notice that we use the comment lines to document the source of the $\alpha(\lambda)$ data, thus they contain a reference. Next to absorption data of real materials, we also offer a set of .abs files that can be used in modeling work more oriented to theory: there is a set of 'Gray *x*E*x*.abs'-absorption files which implement a wavelength independent α. Since SCAPS ≥ 3.3.07, it is (much!) more comfortable to use one of the new SCAPS absorption models to this purpose (see section below).

With the instructions above you can easily make your personal absorption files. Just do not forget toput α in 1/m (not 1/cm), and we cannot enough advise to use the comment lines to document the $\alpha(\lambda)$ data file (for your own later comfort, not for SCAPS…).

3.5.6.2   The optical absorption constant α: traditional model (SCAPS ≤ 3.3.06, december 2016)

The optical absorption constant can be set from either from a model or from a file, see Figure 3.14. When it is set from a model, α(λ) is given by (6).

Figure 3.14   Setting the absorption constant from file (left) or from model (right). The grading can be shown or set by clicking the 'Show/set alpha'-button.

$$\alpha(\lambda) = \left(A + \frac{B}{h\nu} \right) \sqrt{h\nu - E_g} \tag{6}$$

Here E_g is the actual band gap of the material, and A (in cm^{-1}eV$^{-1/2}$) and B (in cm^{-1}eV$^{+1/2}$) are the model parameters.

3.5.6.3 The optical absorption constant α: new models (SCAPS ≥ 3.3.07, january 2018)

The new SCAPS models for optical absorption constant are discussed and illustrated in detail in a dedicated *application note*: SCAPS Application Note Absorption Models.pdf, downloadable from our site. Here we give a brief description, refer to the application note for more information.

Components of the new SCAPS model for optical absorption

SCAPS ≥ 3.3.07 offers 6 sub-model for optical absorption that each can be present or not. These sub-models are described by:

$$\begin{aligned}
\alpha(h\nu) = \quad & (+) \quad \alpha_{bg} & \text{back ground} \quad & \text{constant } \alpha \\
& (+) \quad \alpha_c \cdot u(h\nu - E_g) & E_g\text{-step} \quad & \alpha_c \text{ if } h\nu > E_g, \ 0 \text{ if } h\nu < E_g \\
& (+) \quad \left(\alpha_0 + \beta_0 \frac{E_g}{h\nu} \right) \sqrt{\frac{h\nu}{E_g} - 1} & E_g\text{-sqrt} \quad & \text{and } 0 \text{ if } h\nu < E_g \\
& (+) \quad \left(\alpha_1 + \beta_1 \frac{E_{g1}}{h\nu} \right) \left(\frac{h\nu}{E_{g1}} - 1 \right)^{n_1} & \text{power1} \quad & \text{and } 0 \text{ if } h\nu < E_{g1} \\
& (+) \quad \left(\alpha_2 + \beta_2 \frac{E_{g2}}{h\nu} \right) \left(\frac{h\nu}{E_{g2}} - 1 \right)^{n_2} & \text{power2} \quad & \text{and } 0 \text{ if } h\nu < E_{g2} \\
& (+) \quad \approx \exp\left(-\frac{E_g - h\nu}{E_0} \right) & \text{sub-bandgap} \quad & \text{"glued to other mechanisms"}
\end{aligned} \tag{7}$$

Here (+) means that the sub-model can be omitted or can be added by the user to the total $\alpha(h\nu)$ in the SCAPS user interface. The names 'back ground', 'E$_g$-step'… are the sub-model names that are used by SCAPS. A few remarks should be made:

- There should always be at least one absorption sub-model active. When the user tries to unclick all sub-models in the SCAPS user interface (see below), the back ground model will be forced to be present.
- The model 'Eg-step' is there mainly to please theoreticians. It delivers a constant absorption constant α_c 'above the band gap', thus $h\nu > E_g$ or $\lambda < \lambda_g$, and zero α 'below the band gap'. The band gap is always

the band gap E_g that was input as one of the electronic properties of the material/layer. Hence it will also be varied when E_g is varied as a batch parameter, or in the script.

- The 'Eg-sqrt' model is the only model that was implemented in traditional SCAPS ≤ 3.3.06. In the new SCAPS ≥ 3.3.07, this model can be active or not, as the user decides. The user interface shows the model parameters α_0 and β_0 also in their traditional form A and B. The band gap is always the band gap E_g that was input as one of the electronic properties of the material/layer. Hence it will also be varied when E_g is varied as a batch parameter, or in the script.

- The two power models 'power1' and 'power2' allow much versatility in modelling the $\alpha(h\nu)$ or $\alpha(\lambda)$ behaviour of a layer/material. There are 4 parameters per 'power model': α_1, β_1, E_{g1} and n_1 (and alike for power 2, with index 2). In the user interface panel, λ_{g1} is given as an alternative parameter for E_{g1}: change one of the two, and the other will be adapted in the user interface.

- <u>Important note on the meaning of the band gap parameters E_{g1} and E_{g2}:</u>
  - When E_{g1} (or E_{g2}) are positive, they should be understood as a fixed parameter: they are not related to the actual band gap E_g of the layer/material. In particular, this means that also when E_g is varied (batch or script), the parameter E_{g1} (or E_{g2}) keeps its fixed value.
  - When E_{g1} (or E_{g2}) are input as a negative number, they should be understood as a 'relative band gap', or a multiple of the actual E_g: their value is substituted internally with $|E_{g1}| \times E_g$. When E_g is varied then (batch or script), the internal parameters E_{g1} (or E_{g2}) are varied with it.
  - Example: take a layer/material with band gap $E_g = 1.2$ eV. When setting $E_{g1} = 1.5$, the internal E_{g1} parameter will always be 1.5 eV, regardless of the changes made to E_g. But when the input was $E_{g1} = -1.5$, the internal E_{g1} value is set to 1.5×1.2 eV = 1.8 eV; and when one would vary E_g in a batch calculation from 1 eV to 2 eV, the internal E_{g1} would vary from 1.5 eV to 3.0 eV.

- The sub-bandgap tail mechanism is (of course) only available when the user has checked at least one of the 4 'band gap mechanisms' Eg-step, Eg-sqrt, power1 and power2.

- The proportionality constant in the last term of Eq. (7) is chosen in such a way that the sub-band gap tail is 'smoothly glued' to the $\alpha(h\nu)$ or $\alpha(\lambda)$ curve of all other mechanisms.

- The Eg-sqrt mechanism of the new SCAPS works with the model parameters α_0 and β_0, not with A and B. However, traditional SCAPS definition files will be read and the $(A, B) \rightarrow (\alpha_0, \beta_0)$ conversion will be done automatically.

- When the Eg-sqrt model is the only 'band gap model' present, the new SCAPS ≥ 3.3.07 will also output the (A, B) parameters, and the new definition files will be read and treated correctly by traditional SCAPS versions.

- Also when other band gap models (than the Eg-sqrt model) are present, the new SCAPS versions will output some 'best' (A, B) values. Traditional SCAPS versions will run with these new definition files, but of course will not give exactly the same result as the new SCAPS versions, since they do not have the Eg-step, power1 or power2 models.

The SCAPS user interface for absorption models

The optical absorption block

The optical absorption block in the Layer Properties Panel has been changed in the new SCAPS ≥ 3.3.07, see Figure 3.15. When the optical absorption of a material/layer is set to 'from model', the 'set absorption model' button is active; upon clicking, the Absorption Model Panel opens (Figure 3.16). This panel is operating much like the grading panel and the (spectrum or generation) model panel, that are perhaps familiar to the SCAPS user.
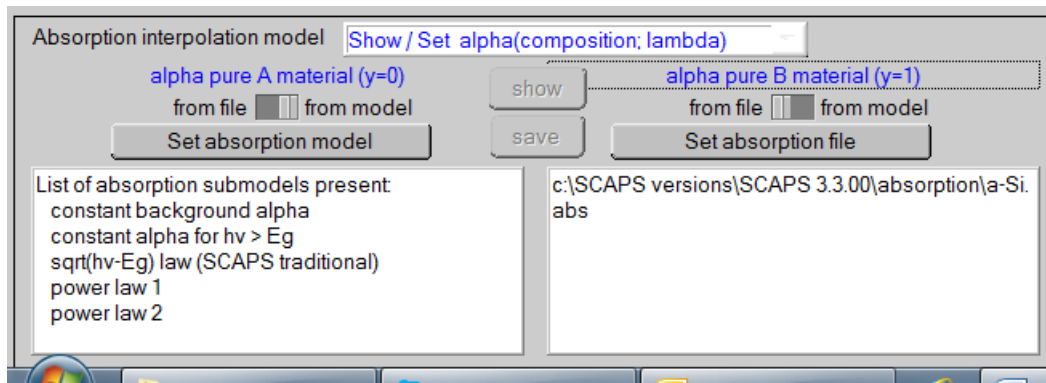
Figure 3.15   The optical absorption block in the Layer Properties Panel in the new SCAPS versions (≥ 3.3.07). For the 'pure A material', absorption 'from model' is set, and a summary of the absorption models present is shown. For the pure B material', absorption 'from file' is set, and the full path of the absorption file is shown.
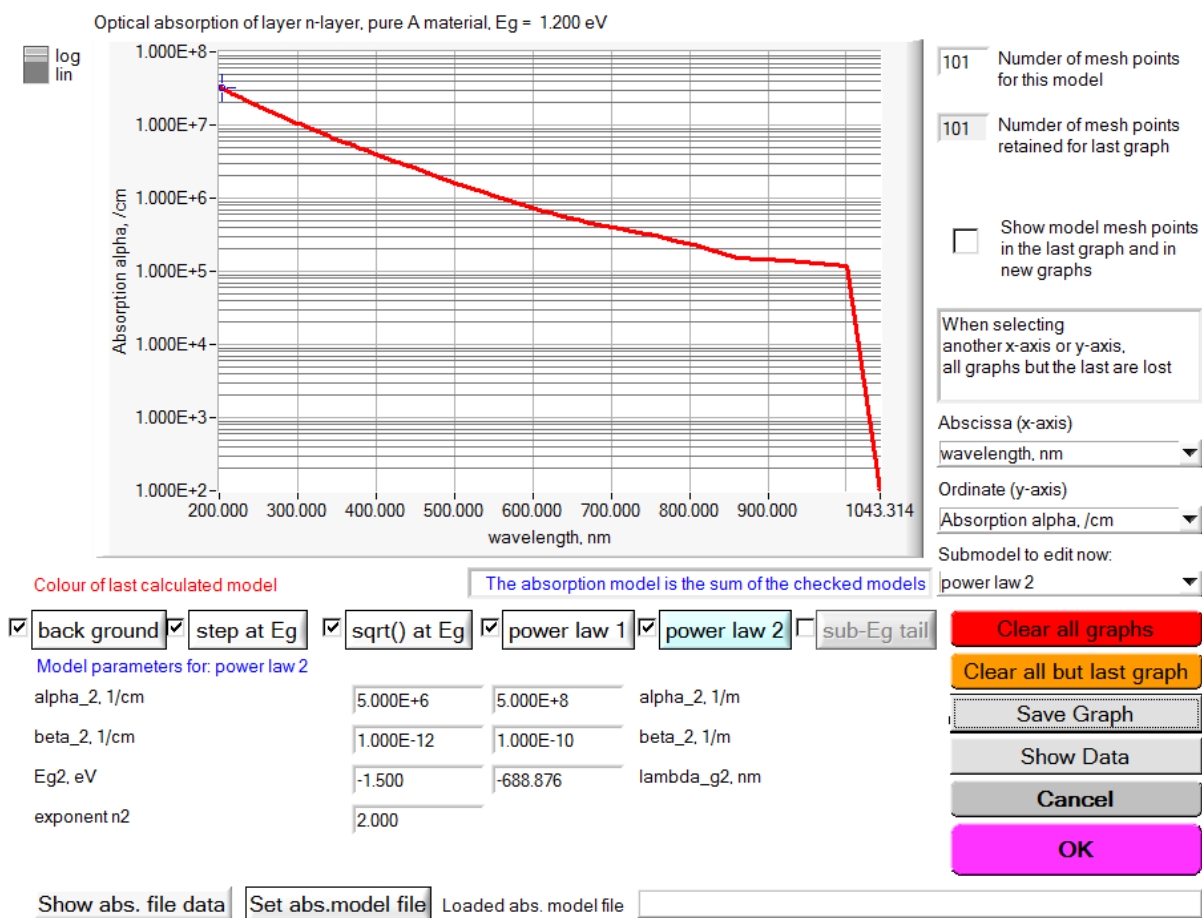
Figure 3.16   The Absorption Model Panel of SCAPS ≥ 3.3.07.

Description of the Absorption Model Panel:

- The title displays the layer name, here 'n-layer', and the actual value of the band gap E_g, here 1.200 eV.
- The 6 SCAPS absorption models 'back ground'… to 'sub-Eg tail' are displayed in a line below the graph. Each sub-model can be clicked on or off. The button with the sub-model name is only active when the sub-model is checked; upon clicking such button (here the 'power2' button was clicked), this button is highlighted in light blue, and the sub-model parameters (α_2, β_2, E_{g2}, n_2) are displayed and can be edited. Some parameters have an 'alternative form'; this can be merely another unit (α_2 and β_2, in 1/cm or 1/m), or another informative form of the parameter (E_{g2} or λ_{g2}). (These are 'alternative facts' that are true ☺).

- Upon changing a parameter value (or value of an alternative parameter), a new $\alpha(\lambda)$ or $\alpha(h\nu)$ curve is calculated and added to the graph. When it starts to look too messy, you can clear all graphs or all but the last calculated one.

- You can select two abscissa (horizontal axis) variables (wavelength λ or photon energy $h\nu$) and two ordinate (vertical axis) values (just α, either in 1/cm or in 1/m). When change the abscissa or ordinate selection, all graphs but the last graph are lost.

- You can also show the $\alpha(\lambda)$ or $\alpha(h\nu)$ data contained in a SCAPS absorption file; once a file selected, you can show or hide the file data with a toggle button. Remark: showing file data does not change the absorption mode from 'from model' to 'from file'! This mode remains 'from model', but at least you can compare your model try-outs with real data.

- Tip: this is also a great way to look to an absorption file when setting up a SCAPS model: immediately you see what it is like!

- The number of mesh points (λ values or $h\nu$ values) in the top right part of the panel of Figure 3.16, is only relevant for this panel. In calculations, SCAPS decides the λ-values at which to evaluate $\alpha(\lambda)$, the generation $G(x, \lambda)$,… based on the calculation settings ordered by the user: e.g. λ values in the spectrum file, λ values at which $QE(\lambda)$ are ordered,…

- Of course all other familiar SCAPS facilities are there: linear/logarithmic view with only one click, saving the graph in some graphical format (great for presentations and reports – usually the graphical quality is not good enough for decent publications), showing the data (and copy/paste them e.g. in Excel).

3.5.6.4 The SCAPS grading algorithm for optical absorption constant α

The grading of the optical absorption (no matter whether is has been defined from a model or from a file) needs a dedicated interpolation algorithm [3] to determine a grading dependent $\alpha(\lambda,y(x))$. To interpolate the optical absorption constant $\alpha(\lambda,y)$ for some composition between the pure material A with composition $y = 0$ and absorption $\alpha_A(\lambda)$, and the pure material B with composition $y = 1$ and absorption $\alpha_B(\lambda)$, SCAPS uses the following algorithm. First, determine the cut-off wavelengths λ_{gA} and λ_{gB} of the pure materials, and a characteristic wavelength λ_{0A} and λ_{0B} in the near UV wavelength range. Usually, the $\alpha(\lambda)$ curves have a maximum (peak) in the near UV, if not one can take an arbitrary value for λ_{0A} and λ_{0B}. Then determine the cut-off wavelength λ_g of the compound with composition y: $\lambda_g = 1240\text{nm.eV}/E_g(y)$, and the UV peak wavelength λ_0 from linear interpolation between λ_{0A} and λ_{0B}. A first estimation for $\alpha(\lambda)$ is then obtained by evaluation α_A at a wavelength λ_A given by

$$\lambda_A = \frac{\lambda_{gA}(\lambda - \lambda_0) + \lambda_{0A}(\lambda_g - \lambda)}{\lambda_g - \lambda_0} \tag{8}$$

A second estimation is found by evaluation α_B at a wavelength λ_B found in a way similar to Eq. (8). Then take a weighted logarithmic average between the two estimations:

$$\log \alpha = (1 - y)\log\left[\alpha_A(\lambda_A)\right] + y\log\left[\alpha_B(\lambda_B)\right] \tag{9}$$

The merits of this interpolation algorithm are discussed in [3].
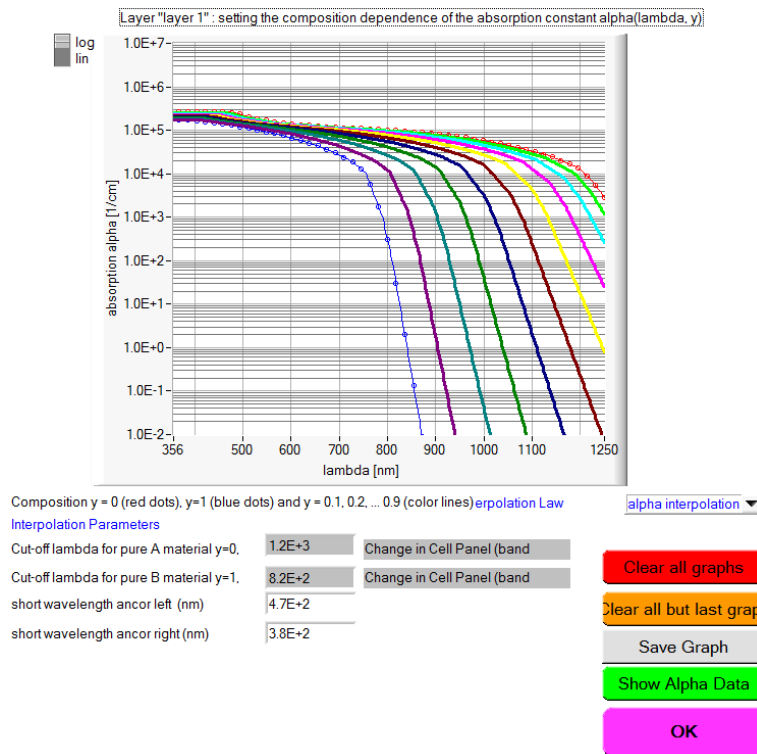
Figure 3.17 The optical absorption constant $\alpha(\lambda, y)$ of a compound material with composition y, thus $A_{1-y}B_y$, calculated by SCAPS with a dedicated interpolation algorithm. Here, the pure A material is CuInSe$_2$, and the pure B material CuInS$_2$. The data can be obtained in tabular form by clicking Show Alpha Data (the green button)

3.5.7 The actual position dependent grading results

The graded properties get their position dependence directly (N_D (x), $N_A(x)$, $N_t(x)$ when these options are selected), or indirectly via the x-dependence of the composition y: $P[y(x)]$, all other cases. The x-dependent values that are finally used in the calculations can be obtained from the Cell Definition Panel, but only after at least a calculation is done (a working point calculation is sufficient, unclick I-V, C-V, C-f, and QE). Hereto click one of the green save, show or graph view buttons on the Cell definition panel. The Graph View button was added in SCAPS3.3.1, March 2015, finally ☺. All grading data: [Save] [Show] [Graph View].

Upon clicking of the View Grading button, the View Grading Panel of Figure 3.18 is opened. In SCAPS ≥ 3.3.07, also an 'optical absorption button' is added it. By clicking it, a view of the absorption constants $\alpha(\lambda)$ or $\alpha(h\nu)$ of (the middle of) all layers together is shown in one graph. The α values apply to the middle of each layer (important if the layers were graded, otherwise not). Also, we added a Show Data button at the bottom right of this panel. Is works as the 'show buttons' in all other SCAPS panels. In particular, you can copy/paste from this SCAPS output window into your favourite program for further processing of the data (Excel, Origin, whatever).
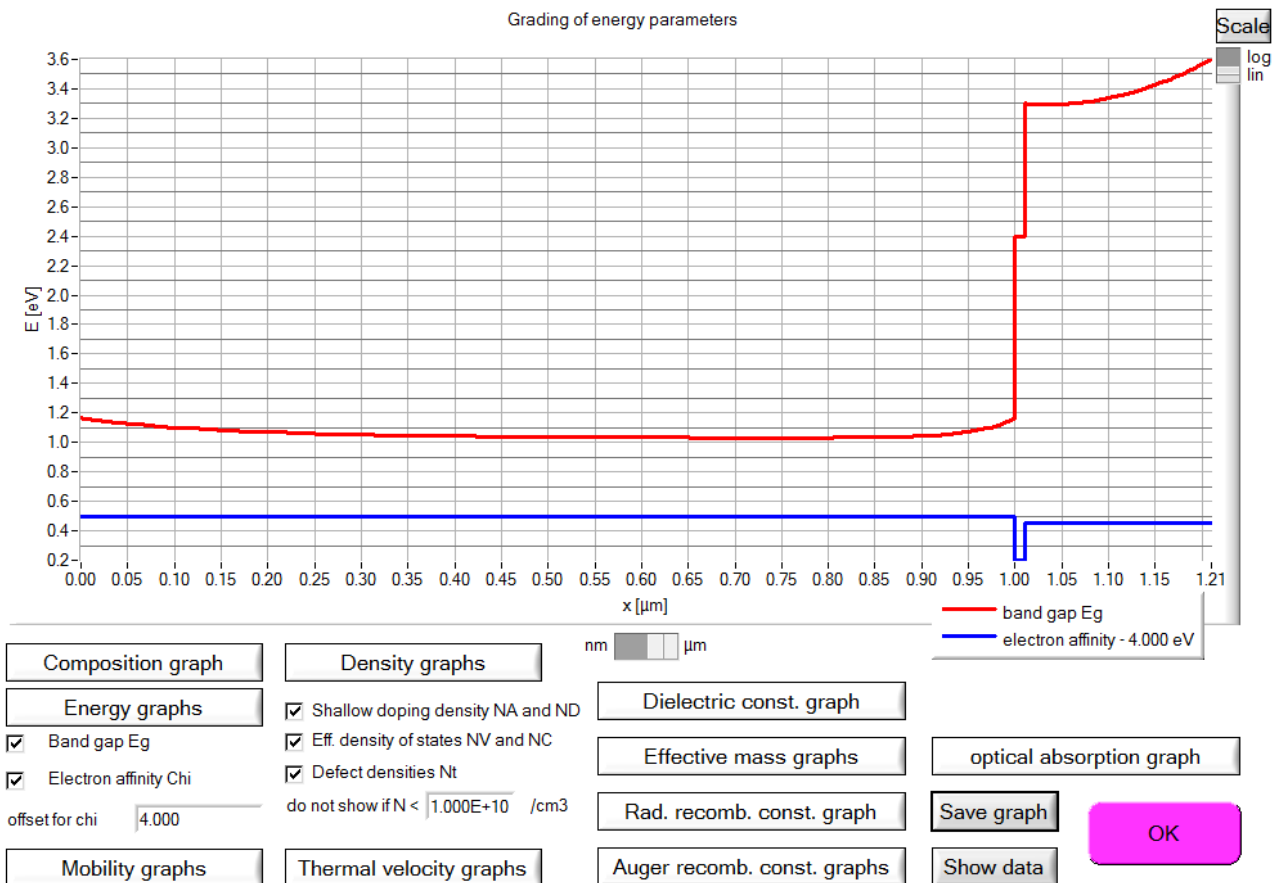
Figure 3.18 The View Grading Panel allows a quick overview of all parameters that can be graded. Parameters that one would possibly like to view in one graph are grouped together.

When using the green save button (not the show-button) you also get the values of all α(λ,x)-values, where the λ-values are the wavelengths present in the spectrum you used to perform the simulation and the x-values the mesh-points. Combining this facility with a uniform y-graded layer allows you to construct personalized absorption files as well ☺.

If you want to have a graphical view of the graded data, the best way to go is (was?) to use the recorder function which is introduced in Chapter 8. Then you have access to more variables and saving and showing options as well.

3.5.8 A materials approach

All parameters of a layer can be defined separately, but they can also be loaded from and saved to a '.material' file, using the appropriate buttons on the layerpanel. [Load Material] [Save Material]

A material consists of all layer parameters excepting the layer name, the layer thickness, the doping density and the defect properties.

3.5.8.1 Saving materials

Materials can be saved. This results in an ASCII-file with extension '*.material'. There are four different options to save a material: both materials, pure A, pure B and pure Y. The options pure Y and both materials are not available when the composition grading is set to pure A or pure B. In that case only one option (pure A or pure B) is available.
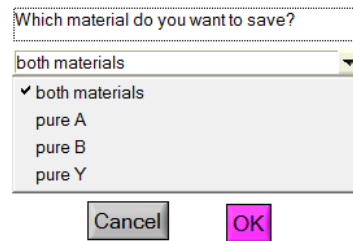
Figure 3.19   Available options for saving materials

Saving a material using the pure A/B option only saves the parameter values of the A/B-material of the layer. All parameter grading is set to uniform.

Saving a material using the both materials option saves both parameters of the A/B-material of the layer together with the appropriate parameter grading parameters.

When saving a material using the pure Y option you should give a fraction y which defines the material to be saved as $A_{1-y}B_y$. The appropriate parameter values for this material are then calculated according to the values for the A/B-material of the layer and saved as a uniform material (similar to pure A/B). The absorption for this material is also calculated as an interpolation of the absorption for A and B and is saved in an absorption file. This file is automatically added to your absorption folder and gets the same name as your material-file extended with '.abs'.

3.5.8.2   Loading materials

Materials which have been saved, can of course be loaded as well. Three options are available: pure A, pure B and both materials. When the composition grading of the layer was saved to pure A/B however, only the pure A/B option remains available for loading.

Loading a material using the both materials option sets all material parameters of the A- and B-side of the layer, including the parameter grading. When loading a material which has been saved as a pure A/B/Y material, the parameters of the A- and B-side of the layer will be identical and all parameter grading will be set to uniform.

Loading a material using the pure A/B option only sets the parameters of the A-/B-side material of the layer. If the parameter grading was uniform it will be set to linear.

If the materials file contains information on thickness, on the shallow doping densities or on defects, you can load this information or not: select or unselect the appropriate check box in the Load Material Panel (Figure 3.20). As a materials file can contain file names (optical absorption files, grading files), you should make sure that these files are indeed present in the appropriate SCAPS directory.

Material files can also be varied in a batch calculation (see Chapter 7).
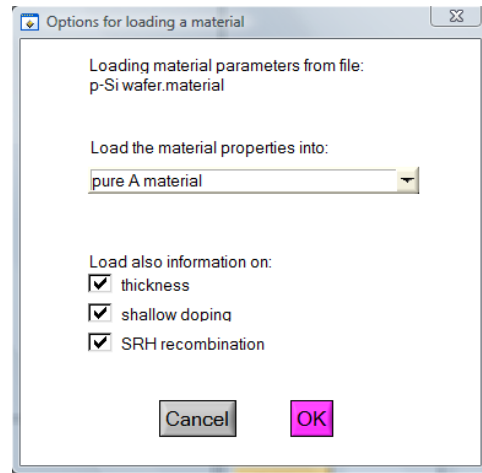
Figure 3.20 Available options for loading materials

3.5.9 A frequently asked question (FAQ) about grading

Sometimes we think of a desired or supposed band diagram, with a sloped (non horizontal) conduction band and/or valence band. The model input parameters are the grading of the electron affinity $\chi(x)$ and of the band gap $E_g(x)$. There is sometimes misunderstanding about the relation between the $\chi(x)$ and $E_g(x)$ input, and the band bending $E_C(x)$ and $E_V(x)$ they cause: the result depends on the doping density and on the voltage and illumination conditions. The rules of thumb are given below and illustrated in Figure 3.21 and Figure 3.22 below:

- In equilibrium, in a neutral, *p*-type region (e.g. the absorber bulk in CIGS), $E_F = E_{Fp}$ = horizontal (equilibrium), E_V is a fixed amount $kT \times \ln(N_V/N_A)$ under it (depending on the doping density N_A; supposing that both N_A and N_V are uniform and not graded): thus the valence band will be horizontal, regardless the band gap and/or electron affinity grading. The conduction band E_C is then placed at a distance E_g above E_V, thus: $E_C(x) = E_V + E_g(x)$, and will thus be sloped when $E_g(x)$ is graded. You will not see any effect on electron affinity grading.

- in equilibrium, in an *n*-type region, $E_F = E_{Fn}$ = horizontal. The conduction band E_C is placed at a fixed distance $kT \times \ln(N_C/N_D)$ above it (this is a constant when both N_D and N_C are uniform and not graded). Thus the conduction band will be horizontal, regardless the electron affinity grading. The valence band will then be placed a distance E_g under E_C, thus: $E_V(x) = E_C - E_g(x)$ and will be sloped when $E_g(x)$ is graded.

In all other circumstances (applied V, illumination, in a depletion layer, grading of the doping densities $N_D(x)$, $N_A(x)$ or of the densities of states $N_C(x)$, $N_V(x)$) there are no simple rules of thumb, it gets too complicated. Trust e.g. SCAPS for it. But always check SCAPS (that is, check what you have input) by always looking to the equilibrium band diagram and applying the rules of thumb above where possible (that is, in the flat-band regions).
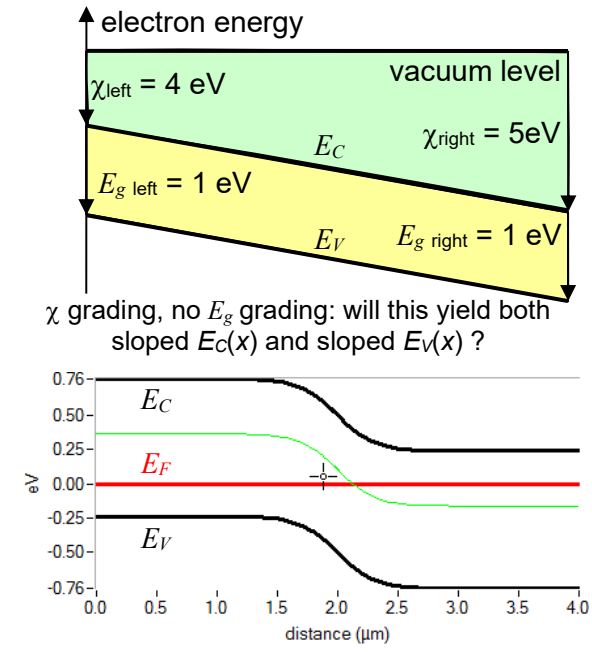
Figure 3.21   Developing 'intuition' about grading. Grading schemes of electron affinity $\chi(x)$ and of band gap $E_g(x)$: $\chi(x)$ is graded, but E_g is not. What is suggested by the naïve sketch (top) and what really happens in a pn junction (SCAPS calculation, bottom).
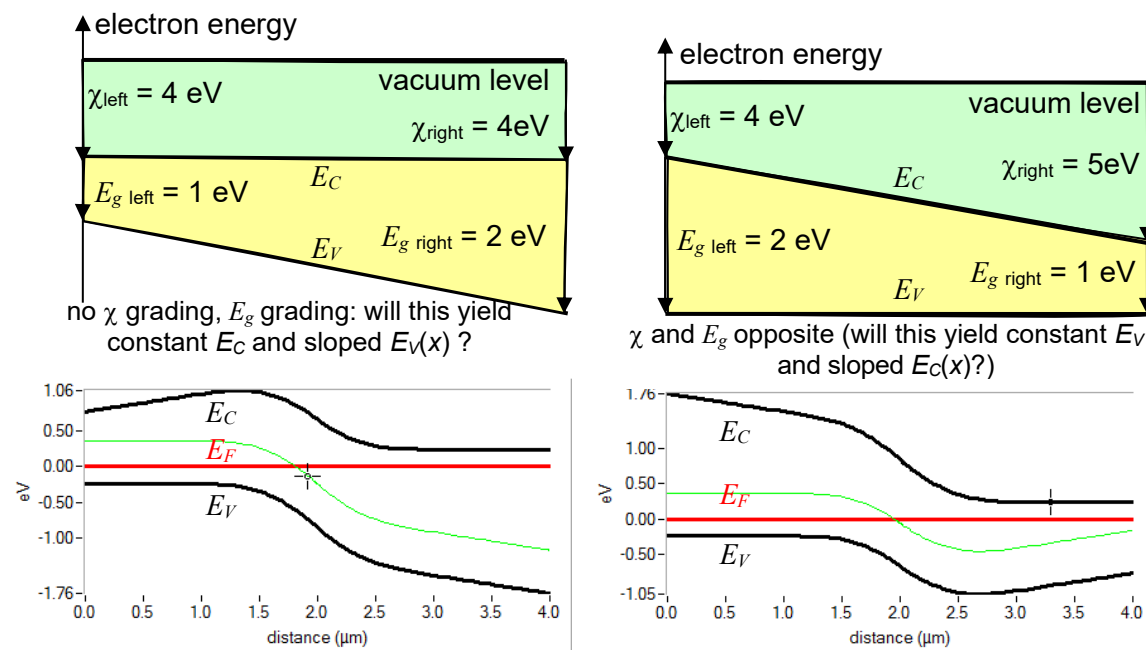
Figure 3.22   [more…] Developing 'intuition' about grading. Grading schemes of electron affinity $\chi(x)$ and of band gap $E_g(x)$: [left] no χ grading but an $E_g(x)$ grading and [right] both $\chi(x)$ and $E_g(x)$ are graded, but their sum $\chi(x) + E_g(x)$ = 6 eV = constant. What is suggested by the naïve sketch (top) and what really happens in a pn junction (SCAPS calculation, bottom).

3.6 Defects and recombination

In a diode, current is converted from hole current at the p-contact to electron current at the n-contact. This means that somewhere in the diode recombination **MUST** take place, even in the most ideal device. So the

user **MUST** specify recombination somewhere, **at least at one** place (in a layer, at a contact or at an interface). If (s)he does not do so, a convergence failure will result in non-equilibrium conditions (non zero voltage, and/or illumination).

In the bulk of a semiconductor layer, three different kinds of recombination processes can be introduced: through defects, radiative and Auger.

3.6.1 Adding defects

Up to seven defects can be introduced in a semiconductor layer. The parameters governing each defect can be edited by clicking the appropriate Add/Edit-button, Figure 3.23, which opens the 'defect properties panel', Figure 3.24. Also, you can right-click on one of the 'defect summary text boxes'; a panel then opens where you can remove, duplicate or add a defect, much the same as it was with removing, duplicating and adding a layer (Section 3.5.1 and Figure 3.7, page 10); of course, 'splitting a defect' is not available as it has no meaning.
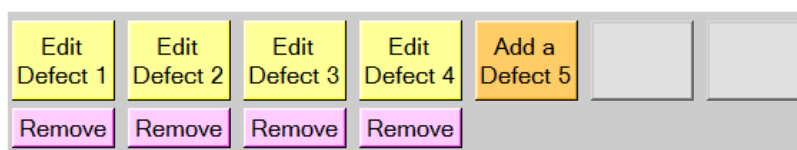
Figure 3.23   Adding, editing and removing defects.

Figure 3.24   The defect properties panel

3.6.2 Multivalent defects

The most common defects in semiconductors are either donor or acceptor defects having two possible charge states. However, there exist defects with more than two different charge states (multivalent defects). In SCAPS you can add up to four defect levels (= five charge states) for each defect. The charge present on a defect can be varied from -3 to +3 elementary charges. Because the most common multivalent defects are double donors (charge states 0, +1 and +2), double acceptors (charge states 0, -1 and -2) and amphoteric

defects (charge states 0, +1 and -1), special facilities for them are provided in SCAPS. The charge states connected to a defect can be set by the 'defect type button'

If the defect is not multivalent (donor-acceptor-neutral), all defect parameters can be set on the defect density panel, otherwise clicking the defect type button will result in opening the 'Multiple level defects properties panel', Figure 3.25.

The algorithms used by SCAPS to calculate the recombination through defects are explained in [2].

Figure 3.25 'Multiple level defects properties panel'

3.6.2.2 A neutral defect

The "**neutral**" defect is an idealization of a defect which contributes to the Shockley-Read-Hall recombination but does not contribute to the space charge. In the case of a "neutral" defect, only the product of σ and N_t affects the dc and ac solutions, through the carrier lifetimes; τ_n e.g. is given by $1/(\sigma_n . N_t . v_{th})$: in this case, the defect centers contribute to recombination but not to the space charge. This type of defect can be chosen if one wants to specify electron and hole lifetimes without specifying a defect density which affects the space charge.

In other words: a neutral defect does not exist in reality, it is an idealization to help you to create a model step by step.

3.6.2.3 More than two charge states

When choosing the defect type as double donor, double acceptor, amphoteric or custom defined multilevel the user is redirected to the multiple level defects properties panel (see Figure 3.25). Some of the properties which were already available on the defect properties panel are available here as well, together with the properties which are specific for a multivalent defect. Starting from this panel the user can edit the defect and its levels.

Only when the defect type *custom defined multilevel* has been selected the user can add up to four different levels, otherwise the number of levels is restricted to two and the charge types are predefined. The 'insert' buttons are used to add a specific level. In order to remove a level the user has to click on a level with the right button of the mouse.

The charge type and the energy of a level can immediately be accessed on the *multiple level defects properties panel*. The charge types MUST be introduced in an ascending way (level 1 the most positive charge), or all levels must be neutral. In most cases the most positive level will be situated the closest to the conduction band, a warning is given with the option for the user to define the defect differently, but the user is not obliged to comply with the warning.

The other defect properties of a level can be edited by double-clicking on the text-box belonging to each level. This opens a panel similar to the defect density panel, however the properties which are not level specific but general to the entire defect are dimmed and can't be edited. The only properties which can be edited are the capture cross sections (for electrons and holes), the energy level and the IPV-properties (§3.6.4).

Several features of the energetic distribution can also be accessed on the multiple level defects properties panel. The *energetic distribution*, the *reference for defect energy levels* and the *characteristic energy* are the same for all levels of one defect. More info about these properties can be found in§3.6.3. If the checkbox *Use Correlation Energy* is checked, the energy difference between level 2 and level 1 is displayed in the window next to the checkbox. When the user edits this correlation energy the energy level of level 2 will be changed accordingly with respect to level 1.

The degeneracy factors (see [2]) take as a default value Eq. (10), with H the total number of charge states.

$$g_s = C_H^s = \frac{H!}{s!(H-s)!} \tag{10}$$

However, their values can be set to one as well by clicking the appropriate checkbox.

When the user has finished editing the properties of the levels the *multiple level defects properties panel* can be quitted by the *return to the defect definition panel*-button. Then the *defect properties panel* is again displayed, but the level specific properties are no longer visible. It is a kind of survey of the defect. Now the changed to the defect can be saved using the *add/accept*-button or ignored using the *cancel*-button.

3.6.3 Energetic distribution of defect levels

Next to a discrete defect energy level, also defect whose levels display a distribution in the band gap can be modeled in SCAPS. There are five different options, listed in Table 3.2.

Table 3.2   Overview of the energetic distributions of defects available. E_t is the energy level ('trap level'), E_c is the characteristic energy (not the conduction band edge energy here!). The defect density $N_t(E)$ is given in cm^{-3}/eV (except for *single*, where it is given in cm^{-3}). N_{peak} is the energetic density at the peak of the distribution, in cm^{-3}/eV. N_{tot} is the total density, integrated over all energies in (cm^{-3}). The parameters E_t, E_C, and N_{tot} or N_{peak}, and the shape of the energetic distribution ('single', or 'uniform'…) are set in the Defect Properties Panel for each defect. The width parameters w_G and w_t are set for all defects together in the Numerical Panel.

	Range	$N_t(E) = \dots$	$N_{tot}(N_{peak})$	remarks
Single	$[E_t; E_t]$	$N_{tot} \times \delta_{E_t}$		
Uniform	$\left[E_t - \dfrac{E_c}{2}; E_t + \dfrac{E_c}{2}\right]$	N_{peak}	$N_{tot} = E_C N_{peak}$	
Gauss	$\left[E_t - \dfrac{w_G}{2}E_c; E_t + \dfrac{w_G}{2}E_c\right]$	$N_{peak} \times \exp\left[-\left(\dfrac{E - E_t}{E_c}\right)^2\right]$	$N_{tot} = E_C N_{peak}$	(1)
CB tail	$[E_t - w_t E_c; E_t]$	$N_{peak} \times \exp\left(\dfrac{E - E_t}{E_c}\right)$	$N_{tot} = E_C N_{peak}$	(2)

VB tail $\quad\left[E_t; E_t + w_t E_c\right]$ $\qquad N_{\text{peak}} \times \exp\left(-\dfrac{E - E_t}{E_c}\right)$ $\qquad N_{\text{tot}} = E_C N_{\text{peak}} \sqrt{\pi}$ $\qquad$ (3)

Remarks:

(1) w_G can be defined in the numerical panel; default value: w_G = 6.0; see Figure 3.26; G stands for *Gauß*.

(2) w_t can be defined on the numerical panel; default value: w_t = 7.0; see Figure 3.26; t stands for *tail*.

Whenever the defect energy distribution is not single, this distribution is discretized as being a specified number of single defect levels. This number can be set on the numerical panel, its default value is 7.
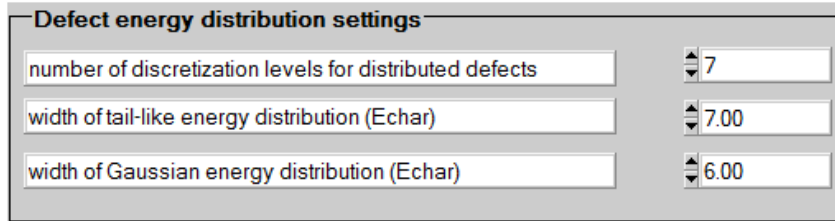
Figure 3.26 Defect settings on the numerical panel with their default values. Top: the number of discretization levels, bottom: w_t and w_G.

The energy level has to be defined with respect to a reference energy level. This can be set on the defect density panel or on the multiple level defects properties panel. The available choices are: above E_V (above the valence band level); below E_C (below the conduction band level) and above E_i (above the intrinsic level

$$E_i = E_V + \frac{E_g}{2} + \frac{k_B T}{2} \ln\left(\frac{N_V}{N_C}\right)).$$

3.6.4 Impurity photovoltaic effect (IPV)

SCAPS is able to simulate the IPV-effect. Its parameters can be set on the defect density panel. The algorithms and an example is given in [8, 9]. More examples are found in articles referring to this article, e.g. [10]. An example '*.def'-file is provided with the SCAPS installation with ample comments.

3.6.5 Short overview of defect values

When all defect-parameters are set, an overview is given in the layer definition panel, Figure 3.27.
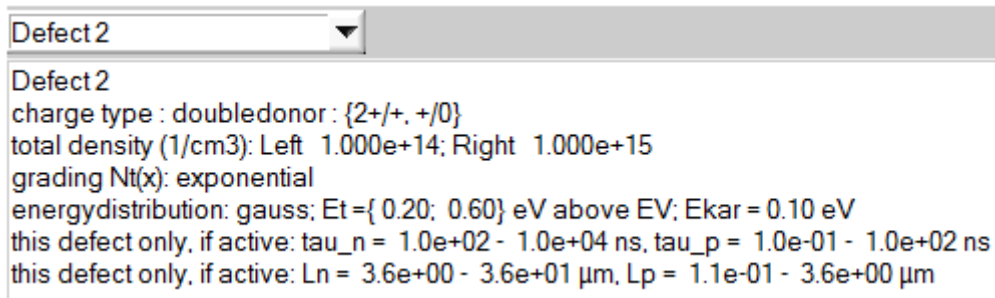
Figure 3.27 Overview of the defect parameters

Next to the charge type, the defect density (and its spatial and energetic distribution) also an overview of the carrier lifetime and diffusion length is given. These are calculated (Eqs. (11) and (12)) only at the left and right side of the layer for every energy level. Only its maximum and minimum value are listed.

$$\tau = \frac{1}{\sigma v_{th} N_t} \qquad (11)$$

$$L = \sqrt{D\tau}$$ (12)

IT SHOULD BE STRONGLY EMPHASIZED THAT THESE VALUES ARE ONLY CALCULATED TO GIVE THE USER A ROUGH IDEA ABOUT THE RECOMBINATION OF THIS DEFECT, NONE OF THESE VALUES ARE USED IN THE ACTUAL SIMULATIONS! MORE ADVANCED ALGORITHMS ARE USED THERE, see e.g. [2].

3.6.6 Radiative and Auger recombination

It is possible to introduce radiative and Auger (band-to-band) recombination in SCAPS, according to Eqs. (13) and (14).

$$U_{\text{radiative}} = K\left(np - n_i^2\right)$$ (13)

$$U_{\text{Auger}} = \left(c_n^A n + c_p^A p\right)\left(np - n_i^2\right)$$ (14)

The parameters K, c_n^A and c_p^A can be set on the layer panel. SCAPS does not give a hint for a plausible K value ☹; for Si you can take $K_{\text{Si}} = 1.8 \times 10^{-15}$ cm^{-3}s^{-1}, for GaAs $K_{\text{GaAs}} = 7.2 \times 10^{-10}$ cm^{-3}s^{-1}. In CIGS the recombination is most probably dominated by defects, you can take $K = 0$, unless you want to simulate electroluminescence. In Si appropriate values for the Auger constants are: $c_n^A \approx c_p^A \approx < 3 \times 10^{-31}$ cm^{-6}s^{-1} .

Band to band recombination			
Radiative recombination coefficient (cm³/s)	1.000E-5	1.000E-5	uniform ▾
Auger electron capture coefficient (cm^6/s)	1.000E-25	1.000E-25	uniform ▾
Auger hole capture coefficient (cm^6/s)	1.000E-25	1.000E-25	uniform ▾

Figure 3.28 Band to band recombination parameters. From top to bottom: K, c_n^A and c_p^A.

The contributions of the different recombination processes to the current can be assessed on the IV-panel (see §6.4.3) and in the recorder facility (see Chapter 8). When you are sure that all direct band-to-band recombination is radiative, and that all emitted photons can get out of the cell to your measurement set-up, you can interpret $J_{\text{radiative}}$ as the electroluminescence signal. However it is not labeled as such in SCAPS.

3.7 Radiative recombination and the Shockley-Queisser limit for the efficiency parameters

The default value for the radiative recombination constant K is $K = 0$. However, there is a physical limitation to radiative recombination (or 'band-to-band recombination): it cannot be lower than the so-called Shockley-Queisser limit. When setting the default values for band-to-band recombination ($K = 0$) and Auger recombination ($c_n^A = c_p^A = 0$), and when setting very, very low values (unrealistically low) for the other recombination mechanisms (contacts, interfaces, SRH recombination in the semiconductor layers), the simulated efficiency parameters η and V_{oc} could exceed their Shockley-Queisser limit. To prevent this, the user can force the band-to-band recombination to its minimal value imposed by the Shockley-Queisser theory (instead of zero), and this will prevent a 'Shockley-Queisser accident' in the simulation. You can set the S-Q option in the Numerical Panel (accessible with the orange button in the Cell Definition Panel), at the bottom right corner. There are three options: *never*, *if needed* and *always*. The default is *never*, as it was in traditional SCAPS. The new S-Q options are available in SCAPS ≥ 3.3.11 (September 2023). For a detailed explanation, the user is referred to an Application Note available from our installation website.

3.8 Metastable defect transitions

In the chalcopyrite material system often metastable behaviour of the samples is observed. This is possibly due to the presence of defects which undergo transitions which are accompanied by large lattice relaxations and thermal activation over energy barriers. SCAPS is able to simulate this behaviour, the algorithms which

are implemented and an example can be found in [11]. An introduction to metastable defects in SCAPS was given in [11'].

3.8.1 Principles

Metastable effects take place on a long timescale (which can vary a lot, depending on the actual conditions). SCAPS does not solve transient problems. Hence, the following philosophy has been followed.

A metastable defect is considered to exist of two different configurations, a donor and an acceptor configuration, which are separated by energy barriers associated with lattice relaxations. Each separate configuration behaves as a conventional defect, which can have a multivalent character. The transition between the different configurations requires a simultaneous capture or emission of two free carriers together with the thermal activation of an energy barrier. Hence, at lower temperatures, this transition will not occur. Similarly the transition will not be observed in an admittance measurement when the change of the quasi-Fermi levels occurs at higher frequencies.

In a typical experiment, the studied sample is brought to a well defined metastable state, i.e. a distribution over the donor and acceptor configuration, by applying certain initial voltage/illumination condition at elevated temperatures for a sufficiently long time. Afterwards the sample is cooled down in order to inhibit further changes in the metastable state of the sample and measurements are performed.

In order to simulate this, first the distribution over the acceptor and donor configuration of the defect is calculated under initial voltage/illumination conditions, see §4.4. This distribution, together with the total defect density is used to calculate the defect density belonging to the different configurations of the metastable defect. These defect densities are kept constant and are used to perform further simulations under measurement voltage/illumination conditions.

3.8.2 Introduction of a metastable transition

A metastable defect is assumed to have two different configurations (acceptor and donor). Each of these configurations has to be introduced as if it were a regular (multivalent) defect. Metastable properties can only be set for defects with single energy levels, not for defects that occupy a band of energies. If at least two defects are present in a layer which have compatible charge states, they can be set to be the two configurations of a metastable defect by checking one of the 'metastable transition'-boxes on the layer panel, Figure 3.29. By clicking the 'edit'-button, the 'Metastable Defect Definition Panel' (Figure 3.30) opens, where all parameters governing the metastable transition between the two configurations can be set/edited.
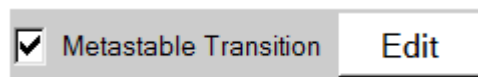
Figure 3.29   Allowing a metastable transition.

Figure 3.30   The Metastable Defect Definition Panel

Its main components are discussed below:

Charge states: This is a list which gives all possible combinations of charge states available which can be transformed into one another through a double capture/emission process. If none are available, you will not be able to access the Metastable Defect Definition Panel. If more than one option is available, you can select the one you want.

Activation Energies: The transition between the two configurations of the metastable defect can proceed through four processes: electron capture (EC), hole emission (HE), electron emission (EE) and hole capture (HC). Each of these processes has its own activation energy which can be set. The Fermi level position where the two configurations are equally stable under thermal equilibrium conditions is called the transition energy (TR) and can also be set. These four activation energies together with the transition energy are not independent as they have to obey to the principle of detailed balance. Only three of them can be chosen independently, the two remaining are then calculated by the program. You can decide which energies you want to provide in the '*Activation Energies to be set*'-list. When a band gap grading is present, the calculated energies can vary throughout the layer. The values which are displayed in the user interface are then those which are valid for the right side of the layer. The transition energy can be set with respect to the valence band, conduction band or intrinsic energy level in a similar way as a regular defect level.

Metastability type and electron/hole processes: The double capture and emission processes can be obtained in different ways. The electron capture process e.g. can consist out of two simultaneous single electron capture processes (*Double electron capture*) or a single electron capture together with a simultaneous hole emission process (*Electron capture + Hole emission*). This has an influence on the details of the detailed balance calculation and needs to be set. These properties can also automatically be set to the

standard value for the common V_{Se}-complex and In_{Cu}-complex in the CIGS material by selecting respectively '*Se Vacancy*' or '*DX centre*' in the *Metastability type*-list.

Capture cross sections: You cannot change the four capture cross sections in the Metastable Defect Definition Panel, they are shown there for your information only. Instead, you should set these in the Multilevel Defect Properties Panel of Figure 3.25. To know which of the σ_n and σ_p values of the defects (possibly multivalent) constituting the metastable defect, you should (alas ☹) understand some of the details of the metastable transition mechanisms. To help the user a bit, this information is also displayed on the Metastable Defect Definition Panel from SCAPS 3.3.07 of november 2018 onwards, see Figure 3.30.

Defect density: The defect density of the metastable defect can be set in a similar way as for a regular defect. Pay attention! When a defect is set to be one of the configurations of a metastable defect density, the defect density which you can set on the *defect properties panel* will be ignored. Instead at every meshpoint the relevant defect density of this configuration will be calculated from the total defect density of the metastable defect.

Initial donor occupation guess: In order to start the iteration process an initial guess has to be made for the fraction of the metastable states that are in the donor configuration. This fraction is the same for all meshpoints.

3.8.3 Help, the buttons to introduce/edit metastable properties are not available!
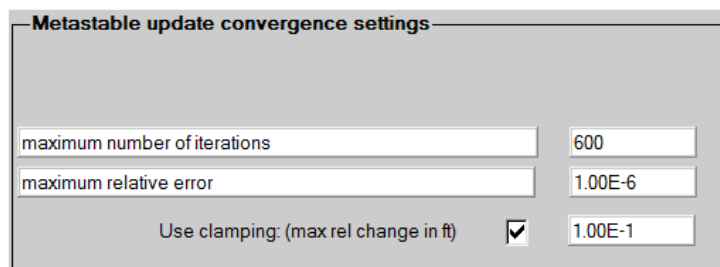
One of the aims of SCAPS is to keep the user interface very intuitive and easy to learn. Since the first release, a multitude of extra facilities have been added to the program. These extensions increase the number of situations which can be simulated, but also complicate the learning process of a user making his/her first simulations.

The simulation of metastable defects is considered to be 'advanced' and should not be one of the worries of a rookie in the bright world of numerical simulations with SCAPS. As a result, most of the buttons governing metastabilities are only accessible/visible when the user really expresses her/his intention to use them:

- The 'Metastable Defect Definition Panel' can only be accessed if at least two defects are present in the layer AND at least one of the charge states of one defect can be transformed into one of the charge states of a second defect through a double capture process.

- The 'Initial State Workingpoint Panel' can only be accessed if at least one metastable defect is present in the definition file which is currently loaded.

3.8.4 Numerical settings

The calculation of the occupation of the different metastable configurations is done in an iterative way. The maximum number of iterations and the minimum remaining error can be set on the numerical panel, see Figure 3.31.

Figure 3.31   Numerical settings for metastable simulations

When the calculation is numerically unstable it can be stabilized by using a clamping factor. This process is described in [11]. This factor should be chosen between 0 and 1, 1 corresponds to no clamping. The

smaller this factor the more stable, but also the slower the calculation gets. Values below 0.001 are usually very slow.

3.9 Interfaces

Between any two semiconductor layers an interface can be defined. The main algorithms used in SCAPS are discussed in [12].

The model which is implemented for interface transport in SCAPS is thermionic emission. The thermal velocity of the interface transport equals the smallest thermal velocity of the two neighboring layers. The use of this model implies that there will always be a (small) step in the quasi-Fermi level energy values at an interface, even when there are no band offsets.

Just as in a bulk layer recombination centers can be present at an interface. The definition of interface defects is very similar to bulk defects. However, there are only three possible defects and there charge type cannot be multivalent. Recombination in interface states is modeled by the Pauwels-Vanhoutte theory [13], which is an extension of the Shockley-Read-Hall theory.

Two tunneling processes are implemented which involve interfaces: intraband tunneling and tunneling to interface defects, see §3.10.

3.10 Tunnelling

SCAPS treats (some) tunnelling mechanisms in a semi-classical way: band to band tunnelling, intraband tunnelling, tunnelling to interface defects and tunnelling to contacts. The main algorithms are discussed in [4], a more elaborate version is available in [14] (in Dutch, on request). PAY ATTENTION: Tunnelling is only taken into account in the solution of the dc-problem, not in the ac-problem. Hence only indirect tunnel influences on the admittance (through the setting of the dc-state of the sample) are simulated!

Band-to-band tunnelling is the tunnelling between conduction and valence band states, see Figure 3.32. Intraband tunnelling is the tunnelling between states both in the same (conduction/valence) band, see Figure 3.33. The valence and conduction band consists in (almost) every structure out of bulk layers and interfaces. Band-to-band tunnelling and intraband tunnelling can be allowed/forbidden for every layer/interface separately by checking the appropriate checkbox on the layer/interface panel.

Next to band-to-band and intraband tunnelling, also tunnelling between the bands and an interface defect, or the bands and a metal contact is possible, see Figure 3.34. These processes can be set for each interface defect separately on the interface defect properties panel and for each contact on the contact definition panel. Whether or not these tunnelling processes are allowed does not depend on the layer/interface settings. E.g. if tunnelling to the left contact is allowed on the contact properties panel, tunnelling between this contact and all layer/interfaces will be allowed even if the 'allow tunnelling'-checkbox is unchecked in one or more layers/interfaces.

Band-to-band tunneling at reverse bias   Band-to-band tunneling at forward bias
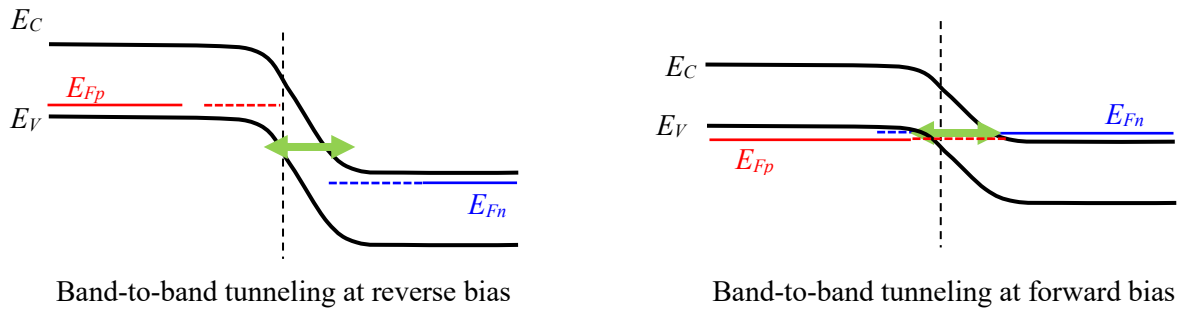
electron mobility (cm²/Vs)		5.000E+1
hole mobility (cm²/Vs)		5.000E+1
☑ Allow Tunneling	effective mass of electrons	3.000E-1
	effective mass of holes	8.000E-1
no ND grading (uniform)		
shallow uniform donor density ND (1/cm3)		1.000E+1

Relevant part of the Layer Properties

Figure 3.32   Band-to-band tunnel mechanisms at reverse and at forward bias. Note that band-to-band tunneling at forward bias (Esaki tunnel diode) is only possible with very heavily doped, degenerated semiconductors, where the Fermi level is inside the conduction and/or valence band; these heavy doping phenomena (Fermi statistics, band gap narrowing…) are not implemented in SCAPS. At the bottom, the check box and m_{eff}^{*} values to be enabled in the Layers Properties Panel are shown.

intra-band tunneling in the conduction band

intra-band tunneling in the valence band
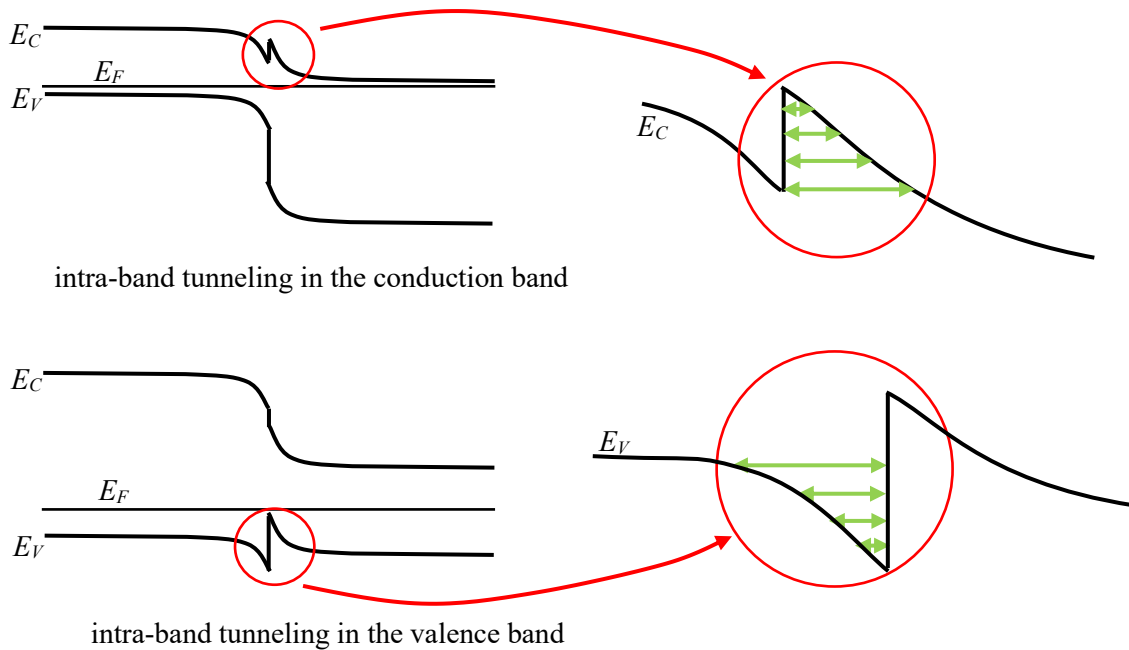
Effective mass of electrons	⇕ 1.000E+0
Effective mass of holes (rel.)	⇕ 1.000E+0
☑ Intraband Tunneling	

Relevant part of the Interface Panel

Figure 3.33   Intra-band tunneling in the conduction and in the valence band. At the bottom, the check box and m_{eff}^{*} values to be enabled in the Interface Panel are shown.
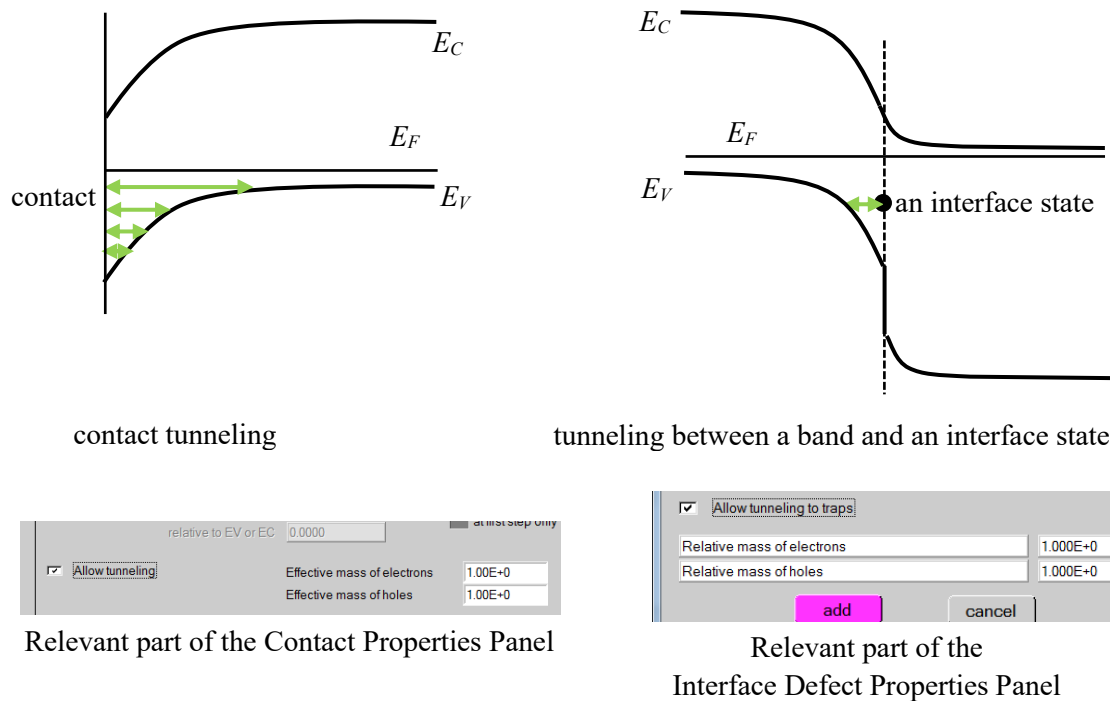
Figure 3.34   Contact tunneling and tunneling to interface states: implemented in SCAPS. At the bottom, the check box and m_{eff}^* values to be enabled in the Contact Properties Panel and the Interface Defect Properties Panel are shown.

3.10.2 Numerical tunnel settings

The tunnel probability is determined by the effective mass of the two states which are involved in the process. These masses can be different, and can be set differently in SCAPS. You can select on the numerical panel, which of the two masses to use. The default choice is to use the minimum effective mass.

SCAPS uses the WKB-approximation in order to calculate tunnel probabilities. This approximation is only strictly valid for large enough barriers. When no precautions are taken, the tunnel rate for very thin and shallow barriers is hugely overestimated. In order to avoid this you can set a minimum height of the tunnel barrier (default = $2k_BT$), for such small barriers the current is determined by drift and diffusion rather than tunnelling anyway.

Up to now four different tunnel processes are present in SCAPS. Each of them can easily be forbidden/allowed by (un)checking the appropriate box on the numerical settings panel.

As it is often inconvenient to check all *allow tunneling* boxes on the different layer and interface panels, this can automatically be performed by clicking the *check tunneling in all layers* button on the numerical panel; it is still advised to check afterwards which tunnel mechanisms are enabled/disabled in the various panels.
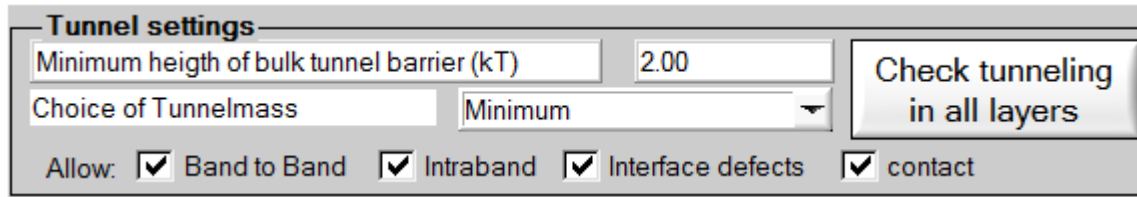
Figure 3.35   Tunnelling settings on the numerical panel

3.11 The blue button

In SCAPS 2.8.4 and earlier, it happened that a panel requiring input was hidden under other panels, and difficult to find; it could also happen that such panel disappeared, which caused a hang-up of the program. These problems have been fixed now. There are some limitations to the number and kind of panels that can be opened or operated simultaneously. Above all, the F1 key acts as an emergency button that brings the cell definition panel on top; on that panel, you find a new blue button [Other panels need input first!] that displays a nicely structured list of all input panels currently open, and that allows you to navigate between them (or to quit all input at once). Pressing the F2 key redirects you to the current layer or interface panel.

3.12 Saving and loading problem definitions

Almost every setting in SCAPS can be saved to a file and loaded again. These files are standard ASCII-files which can be read and edited with e.g. *Notepad*. However, editing is at own risk, and we highly advise against it. An overview of the main file-types with their default extension and directory are given in Table 3.3.

Table 3.3   Main file-types with their default extension and directory

SCAPS filetypes	default extension	default directory
problem definition	.def	scaps\def
material properties	.materials	scaps\materials
action list	.act	scaps\def
initial workingpoint	.wp2	SCAPS\def
batch settings	.sbf	scaps\bdf
recorder settings	.srf	scaps\bdf
scaling	.scl	scaps\scaling
SCAPS scripts	.script	scaps\script
all SCAPS settings in one	.scaps	scaps\def

Other files can not be edited within SCAPS but are needed as input files. These files are text files (ASCII files) that you can construct and edit with a simple text processor, e.g. *Notepad*, see Table 3.4. Each file has a header that can be used to explain what the data represent and where they are coming from. The files distributed with SCAPS can serve as an example to construct ones own input files. Take in mind that SCAPS expects meter-based SI units in input files, except for position x (μm) and for wavelength λ (nm).

Table 3.4   File-types of SCAPS input data files

SCAPS filetypes	units	default extension	default directory
spectrum $\Phi(\lambda)$	nm, W/m^2	.spe	scaps\spectrum
generation $G(x)$	μm, 1/m^3.s	.gen	scaps\generation
absorption $\alpha(\lambda)$	nm, 1/m	.abs	scaps\absorption
filter $T(\lambda)$ or $R(\lambda)$	nm, %	.ftr	scaps\filter
optical capture cross section $\sigma_n(\lambda)$, $\sigma_p(\lambda)$	nm, m^2	.opt	scaps\optcapt
grading $y(x)$, $E_g(y)$, $N_A(x)$, …	x in μm, properties in m-based SI units	.grd	scaps\grading

When definition files are exchanged between users (or between your lab computer and your home computer), it can happen that a .def file contains an input file name (.abs, .ftr, .opt, .grd, …) that is not present on the computer. When leaving the solar cell definition panel with the OK button, SCAPS checks if all files are found, and if not brings you to a panel where you can either browse for an existing filename, or set the property to the default internal model instead of 'from file input'.
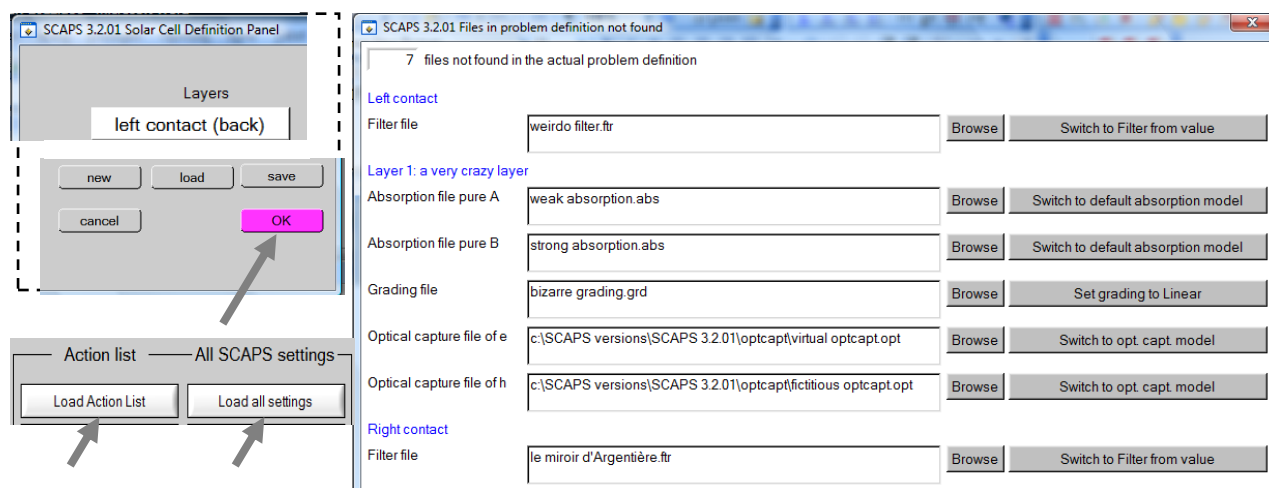
Figure 3.36   SCAPS checks if all necessary files can be found: after loading an action list or loading all SCAPS settings, and after leaving the Solar Cell Definition Panel with 'OK' (see arrows in screen shots at the left side). Then, a panel opens that displays a list of all files not found in the problem definition (see screen shot at the right side). You can either browse for an existing file, or set the property 'to model'.

Please note that such check is not carried out when the problem has been input or changed by the SCAPS the batch set-up: you still have the possibility to cause a program crash by inputting non-existing files. Also, when SCAPS is in non-interactive mode (see 7.4 and Figure 7.5), the Check Files Panel is not opened, but everywhere a file is not found, the corresponding property is set 'to model' (this is equivalent with clicking all the 'Set to model' buttons, at the right side of Figure 3.36); a message of what was unfound and has been set to model is written to the SCAPSErrorLogFile.log file.

You can leave the Check files Panel without fixing all unfound files (that is, either browsing for an existing file, or setting the property to model). The calculate button is then disabled, and you see a red warning at the bottom, right of the action panel (Figure 3.37).
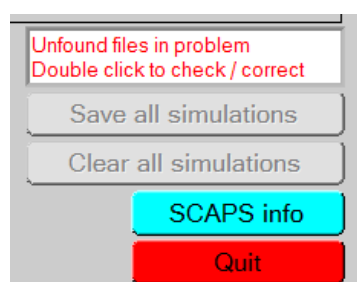
Figure 3.37   When not all unfound files are fixed, a red warning is displayed at the bottom/right of the action panel.

Chapter 4: Working point definition

The workingpoint specifies the parameters which are not varied in a measurement simulation, and which are relevant to that measurement.

4.1 General

These parameters are (immediately) available on the action panel (Figure 4.1).
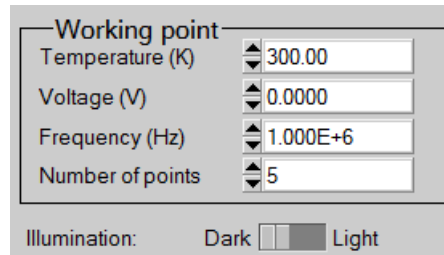
Figure 4.1   Setting the working point conditions, available on the action panel.

- The temperature T: relevant for all measurements. Note: in SCAPS, only $N_C(T)$, $N_V(T)$, the thermal velocities, the thermal voltage k_BT and all their derivatives are the only variables which have an explicit temperature dependence; you must input for each T the corresponding materials parameters yourself.
- The voltage V: is discarded in *I-V* and *C-V* simulation. It is the dc-bias voltage in *C-f* simulation and in QE(λ) simulation. SCAPS always starts at 0 V, and proceeds to the working point voltage in a number of steps that you also should specify, see §5.1.2.
- The frequency f: is discarded in *I-V*, QE(λ) and *C-f* simulation. It is the frequency at which the *C-V* measurement is simulated.
- The work point voltage V and illumination intensity (e.g. in sun) are approached from the equilibrium state in Number of points = n steps: the voltage from 0 to V linearly in n steps, and the illumination intensity P_{in} logarithmically from $10^{-10} \times P_{in}$ to $1 \times P_{in}$.
- The illumination: is used for all measurements and is discussed below in §4.2.

4.2 Illumination conditions

When performing simulations under illumination, you can further specify the illumination conditions. The basis settings are: dark or light, choice of the illuminated side, choice of the spectrum. If you have an optical simulator at your disposal you can immediately load a generation profile as well in stead of using a spectrum.

SCAPS 3.3.05, december 2016: both the incoming spectrum and the generation can be specified either by an input file, or by an analytical model. This is treated in Section 4.3.

4.2.1 Internal SCAPS calculation

SCAPS is able to perform simple optical simulations in order to calculate the generation in your structure. The internal optical SCAPS model is described in Section 4.3.1 below. Hereto the illumination conditions have to be specified (Figure 4.2).

Figure 4.2   Specifying the illumination conditions.

- The illumination side can be changed (left/right)
- A light spectrum file can be selected. Several spectrum files are available in the SCAPS distribution. The default is a one sun (= 1000 W/m^2 = 100mW/cm^2) illumination with the 'air mass 1.5, global' spectrum. A lot of monochromatic spectra are available as well. They come in two versions. They either contain a total illumination power of 1000 W/m^2 (1 sun) in the wavelength band (e.g. '600 nm.spe'), or they contain a number of photons that would yield a light current J_L = 20 mA/cm^2 if QE were unity (e.g. '600nm fixed JL.spe'). All spectrum files which come with the SCAPS distribution are documented with comments. All spectrum files are ASCII-files and can hence be user defined. The first line contains the number of wavelengths that appear in the spectrum file. This line is immediately followed by two columns. The first column specifies the wavelength λ (nm). Suppose the spectral density (in Watt/m^2.nm) is given by P: the incident power in the wavelength interval [λ - dλ/2 , λ + dλ/2] is then given by $Pd\lambda$ (in Watt/m^2). The second column in the spectrum file contains this incident power $Pd\lambda$ (in Watt/m^2) in a wavelength interval dλ around λ. As such, the total incident power is given by the sum of the values in the second column. The wavelength interval dλ considered around a wavelength λ_i is: dλ = (λ_{i+1} - λ_{i-1})/2. For the first wavelength λ_1 it is dλ = λ_2 - λ_1, and for the last wavelength λ_N it is dλ = λ_N - λ_{N-1}. The maximum number of wavelengths appearing in the spectrum file equals 2500. The number of incident photons, per m^2 and per s, of wavelength between λ-dλ/2 and λ+dλ/2 is then set equal to 5.035×10^{15} × λ × $Pd\lambda$, λ in nm, $Pd\lambda$ in Watt/m^2. The numerical factor equals 10^{-9}/hc where Planck's constant h and the light velocity c are expressed in SI units. For convenience, the spectrum file may be preceded by comment lines, which begin with the character ">" or "/". These lines are not read by SCAPS. They can be used to identify the source or the purpose of a given spectrum file.
- The incident power of the incoming light can be weakened or amplified by applying a neutral density filter. The value ND of this filter can immediately be set, or the transmission of the filter T = 100% × 10^{-ND}. You can to set ND < 0 or T > 100 %, to do simulations under concentrated light. The incident power can also be be attenuated by an internal/reflection at the front contact which should be specified on the cell definition panel.
- No reflections at the interfaces are implemented. Only at the contacts you can specify a (wavelength dependent) reflection/transmission (§3.2).
- A short/long wavelength cut-off filter can be applied.

4.2.2 Generation from file

There is also a possibility to input the generation in the structure immediately. This can be useful if you have a better optical simulator than SCAPS, or if you want to mimic e.g. EBIC measurements.
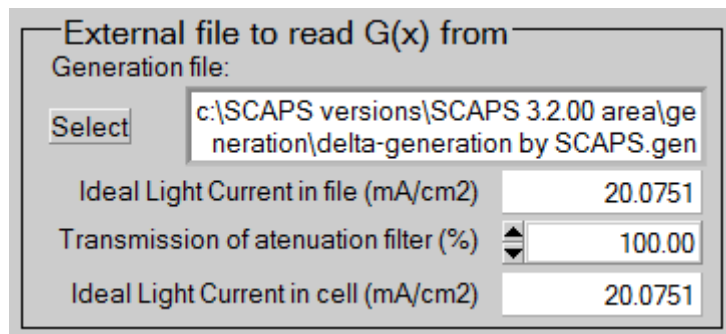
Figure 4.3   Settings when using a generation file.

The advised way to create a $g(x)$ file is as follows: set up a problem in SCAPS; perform a calculation with the internally calculated $g(x)$. Save the results in the EB-panel (energy bands panel), or, alternatively, press show in the action panel and copy/paste the desired data in a user file (e.g. Origin or Excel). Then you have the x-coordinates of the nodes used by SCAPS. Make a generation file with the $g(x)$ data from your own model, calculation or measurement… Mind that SCAPS input files want data in SI units (only position x is wanted in μm, not m). As a consequence, a SCAPS genereation file expects the $g(x)$ data in two columns, first column x in μm, second column g in $m^{-3}s^{-1}$. For an example of a valid format, see the examples coming with SCAPS (file extension.gen).

You can always save the generation $g(x)$ that was set by the internal SCAPS illumination model in a valid SCAPS generation input file using the script command `save results.SCAPSgenerationfile`, see section 10.4.3. And then you can replace the g data in the second column by your own data.

To use your own generation file, click 'Generation of eh pairs determined from file' in the action panel, and select your file. When the $g(x)$ data in your file do not use the x values used by SCAPS, the program will have to interpolate between your x values, and this might cause inaccuracy. To judge the interpolation, you can visualize the $g(x)$ data in the *Generation-Recombination profiles panel,* see §6.4.2. The $g(x)$ profile you will see then is, however, only updated after a calculation has been performed with the calculated or user specified $g(x)$.

Additionally you can still use an attenuation filter, which would mimic the effect of a neutral density filter. Again, setting an attenuation > 100 % corresponds to a measurement under concentrated light. The ideal light current in the file (Figure 4.3) is the total number of *eh* pairs in the generation file, thus integral of all g values over the cell thickness, and divided through the elementary charge q. The idelal light current in the cell is the ideal light current in the file, multiplied with the transmission of the attenuation filter.

A generation file is a table of x, g values (x in μm, g in $m^{-3}s^{-1}$). It contains information about the number of *eh* pairs generated, but not about the energy of the photons that created the *eh* pairs. As a consequence, no solar cell efficiency can be calculated. However, a 'collection efficiency', that is the ratio of the short circuit current to the ideal light current in the file, can be calculated, and is displayed in the IV panel. Also, a relevant spectral response QE cannot be calculated, since the generation file does not contain any information on wavelength.

4.3 Generation and spectrum models (SCAPS 3.3.05)

4.3.1 The internal optical model of SCAPS

In its default mode, SCAPS calculates the generation of *eh*-pairs $G(x)$ from the incident photon flux $N_{phot0}(\lambda)$ from a very coarse optical model. This model contains only reflection/transmission at the two

contacts, and absorption in the semiconductor layers. Thus: no interference, no scattering, no intermediate reflectors… For a single layer of thickness d, with uniform optical absorption constant $\alpha(\lambda)$ (uniform here means that α does not depend on position within the layer, thus $\alpha(\lambda, x) = \alpha(\lambda)$), and for light incident from the left with photon flux $N_{phot0}(\lambda)$, the photon flux $N_{phot}(\lambda, x)$ at each position in the layer is given by

$$N_{phot}(\lambda, x) = N_{phot0}(\lambda) \cdot T_{front}(\lambda) \cdot \exp(-x\alpha(\lambda)) \cdot \frac{1 + R_{back}(\lambda)\exp(-2(d-x)\alpha(\lambda))}{1 - R_{back}(\lambda)R_{int}\exp(-2d\alpha(\lambda))} \qquad (15)$$

where

$T_{front}(\lambda)$ is the (possibly wavelength dependent) transmission of the front contact; the front contact is where the light is incident, here the left contact. T_{front} is in SCAPS a property of the contact, and thus is input in the Contact Properties Panel of the front contact.

$R_{back}(\lambda)$ is the (possibly wavelength dependent) reflection at the back contact; the back contact is at the opposite side from where the light is incident, here the right contact. R_{back} is in SCAPS a property of the contact, and thus is input in the Contact Properties Panel of the back contact.

R_{int} is a (not wavelength dependent!) internal reflection at the front contact. In SCAPS, the value of R_{int} or its complement $T_{int} = 1 - R_{int}$, should be input in the Solar Cell Definition Panel, see Figure 4.4.
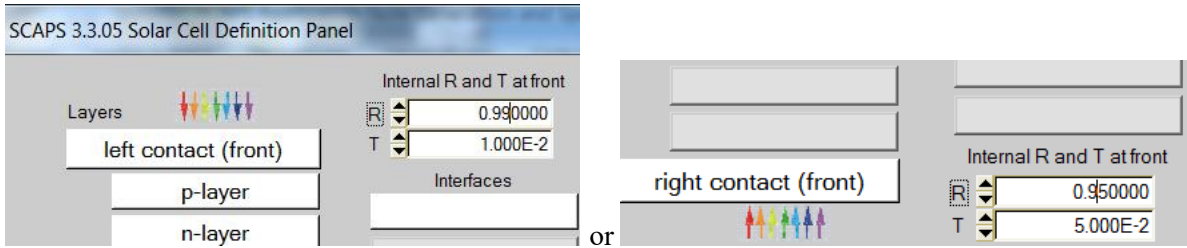
Figure 4.4   Input fields for the internal reflection R_{int}, or its complement the internal transmission $T_{int} = 1 - R_{int}$, in the Solar Cell Definition Panel. The position of the input fields depend on the side where the light is incident. These input fields are only visible if a reflection at the back contact R_{back} is set.

Eq. (15) is simple to interpret: If there is no reflection at the back contact ($R_{back} = 0$), there is only one pass of light through the cell, and the light is extinguished with the familiar exponential function $\exp(-\alpha x)$. If $R_{back} \neq 0$, but the internal reflection at the front contact is zero, $R_{int} = 0$, there is a second pass of the light reflected at the back contact, and an additional extinguishing function $\exp(+\alpha x)$ appears in (15) for the reflected light. If there is internal reflection $R_{int} \neq 0$, there is an infinite number of passes of the light, that are summed up in (15) yielding the denominator.

In SCAPS, this model is extended to more than one layer, to layers with non-uniform (= position dependent) absorption $\alpha(\lambda, x)$, and to the possibility of light to be incident on the other contact. But there is no more physics than that already described in Eq. (15).

From the photon flux $N_{phot}(\lambda, x)$ at each position and for each wavelength, the generation of *eh* pairs is then calculated as

$$G(\lambda, x) = \alpha(\lambda, x) \cdot N_{phot}(\lambda, x) \qquad (16)$$

This is then integrated over all wavelengths of the incoming photon flux:

$$G(x) = \int_{\lambda_{min}}^{\lambda_{max}} G(\lambda, x)d\lambda = \int_{\lambda_{min}}^{\lambda_{max}} a(\lambda, x) \cdot N_{phot}(\lambda, x)d\lambda \qquad (17)$$

where λ_{min} and λ_{max} are the short and long wavelength limits of the incoming photon spectrum $N_{phot0}(\lambda)$, as it is specified in the spectrum input file.

Users that are not satisfied with the optical model of SCAPS, can directly specify the *eh* generation G(x) ate each position. The optical model of Eqs. (15)-(17) is then not used. Since the impurity photovoltaic effect (IPV) relies on this internal optical model (with extensions), assigning IPV properties to defects is incompatible with a user specified generation G(x). Precautions in SCAPS prevent the user to define both in one problem.

4.3.2 Analytical models for the generation and for the spectrum: motivation

Direct specification of G(x) through an input file with extension .gen is intended for users who want to use SCAPS for the electrical calculations, but prefer to do the optical calculations themselves. A number of users have applied SCAPS with a file input for G(x) to simulate electron beam induced current (EBIC) measurements. Typically, they would define intense generation in a small sheet of the cell, and calculate the current response when the central position of the beam (the position of the 'intense generation') is swept or scanned over the cell thickness. For each position of the *e*-beam, a separate generation file should be made, and this proves to be a rather complicated and time consuming procedure. There was a little improvement for those brave users with SCAPS 3.3.02 of july 2015, where script commands where introduced that called one file from a list of file names, but still it was labour intensive work.

Another problem is the mesh, thus the set of *x* values. SCAPS has an internal mesh generator that defines the positions *x* where the semiconductor equations are solved. But a user often has his own mesh generator for his own optical model. When the two meshes do not coincide, interpolation is used by SCAPS, but that can be a cause of numerical inaccuracy. For example, a narrow EBIC profile G(x) specified by a user could fall completely between two adjacent SCAPS mesh points in the middle of a thick layer, so that SCAPS wouldn't even 'see' the *e*-beam…

That is why we introduced in SCAPS 3.3.05 of December 2016 the possibility of defining G(x) from an analytical model, where all parameters of the model (e.g. 'central beam position') can be manipulated in the SCAPS batch, recorder and script facilities. And in the same move, analytical models for the spectrum were implemented.

4.3.3 New facilities in SCAPS 3.3.05

The main SCAPS panel, the Action Panel was modified somewhat.
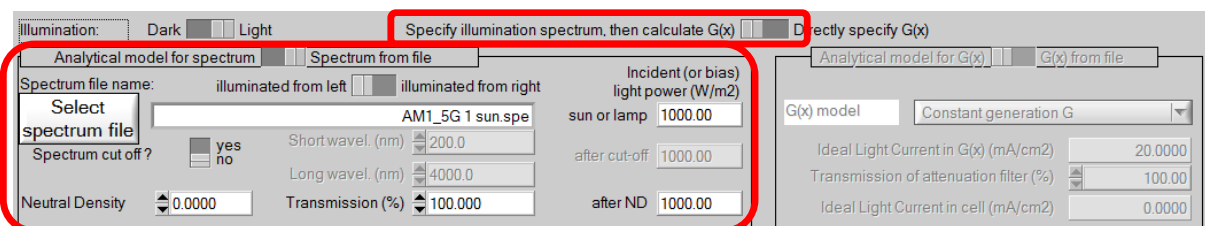
Figure 4.5   Using the internal optical model of SCAPS: define a spectrum, let SCAPS calculate G(x) with Eqs. (15) - (17). The selection switch marked on top of this figure is only active when the light is switched on.

The default setting is shown in Figure 4.5: a spectrum is specified from a file, and the internal optical model of SCAPS is used. External optical filters can be applied: two spectrum-cut-off filters and a neutral density filter. When the model/file switch in the spectrum part of Figure 4.5 is set to 'Analytical model for spectrum' (instead of 'Spectrum from file'), the new facilities to calculate a spectrum from a (simple) model are enabled, see Figure 4.5 The available spectrum models are discussed below.

When in Figure 4.5 the option 'Directly specify G(x)' is set, one can set the generation G(x) from an input file or from analytical model, see Figure 4.7. The available generation models are discussed below.
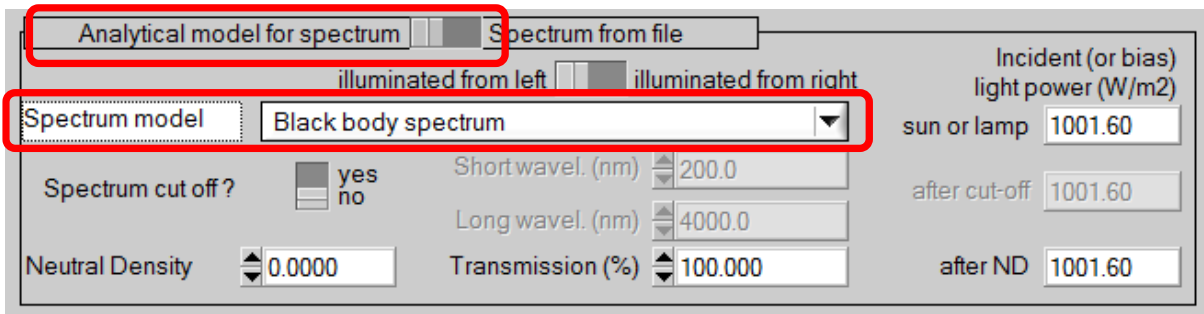
Figure 4.6   Setting an analytical model for the spectrum, here a black body spectrum is selected.
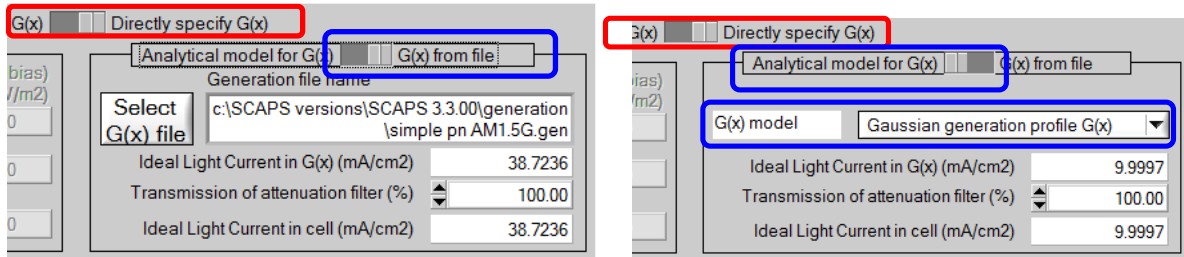
Figure 4.7   Directly specifying the generation $G(x)$. Left: take $G(x)$ from an input file with extension .gen (as in previous SCAPS versions). Right: calculate $G(x)$ from an analytical model (new in SCAPS 3.3.05). The selection switch marked in red is only active when the light is switched on.

4.3.4 Analytical models for the generation $G(x)$
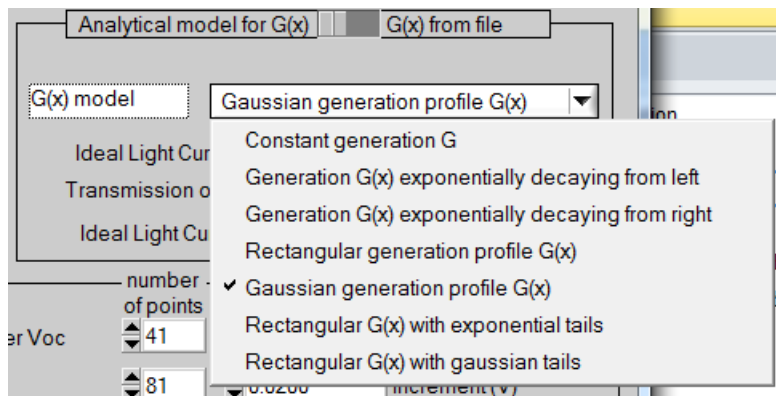
Figure 4.8   The available generation models

As of now, the user can select one from seven models as shown in Figure 4.8. When selecting one of them, e.g. here a Gaussian generation profile $G(x)$, the new SCAPS Model Panel opens, see Figure 4.9. The panel is operated in much the same way as the SCAPS Grading Panel, that is maybe familiar to many users.
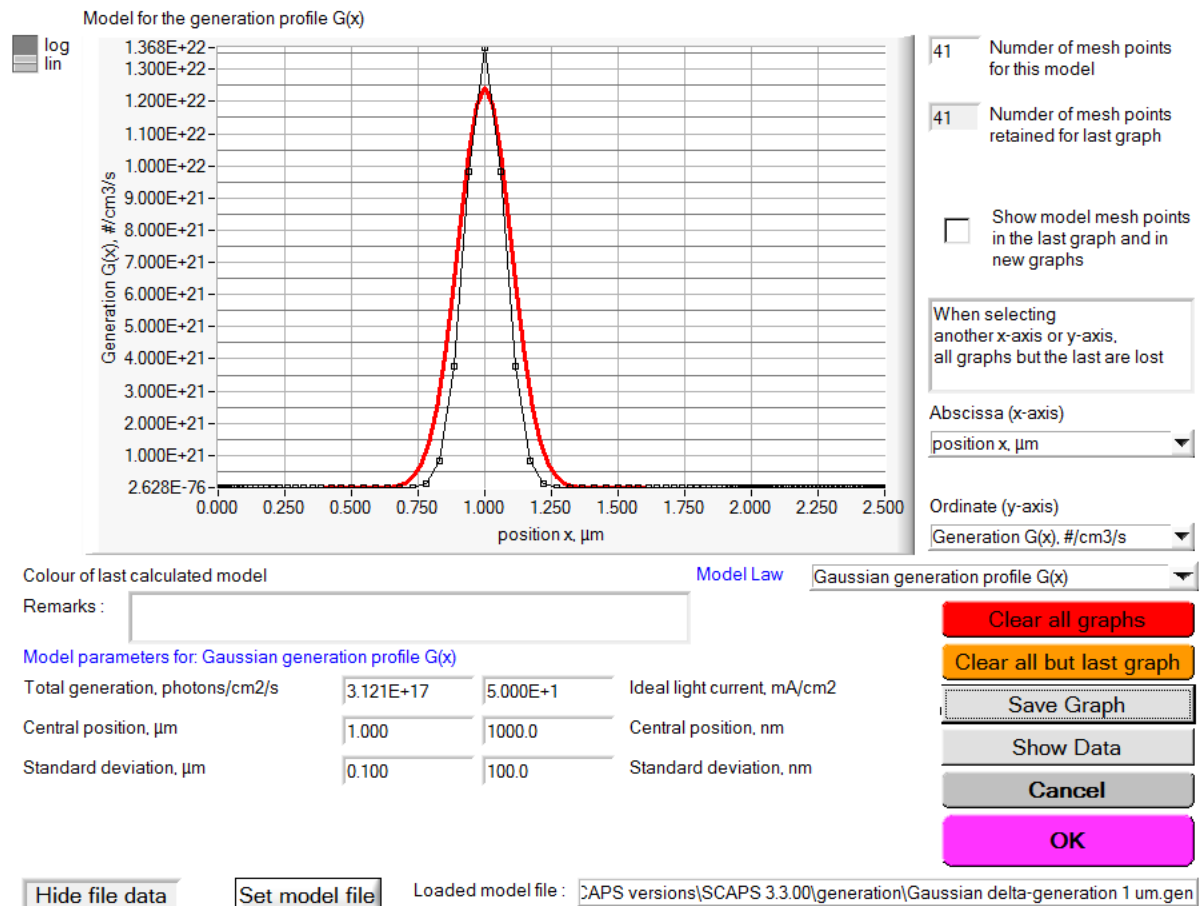
Figure 4.9   The SCAPS Model Panel, showing a modelled gauss curve $G(x)$, and also the generation profile from an input file, that can be used for comparison with the $G(x)$ model. The meaning parameters is (should be, I hope) self-understood.
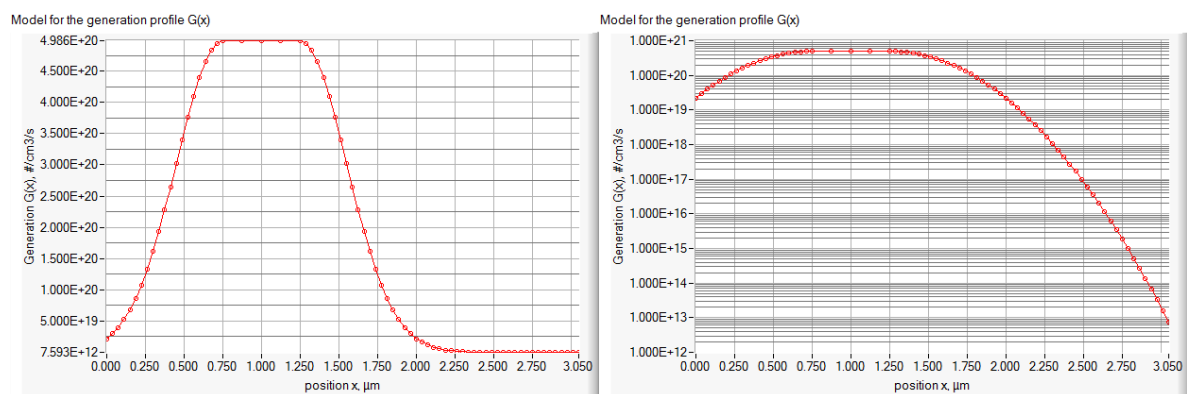
Figure 4.10   Linear and logarithmic view of the generation model 'rectangular with Gaussian tails', with parameters: central position 1 µm, total width of rectangle 0.5 µm, standard deviation of Gaussian tails 0.3 µm. Ideal light current: 10 mA/cm².

4.3.5 The SCAPS Model Panel

The number of x points is set at the top right of the Model Panel (Figure 4.9, $n = 41$ is set). But modelled points with $x < 0$ are discarded. The remaining number of points is displayed for information. In the example of Figure 4.10, 101 x-points were asked, but only 73 have $x > 0$ and are retained. x points beyond the cell thickness d, thus points with $x > d$, are not discarded, although they are not used in the SCAPS calculations of course. But when one would increase the thickness of a layer, points beyond the original thickness could fall inside the cell, and they will be accounted for. The x-mesh of the generation profile $G(x)$ is independent from

the x-mesh of the SCAPS calculations, and usually contains much less points. One can or cannot include the $G(x)$ mesh points into the SCAPS mesh by clicking this option in the Numerical Panel, see Figure 4.11.
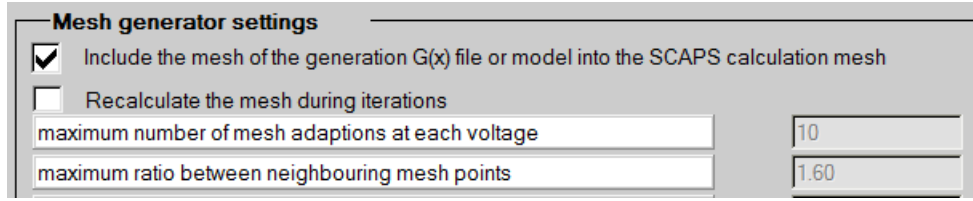
Figure 4.11   In the Numerical Panel, one can ask (or not) to include the mesh points of $G(x)$ into the SCAPS mesh.

Also in the right part of the Model Panel of Figure 4.9, one can select the abscissa (x-axis) and the ordinate (y-axis) of the plot, see Figure 4.12.
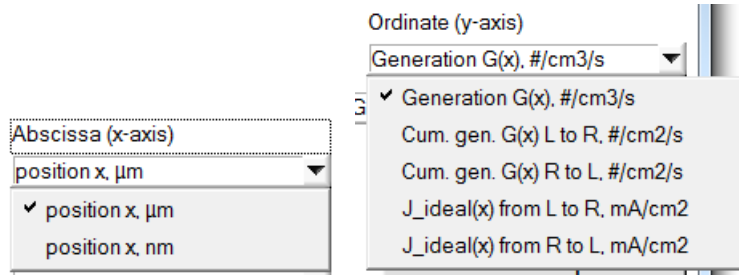
Figure 4.12   Possible choices for the abscissa (x-axis) and ordinate (y-axis) to be displayed in the Model Panel for generation models.

The x-axis options are trivial (micrometer or nanometer units), but the y-axis options are more interesting. The definitions are:

$$G_{\text{cumul, L}\rightarrow\text{R}}(x) = \int_0^x G(x')\,dx' \quad \text{and} \quad J_{\text{ideal, L}\rightarrow\text{R}}(x) = q \cdot \int_0^x G(x')\,dx'$$

$$G_{\text{cumul, R}\rightarrow L}(x) = \int_x^d G(x')\,dx' \quad \text{and} \quad J_{\text{ideal, R}\rightarrow L}(x) = q \cdot \int_x^d G(x')\,dx'$$

(18)

These integrals are calculated numerically. The total generation, expressed as a light current, thus

$$J_{\text{ideal}} = J_{\text{ideal, L}\rightarrow\text{R}}(d) = J_{\text{ideal, R}\rightarrow L}(0) = q \cdot G_{\text{tot}} = q \cdot \int_0^d G(x)\,dx$$

(19)

is displayed in the Action Panel, see Figure 4.7 right (the value 9.9997 mA/cm² indeed slightly deviates from the parameter value 10.0 mA/cm² that was set). If e.g. a substantial fraction of the $G(x)$ profile modelled would have $x < 0$ and thus lay outside the cell, the displayed J_{ideal} value would substantially be lower than the J_{ideal} parameter value set.

The name of the model parameters is self-evident, we hope. For most parameters, you have two alternative forms of the same parameter; input one and the other is adapted automatically. Examples are: positions in μm or in nm; total (integrated over cell thickness) G_{tot} values as a photon flux (# photons.s⁻¹cm⁻²) or as an ideal current (mAcm⁻²). And finally, as is customary in SCAPS, there is only one mouse click between you and your logarithmic thoughts ☺.

You can display also the $G(x)$ data from an input file, but that is only for information: it allows to compare a modelled $G(x)$ with $G(x)$ data from a file. To actually set $G(x)$ from an input file, proceed as in Figure 4.7 left. (And don't forget, SCAPS is expecting the input files as a table of x (in μm) G_{eh} (in #/m³.s), separated by white space (blanc or tab)).

4.3.6 The generation models

4.3.6.1 Constant generation G

There is only one parameter, G_{tot}, expressed either as a photon flux or as an ideal current. This generation is smeared out uniformly over the cell thickness. Thus, if the cell thickness is changed subsequently, the generation $G(x)$ is changed accordingly, but G_{tot} remains the same.

4.3.6.2 Exponentially decaying $G(x)$

The first parameter is G_{tot} or J_{ideal}, as in the previous model. The other parameter is the characteristic length of the exponential decay. The $G(x)$ laws are

$$G(x) = \frac{G_{tot}}{L_{decay,\,L}} \exp\left(-\frac{x}{L_{decay,\,L}}\right) \quad \text{or} \quad G(x) = \frac{G_{tot}}{L_{decay,\,R}} \exp\left(-\frac{d-x}{L_{decay,\,R}}\right) \tag{20}$$

In these equations, d is the actual cell thickness.

4.3.6.3 Gaussian generation profile $G(x)$

The parameters are: G_{tot} or J_{ideal}, the central position $x_{central}$ and the standard deviation σ. Part of the profile could fall left of the cell (thus $x < 0$), and then will be discarded. When the cell is too thin for the defined profile, part of the Gauss curve with $x > d$ will not be accounted for.

4.3.6.4 Rectangular generation profile $G(x)$

The first parameter is G_{tot} or J_{ideal}, as in the previous models. The other parameters are the central position of the rectangle, and the full width. When you specify e.g. $x_{central} = 1$ μm and $w_{full} = 0.3$ μm, the generation will be a constant between $x = 0.7$ μm and 1.3 μm and 0 outside. When the cell thickness was only 1 μm, or you set a total thickness of 1 μm afterwards, only the left half of the generation profile will lay within the cell.

4.3.6.5 Rectangular generation profile $G(x)$ with tails

This works like a rectangular profile, but with a tail left and a tail right; both tails are symmetric. They can be exponential (then define the decay length), or (half) Gaussian (then define the standard deviation).

For all $G(x)$ profiles with exponentials or Gauss curves, it might be worth while to look at the logarithmic graph!

4.3.7 The spectrum models

The available spectrum models are shown in Figure 4.13.
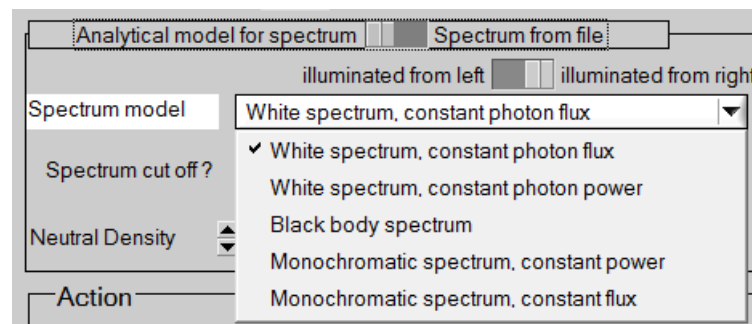
Figure 4.13   The available spectrum models

Again the SCAPS Model Panel opens upon selecting a spectrum model. The options for the *x*-axis and *y*-axis for spectra are shown in Figure 4.14.
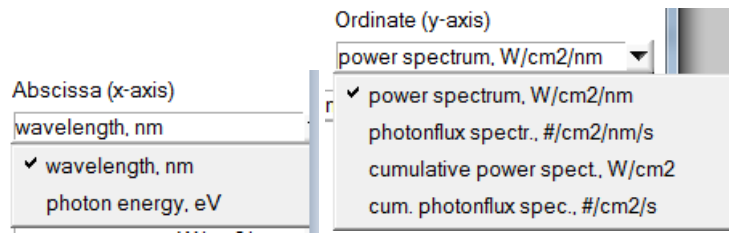
Figure 4.14   Possible choices for the abscissa (*x*-axis) and ordinate (*y*-axis) to be displayed in the Model Panel for spectrum models.

The abscissa can be switched between wavelength and photon energy. The ordinate can be based on the photon flux (the number of photons incident at each wavelength) or the photon power (the photon power incident at each wavelength). Both can be shown as such, or integrated from the short wavelength limit of the spectrum ('cumulative spectrum').

There are two versions of white spectra: one has a constant photon flux over the specified wavelength range (expressed as a total photon flux or ideal current), the other has a constant photon power over the wavelength range (expressed as total incoming power, in suns or in mW/cm²; 1 sun = 100 mW/cm² = 1000 W/m²). Similarly, there are two versions of monochromatic spectra, one with specified total photon flux, and one with specified total photon power. The black body spectrum is there to please our colleagues more oriented towards fundamental theory. Of course, the black body temperature is the main parameter (input $T_{blackbody}$ or the wavelength of the maximum, according to Wien's displacement law). As you will observe when playing with this model, black body radiation extends very far in the infrared wavelength range. To avoid spending many λ-mesh points on not-useful wavelengths, one can limit the λ range of interest.

4.4 The initial working point

In order to perform simulations for metastable defects (§3.8) two (different) working points are needed. The working point which is present since the birth of SCAPS still corresponds to the working point which is applied during the 'measurement'. A second working point 'the initial working point' corresponds to the conditions which are applied to the structure in order to bring it in the desired metastable state. For example if you want to simulate a red-on-bias experiment, this working point will consist of a negative voltage bias and red illumination.

Pay attention! The temperature at this initial working point is usually high as the metastable transitions are thermally activated. Many metastabilities can not be induced at low temperature because the transition is too slow and you never get to an equilibrium situation in a reasonable time span. SCAPS always calculates equilibrium conditions, and waits in fact until an infinite amount of time has passed. You should take this into account when setting a low initial working point temperature .

The 'Initial State Workpoint Panel', where you can set the initial working point can be accessed from the action panel only if at least one metastable defect is present in the problem definition.
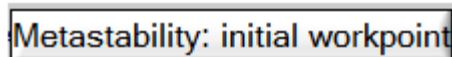
Figure 4.15   Accessing the initial workpoint panel: press this button on the action panel.

You can save and load all settings of the initial state workpoint into and from a file. These files have extension .wp2 (from 'second work point'), and reside in the \def folder.

4.5 Shunt conductance and series resistance

It is possible to introduce an external shunt conductance and series resistance to the structure on the action panel, see Figure 4.16.

Figure 4.16 Introducing external shunt conductance and series resistance

Both resistances can be switched on/off and for a shunt conductance you can either define its resistance or conductance.

These external resistances only affect *I-V*, *C-V* and *C-f* simulations (no *QE* simulations) The voltage which you apply to the sample (either the working point voltage or the voltage in the iv or *C-V* simulation) is interpreted as the internal voltage over the sample, without external resistance. This voltage gets then recalculated after simulation in order to get the external applied voltage in *I-V* and *C-V* simulations. In small signal simulations, the value of *G* and *C* are also recalculated after simulations.

The values of the external resistances are saved in the definition file of the solar cell structure.

Chapter 5: Single shot calculations

The main functionality of SCAPS is to solve the one-dimensional semiconductor equations. In the bulk of the layers these equations are given by (21) together with the constitutive equations (22).

$$\begin{cases} \dfrac{\partial}{\partial x}\left(\varepsilon_0\varepsilon\dfrac{\partial\Psi}{\partial x}\right) = -q\left(p - n + N_D^+ - N_A^- + \dfrac{\rho_{def}}{q}\right) \\[2ex] -\dfrac{\partial J_n}{\partial x} - U_n + G = \dfrac{\partial n}{\partial t} \\[2ex] -\dfrac{\partial J_p}{\partial x} - U_p + G = \dfrac{\partial p}{\partial t} \end{cases} \tag{21}$$

$$\begin{cases} J_n = -\dfrac{\mu_n n}{q}\dfrac{\partial E_{Fn}}{\partial x} \\[2ex] J_p = +\dfrac{\mu_p p}{q}\dfrac{\partial E_{Fp}}{\partial x} \end{cases} \tag{22}$$

Together with appropriate boundary conditions at the interfaces and contacts, this results in a system of coupled differential equations in (Ψ, n, p) or (Ψ, E_{Fn}, E_{Fp}). SCAPS numerically calculates a steady state and a small signal solution of this system. Hereto, the structure is first discretized (creating a mesh). A steady state workingpoint situation (see §Chapter 4) is calculated and when required a small signal analysis is performed.

5.1 Calculation roadmap

5.1.1 Meshing

The first step in every calculation is to discretize the structure. The meshing algorithm of SCAPS provides:

- Coarse meshing in the middle of a layer
- Finer meshing near the interfaces and contacts
- Two discretization points (with identical spatial coordinate) for each interface
- The mesh can be optimized during calculation.

  The result of the discretization algorithm is illustrated in Figure 5.1.
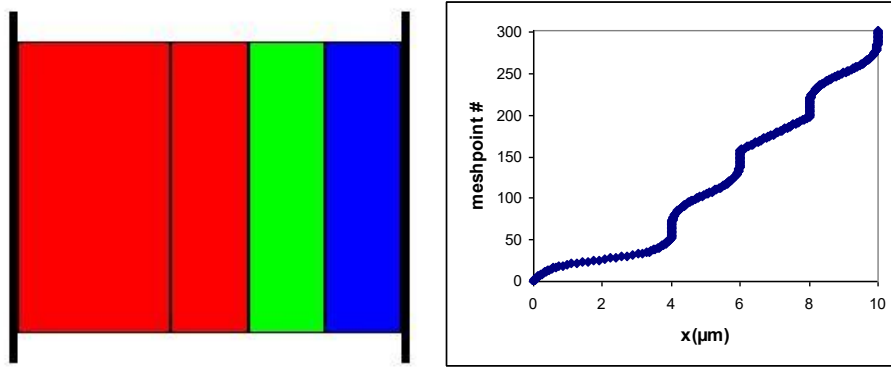
Figure 5.1   Result (*right*) of the discretization of the structure on the *left*. (*Red: p-type, blue: n-type, green: intrinsic*)

The basic algorithm is designed to provide a lot of points in regions where properties experience large variations (close to interfaces/contacts) and fewer points where the properties are expected to remain fairly constant (in the bulk). However, when performing simulations with strong gradings, with multivalent defects or with the IPV-effect this meshing procedure might be unsufficient. Hence, it is possible to optimize the mesh at every iteration step. This option can be set on the numerical panel as shown in Figure 5.2.

Figure 5.2   Recalculate mesh settings on the numerical panel

When *recalculate mesh* is switched on at each iteration the ratio between the following properties at two adjacent meshpoints are assessed: $\exp(q\Phi/kT)$, $\exp(E_{Fn}/kT)$, $\exp(E_{Fp}/kT)$ (all dimensionless), the recombination rate R and the generation rate G (both R and G in $/m^3$). When that ratio is larger than the maximum ratio f_{max} set on the numerical panel, extra meshpoints will be inserted to enable a better simulation. If the ratio is smaller than the minimum ratio f_{min} set on the numerical panel, the meshpoint is removed. Precautions are taken not to remove too many meshpoints in a row. The ratio of the recombination and generation rate are only taken into account if $q|R| < R_{limit}/L$ or $qG < G_{limit}/L$ respectively. Here R_{limit} and G_{limit} are parameters set on the numerical panel (atypically for SCAPS, these parameters are displayed and saved in units of $\mu A/cm^2$), and L is the total cell thickness. At room temperature, the default value $f_{max} = 1.60$ corresponds to a difference of about 12 mV or 12 meV of Φ, E_{Fn} or E_{Fp} between two adjacent mesh points, whereas the default value $f_{min} = 1.05$ corresponds to a difference of 1.2 mV or meV.

The *recalculate mesh* settings are all saved in the definition file.

The effect of the recalculation of the mesh is illustrated in Figure 5.3, where the defect occupation of an amphoteric defect is calculated.
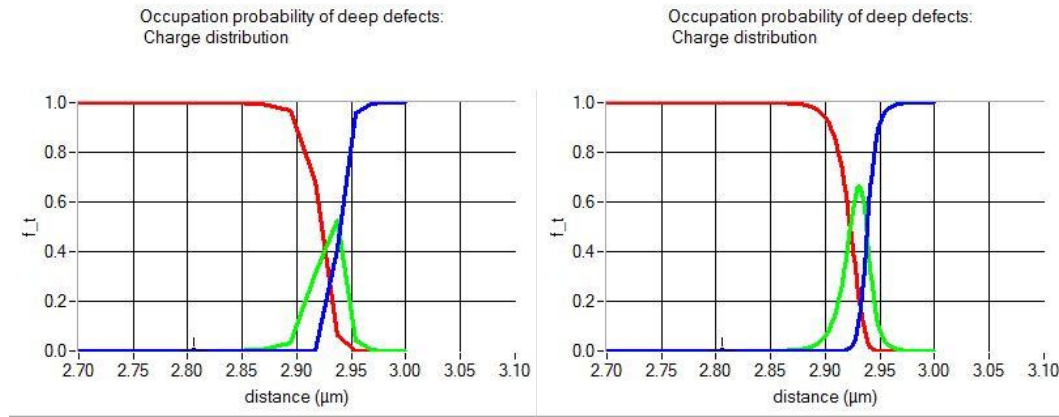
Figure 5.3   Occupation of an amphoteric defect calculated without (*left*) and with (*right*) mesh recalculation

5.1.2 The pathway to a solution

When clicking the 'Calculate: single shot'-button, the measurements you selected are calculated for the given structure. Hereto the system of (21) is solved numerically, using a Gummel iteration scheme with Newton-Raphson sub-steps [12, 15]. The parameters of this scheme are available on the numerical panel as convergence settings. As a rule of thumb, the best way to improve convergence is first to increase the number of iteration steps and then perhaps increase the termination criterion thresholds.

In order to get to a solution, an initial guess has to be made. Care should however be taken, because when this initial guess differs too much from the solution of the system, the iteration procedure will fail to converge (after a reasonable number of iterations steps). SCAPS does not ask an initial guess from the user, but calculates several situations in order to get to a solution under the working point circumstances and at the first measurement point. The strategy which is followed is illustrated in Figure 5.4.
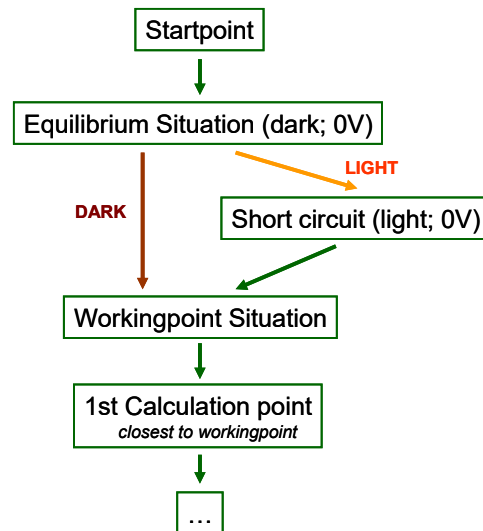
Figure 5.4   Getting to the working point and first calculation point…

Every calculation starts at the startpoint. This is a very simple situation, assuming the quasi-Fermi levels to be zero throughout the structure and no potential drop to be present over the structure. This is used as an initial guess to get to the equilibrium situation (no illumination, no voltage applied). When the working point conditions are under darkness, this equilibrium condition is used as an initial guess to calculate the solution under working point conditions. When illumination is switched on however, the short circuit situation is calculated in an intermediate step to serve as the next initial guess.

When the difference between two situations is too large, the iteration procedure will diverge. Therefore, SCAPS implicitly introduces intermediate situations between the equilibrium/working point; equilibrium/short circuit and the short circuit/working point situation. Hereby gradually increasing the applied bias voltage or

the illumination strength. The number of intermediate steps to be taken can be controlled on the action panel, see Figure 4.1. When divergence occurs, increasing this number can sometimes lead to convergence, however, this slows down your calculation.

When the working point situation is calculated SCAPS immediately calculates the first calculation point of the measurements to be simulated (without any sub-steps). For *C-f* and *QE* simulations, this first point is identical to the working point conditions. For *I-V* and *C-V* simulations a voltage range has been given, independently from the working point voltage. SCAPS will then jump to the edge of this voltage range which is the closest to the working point voltage. IT IS THUS STRONGLY ADVISED TO CHOOSE THE WORKING POINT VOLTAGE IDENTICAL TO THE STARTING POINT OF THE VOLTAGE RANGE YOU WANT TO SIMULATE.

When performing batch (or recorder) simulations, every calculation is started at the startpoint again. This is often unwanted, e.g. when gradually increasing the working point voltage it is more logical to use the previous batch calculation as an input for the next calculation. Varying, voltage, frequency, series/shunt resistance in a batch calculation, this latter approach can be taken by checking the appropriate checkbox on the numerical panel.

5.1.3 Small signal analysis

Capacitance and conductance simulations are calculated by performing a small signal analysis at the working point conditions (of course with the voltage (*C-V*) and frequency (*C-f*) get changed). This analysis leads to an expression for the small signal current, which is a complex number. The cell structure is then analysed as if it were a parallel connection of a (frequency dependent) capacitance and a (frequency dependent) conductance.

$$Y(\omega) = \frac{\tilde{J}}{\tilde{u}_{ext}} = j\omega C(\omega) + G(\omega) \tag{23}$$

5.2 Setting up a single shot simulation

SCAPS is able to simulate four different measurements: *I-V*; *C-V*; *C-f* and *QE*. The settings of these measurements are immediately available on the action panel, see Figure 5.5. Of course, many more user defined measurements can be simulated by exploiting the recorder facility ☺.

Figure 5.5   Measurement simulation settings

- Each of the measurement simulations will only be performed when its appropriate checkbox is checked.
- A voltage/frequency/wavelength range can be set up.
- The number of points (limited to 201) can be set up, or a voltage/wavelength increment or the number of points per frequency decade can be given.
- When a *C-V* simulation is performed you get the *I-V* simulation for free (no need to specify separately).

When performing *I-V* simulations under illuminated conditions, one is often only interested in the voltage range up to the open circuit voltage. However, before starting the simulation V_{oc} is still unknown. Hence, one can ask SCAPS to stop the *I-V* simulation under illumination as soon as the current becomes positive by checking the 'stop after Voc'-option.

5.3 Numerical parameters

All parameters contained in the Numerical Panel are listed in Table 5.1.

Table 5.1   The numerical parameters in SCAPS, with their units as displayed in the Numerical Panel, and their default values.

numerical parameter name	default value	unit
Convergence settings		
maximum number of iterations	1000	-
clamping of changes between iterations		
of electrostatic potential Φ	0.5	kT/q
of electron Fermi level E_{Fn}	0.5	kT
of hole Fermilevel E_{Fp}	0.5	kT
stop criterion: stop if all updates < stop criterion		
of electrostatic potential Φ	0.001	kT/q
of electron Fermi level E_{Fn}	0.001	kT
of hole Fermilevel E_{Fp}	0.001	kT
Mesh generator settings, see section 5.1.1		
recalculate mesh between iterations	yes	-
maximum number of mesh adaptions at each voltage	10	-
maximum ratio between neighbouring mesh points (f_{max})	1.60	-
maximum ratio between neighbouring mesh points (f_{min})	1.05	-
generation limit (G_{limit}), expressed as current density	1.0	$\mu A/cm^2$
recombination limit (R_{limit}), expressed as current density	0.001	$\mu A/cm^2$
Calculation of quantum efficiency QE, see section 6.4.7		
calculation mode	constant N_{phot}	-
monochromatic photon particle flux N_{phot} in QE calculation	3×10^{18}	$cm^{-2}s^{-1}$
monochromatic photon power flux P_{phot} in QE calculation	0.1	mW/cm^2
Defect settings, see section 3.6.3		
number of discretization levels for distributed defects	7	-
width of tail-like energy distribution in multiples of E_{char}	7.0	E_{char}
width of Gaussian energy distribution in multiples of E_{char}	6.0	E_{char}
Tunnel settings, see section 3.10.2		
minimum height of bulk tunnel barrier	2.0	kT
choice of tunnel mass	min of adjacent layers	-
allow band-to-band tunneling	no	-
allow intra-band tunneling	no	-
allow tunneling to interface defects	no	-
allow tunneling to contacts	no	-
Update of metastable occupation, see 3.8.4		-
use clamping in iteration of occupation $f_{metastable}$	no	-
maximum number of iterations	250	-
maximum relative error	0.001	-
Batch and Recorder calculations, see sections 7.4 and 8.2		
to calculate work point, start from previous batch result	yes	-
maximum V for the recording of solar cell characteristics	2.0	Volt
minimum ΔV for the recording of solar cell characteristics	0.02	Volt

Since SCAPS 3.3.02, version august 2015, the numerical parameters of Table 5.1 can be set back to their default values by clicking a button in the Numerical Panel. When saving a problem as an all SCAPS file

(extension .scaps), or as a .def file from within a script, all numerical parameters are saved. When saving as a .def file in interactive mode, the numerical parameters are only saved when their value differs from the default.

5.4 Numerical limitations

Even though many people believe computers to be perfect when performing calculations, they are not!! Every computer has its machine accuracy, which is defined as the smallest (in magnitude) number which, when added to 1.0, produces a floating-point result different from 1.0. This accuracy usually is a small number ($< 10^{-15}$), so that in a pragmatic view one can consider it "accurate enough and close to perfection". Even though this view is commonly adopted, it is branded as "naïve" and plain "fiction" in [7]. For the simulation of semiconductors, this accuracy is indeed often insufficient, and care should be taken to avoid numerically unstable operations. SCAPS does take care, but sometimes things can still go wrong. Below you can find a list of phenomena which should set of an alarm in your head indicating a numerical problem, and tips and tricks to avoid them and to avoid convergence failures:

- A negative capacitance or a capacitance increasing with increasing frequency is often not physical. Though exceptions exist, most of the time this behaviour is a result of a numerical error when calculating the capacitance from the small signal current. This occurs when the imaginary part of the small signal current is (in magnitude) much smaller/larger than the real part. This might be the case when large voltage biases or strong illumination conditions are applied.

- The current through the sample (visible in the lower left corner of the EB-panel, in green) should be constant through the sample. SCAPS is optimized to calculate *pn*-junctions (*p* to the left, *n* to the right) and is much less stable for *np*-junctions. IT IS HENCE STRONGLY RECOMMENDED TO KEEP THE *P*-SIDE OF YOUR STRUCTURE TO THE LEFT AND THE *N*-SIDE TO THE RIGHT. *np*-junctions tend to lead to non-uniform currents through the cell, which is unphysical.

- Keep the variation between different calculation steps limited to avoid divergence. The reason for this should be obvious when you have read §5.1.2.

- Keep it realistic! SCAPS is developed and tested to simulate realistic situations, hence things can go wrong when simulating unphysical situations.

- Don't overdo! Do you really want to know the current at $T = 2$ K and $V = 3$ kV? Charge carrier densities scale exponentially with $1/k_B T$. Hence, the stability of the algorithms exponentially decreases with decreasing temperature, which initially leads to a higher number of iterations needed to get to convergence and at even colder temperatures to an inescapable divergence.

Chapter 6: Result analysis

SCAPS performs a lot of calculations any time you click the 'calculate'-button. Your job as a user is to analyse those results. Fortunately there is a lot of help. The analysis-panels can easily be accesed from the action panel or any other analysis panel, see Figure 6.1.

6.1 Navigating to the analysis

The analysis-panels can easily be accesed from the action panel or any other analysis panel, see Figure 6.1.
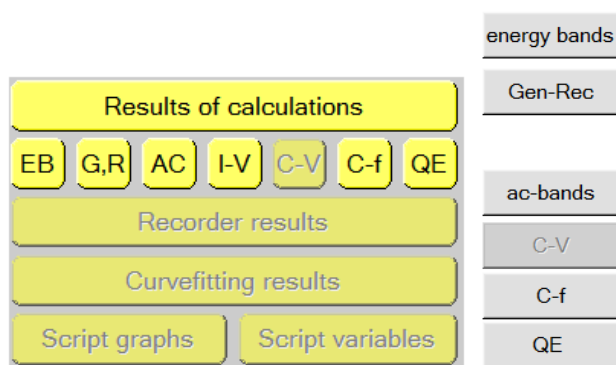
Figure 6.1   Navigating to the resuls from the action panel (*left*) or any other panel (*right*)

Several options are available on every panel: saving data, showing data, saving graphs and plotting the panel (sending to a printer). There are options available for scaling and zooming of graphs and to show more info about the plotted curves. Other options are panel specific.

At the bottom of every panel there are two comment windows. The left window is auto-generated and gives the definition file used with its last saving and the when the simulation was performed. The right window can be used to write personal comments.

6.2 Zooming and scaling

Most of the graphs are scalable. For most axes a logarithmic or linear scale can be chosen and/or the absolute value of the property can be plotted. The axis-range can be set by clicking a 'scale'-button, which opens the following pop-up menu.
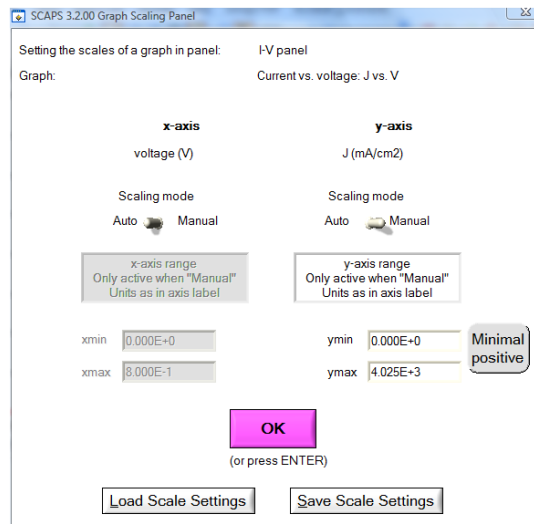
Figure 6.2   The Scaling panel

- The scaling of both axis can be set automatically (default) or manually
- A minimum/maximum value can be given
- The settings on this panel can be saved/loaded (generating an ASCII-txt-file)
- Clicking the minimal positive button sets the y-axis scaling to manual and sets ymin to the smallest positive datapoint which needs to be plotted. This might be very useful when there are negative datapoints in a semi-logarithmic diagram.

(Almost) all diagrams have moreover zooming facilities. In order to zoom in you should hold the CTRL-button pressed whilst drawing a rectangle with the mouse on a graph. For zooming out you should hold the CTRL-button pressed whilst right-clicking with the mouse. Resetting the initial zooming can often be done as well by changing the log/lin property of the axis. This only works if the axis scaling is set to auto.

6.3 Curve info and legend

Performing several simulations, the graphs can get quite crowded (unless you click 'clear all simulations' before every simulation). Hence, facilities are provided to get more info about the simulated curves. When the curve info option (top right corner of any panel) is switched on, any click on a graph will render a pop-up menu with information about the graph, curve and point you clicked on. The point/curve closest to the position where you clicked will be selected.

It is also possible to interactively hide and show curves on graphs. This option is available by right-clicking on the ___plot/legend___ -button (left-clicking leads to plotting the entire panel). A pop-up window is opened which gives a list of all available curves (with a small legend). (Un)checking items in the list leads to hiding/showing the curve on the panel.

6.4 Measurement specific options

6.4.1 The energy band panel

This panel shows the band diagram, the carrier densities, the current densities and the occupation of defects of the last calculated work point. When no measurement is selected or a *QE* or *C-f* measurement is selected it displays thus the working point conditions as set on the action panel (or in the batch). If a *C-V* or *I-V* simulation is performed it displays the conditions of the last calculated voltage point. These conditions are mentioned on the bottom left part of the panel.

The panel is displayed during calculations, as such it provides a small movie of how the bands evolve during simulation. When your computer is too fast, you might want to slow down this movie. This is possible by checking the 'pause at each step'-option on the action panel, see Figure 5.5. This way the program will pause at every calculation step and you can move forward by clicking the continue button (on the EB-panel or the action panel) or abort by clicking the stop-button (on the action panel).

The scaling of the *x*-axis on all graphs of the EB-panel is the same. Hence, unless using the zooming option, all graphs display the same part of the structure you simulated.

The defect level energy of the defects present and the transition energy of metastable defects are indicated on the band diagram graph. When the defect has an energy distribution (e.g. Gauß or CB tail), the energy E_t as set in the problem definition is displayed. This energy can of course be graded. The colour legend used for displaying the defect levels can be chosen either grouped according to the defect type (e.g. single acceptor, amphoteric,…) or grouped according to the charge state of the defect level (e.g. 0/-, 2+,+,…).

On the occupation probability graph the defect level occupation is displayed. One can choose whether to display the occupation with electrons (default) or holes. For a metastable defect transition the 'occupation with electrons' represents the fraction of defects in the acceptor configuration and the 'occupation with holes' the fraction of defects in the donor configuration. Just as for the band diagram graph a colour legend can be selected according to defect type or defect charge states. Additionally, a colour code according to the charge of the states can be chosen. In this case the fraction of defects in a specific charge state are displayed. When this option is selected, neutral defects and metastable transitions are not displayed however.

When the defect has an energy distribution (uniform, Gauß, tail) the occupation of several (typically 7) sublevels in this distribution is shown, and the occupation of outermost levels of the distribution, considered in the calculation, is drawn in a thicker line. You will observe that this occupation diagram can get quite complicated and not straightforward to interpret. The use of the Curve Info feature (Section 6.3) is highly recommended in interpreting the occupation graphs: click on a graph, and Curve Info shows you what this graph represents.

6.4.2 The generation-recombination panel

A more detailed view of the defect occupation and of the recombination and generation can be found on the generation-recombination panel. On this panel several recombination-generation processes can be show by checking the appropriate checkboxes. An overview of the available defect(level)s is given in a tree structure. (Un)checking them will make the corresponding curves (un)visible. A graphical view of the simulated structure or a more detailed overview of the simulation conditions can be shown by switching the
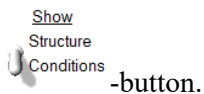
 -button.

6.4.3 The IV-panel

The results of the current-voltage simulations are shown on the IV-panel. The left graph displays all *I-V* simulations. The right graph gives detailed information about the recombination currents in the last simulation. This allows to see the main recombination mechanism in the structure for varying voltages.

If the simulation is performed under illumination, the solar cell parameters are calculated and shown, see Figure 6.3. If SCAPS needed to perform an extrapolation to determine these parameters, e.g. because the simulation range was too narrow, the warning LED below the parameter turns red. As most users set the voltage range from $V_{start} = 0$ to some V_{stop}, the need for extrapolation will first show up for V_{oc} and thus *FF* (e.g. if $V_{stop} < V_{oc}$, when the references for voltage are such that both are positive numbers). When the voltage is more restricted, $V_{stop} < V_{mpp}$ (mpp is the maximum power point), then also extrapolation is needed to determine the efficiency η. When V_{start} was set to 0 (the default value), J_{sc} will be determined precisely (no interpolation or extrapolation needed). When extrapolation was needed, and when SCAPS deems the result not physical, both the extrapolated (but unreliable) parameter and the warning LED are not shown. This is

e.g. the case if SCAPS would find $V_{oc} < 0$ or $V_{oc} > 10$ Volt (we do not believe in supermegagigantic good solar cells…). So, if some of your solar cells parameters do not show up (e.g. you see J_{sc} and η but not V_{oc} and *FF*), just extend your voltage range to include V_{oc}, and do not immediately start to harass us with e-mails about 'malfunctions' in SCAPS[1].

As already mentioned, when the generation was 'from file', no efficiency can be calculated, but the collection efficiency is given instead.
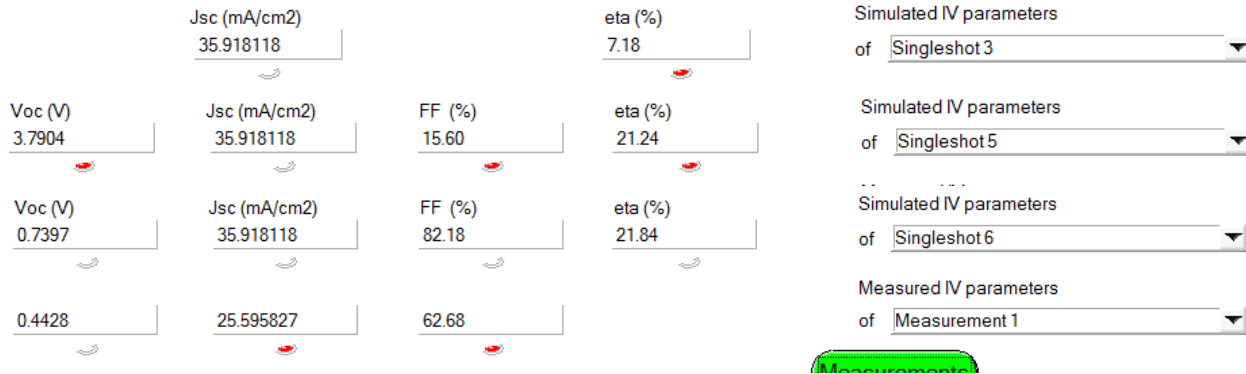
Figure 6.3   Some examples of visualization of the solar cell parameters V_{oc}, J_{sc}, *FF* and η. The three upper lines are for a simulation of `simple pn.def` with $V_{start} = 0$ and $V_{stop} = 0.2$ V, 0.6 V and 0.8 V. The bottom line is for a measurement.

The *I-V* graph is by default directly given as an *I(V)* graph, but the current density *J* can also be scaled or subtracted automatically with the short circuit current by changing the current mode on top of the graph (Figure 6.4). This enables you e.g to perform analyses as discussed in [16] or to study an illuminated *I-V* - curve with a logarithmic scaling (this is also possible by clicking the 'absolute' checkboxes next to the *V* and *I* axes).
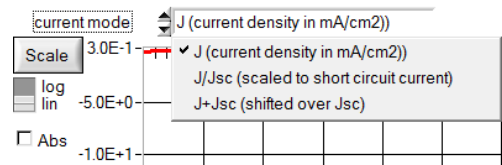
Figure 6.4   Options to display the *I-V* curves: full custom scaling; fast buttons for linear/logarithmic and absolute value/algebraic value; options to scale (with J_{sc}) or shift (over J_{sc}) all *J-V* curves.

6.4.4 The ac-panel

This panel gives an overview of the small signal current (left side of the panel) and of the small signal variations in potential and quasi-Fermi levels (right side of the panel). These properties are complex numbers. The top graphs give the amplitude, the bottom graphs the phase.

6.4.5 The CV-panel

This panel displays the capacitance and conductance as a function of applied voltage. Moreover, a Mott-Schottky diagram and the apparent doping density are calculated. In order to calculate this apparent doping density profile, a numerical differentiation of the data is needed together with a choice of the relative permittivity. These options can be set by clicking the 'Analysis method'-button, which opens the 'Admittance (capacitance) analysis panel' which is discussed in §6.4.5.1.

---

[1] In the first months of 2019 a real avalanche of such help-cries: "help! V_{oc} and *FF* do not show up: what is wrong with SCAPS?" arrived at M.B.'s desk. All of them were by young researchers simulating perovskite solar cells with $V_{oc} > 1.0$ V, with the standard voltage range setting $V_{start} = 0$ and $V_{stop} = 0.8$ V. All of these help requests could have been avoided by simple inspection of the *IV* curve and a little bit of common sense — or by reading this recent addition to the SCAPS Manual (Marc B., 25-3-2019).

6.4.5.1 <u>Admittance (capacitance) analysis panel</u>

This panel allows to set the dielectric permittivity and the parameters of the numerical differentiation used in the calculation of the apparent doping density profile and the AS-analysis (§6.4.6). It also allows an automatic calculation of the attempt-to-escape frequency on the AS-panel. The graph of the apparent doping density profile is not immediately adapted to a change of these parameters. The new parameters are only valid for new simulations. The parameters can however be immediately updated in the AS-analysis as there is a 'recalculate'-button on the AS-panel.

Figure 6.5   The Admittance (capacitance) analysis panel

Only relative dielectric permittivity values on the left/right side of the layers can be chosen by clicking any of the select buttons on the panel.

The layer which is chosen here determines automatically the choice of the layer-parameters (density of states and thermal velocity) that are used to calculate an attempt-to-escape frequency v_0. This frequency is calculated as (24) when the switch is set to electrons or as (25) when it is set to holes. The capture cross section is determined by choosing a defect(level).

$$v_0 = \frac{\xi_0 T^2}{2} = \sigma_n v_{th,n} N_C \tag{24}$$

$$v_0 = \frac{\xi_0 T^2}{2} = \sigma_p v_{th,p} N_V \tag{25}$$

The numerical derivative is calculated by fitting either a best straight line or a best parabola through a point and its neighbours. The number of neighbours which have to be taken into account and the order of the polynomial (straight/parabola) in the middle and at the edges of the simulation/measurement range can be chosen.

6.4.6 The Cf-panel

This panel displays the capacitance and conductance as a function of simulation frequency. Moreover, a Nyquist plot and a plot of G/ω is given. A facility has been added to analyse the results according to the admittance spectroscopy (AS) method outlined in [17] and extended in [18]. By clicking the 'Admittance spectroscopy'- button the AS-panel is displayed, see Figure 6.6.
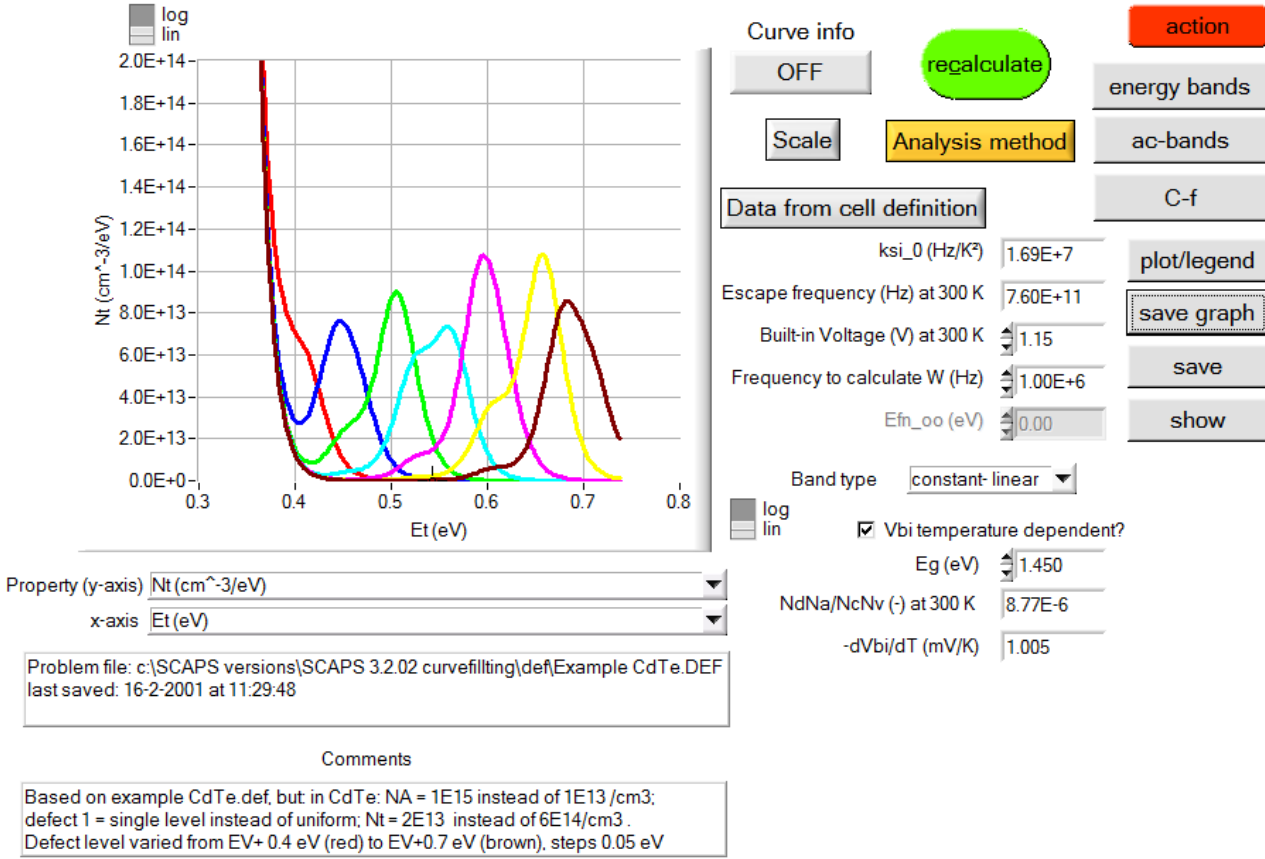
Figure 6.6   The AS panel

The terminology and parameters used in this panel are those defined in [17, 18]. You can select several properties as abscissa and ordinate
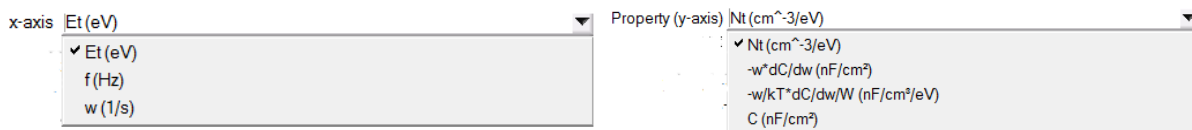
Figure 6.7   AS panel: abscissa (left) and ordinate (right) choices

The depletion width used in the formulas is immediately determined from the capacitance value at a specified frequency. The dielectric permittivity used to calculate this depletion width and the parameters governing the numerical differentiation can be specified by clicking the 'Analysis method'-button, which opens the 'Admittance (capacitance) analysis panel', see §6.4.5.1. When clicking the 'Data from cell definition'-button the attempt-to-escape frequency is calculated from the data set on the 'Admittance (capacitance) analysis panel' as well. Its value can however afterwards be changed again. The built-in voltage which is needed for the calculation of N_t can be chosen temperature dependent according to (26).

$$V_{bi} = E_g + k_B T \ln\left(\frac{N_A N_D}{N_C N_V}\right)$$

(26)

6.4.7 The QE-panel

This panel allows to analyse the *QE* simulations. Specifically, the meaning of *QE* in SCAPS is *external* quantum efficiency *QE*: the number of electrons leaving the cell (ass current) divided by the total number of photons incident on the cell.

On the horizontal axis one can display either the wavelength or the photon energy of the monochrome light. On the vertical axis one has a wider choice:
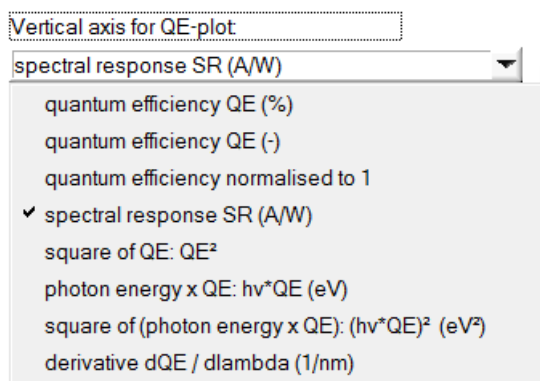
Figure 6.8   Ordinate choices on the QE-panel

The fourth option (spectral response *SR*) is a recent addition (august 2017) to serve users that want to simulate photodetectors. *SR* is defined as the short circuit current divided by the total incident light power, and thus is in A/W (this is also mA/cm^2 divided by mW/cm^2, should you insist). Note that $SR = QE/h\nu$ when *SR* is in A/W, *QE* dimensionless and $h\nu$ in eV. The four last options are sometimes used to derive a band gap value from *QE* measurements or simulations.

The *QE* is calculated by comparing the current at the workingpoint conditions and the current when adding an additional amount of monochromatic photons. This number of photons can be set on the numerical panel, Figure 6.9. When both currents are large compared to their difference, it is possible that your computer makes a numerical error in the subtraction leading to a non-physical *QE* which is negative or larger than 100 %.
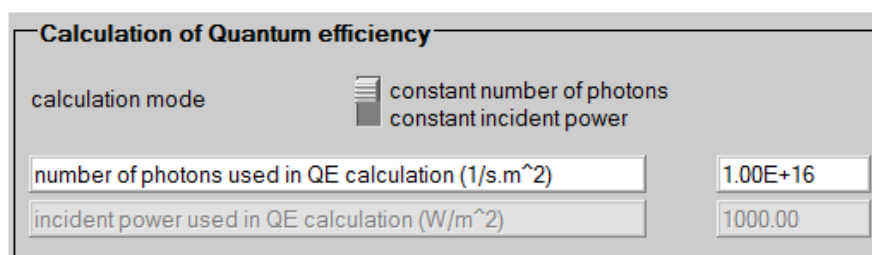
Figure 6.9   Setting the number of photons for the *QE* simulation in the Numerical Panel.

In SCAPS 3.3.02, version august 2015, the defaults values displayed in standard units of the SCAPS panels (thus: cm, mA, mW-based), and were changed to P_{phot} = 0.1 mW/cm^2, corresponding to 0.001 sun, and to 3×10^{18} /cm^{2}s, that is for (λ = 620 nm, $h\nu$ = 2 eV) photons, also roughly corresponding to 0.001 sun. These default values could be representative for actual *QE* measurements with a monochromator set-up.

6.5 Managing measurement data

SCAPS provides facilities to compare simulation results with measurement data. Measurement data can be loaded by clicking the Measurements -button, which redirects to the 'Manage measurements panel', Figure 6.10.

6.5.1 The Manage measurements panel

This panel shows a list of the measurement files which are currently loaded. Additional measurements can be added, measurements can be removed or replaced and the order of the list can be changed. By (un)checking items in the list you can control which measurements have to be displayed in the graphs. When selecting an item in the list, a summary of its properties are displayed on top of the panel. This allows to check whether you have loaded the measurement you intended and whether the working point conditions of the file have been read correctly.
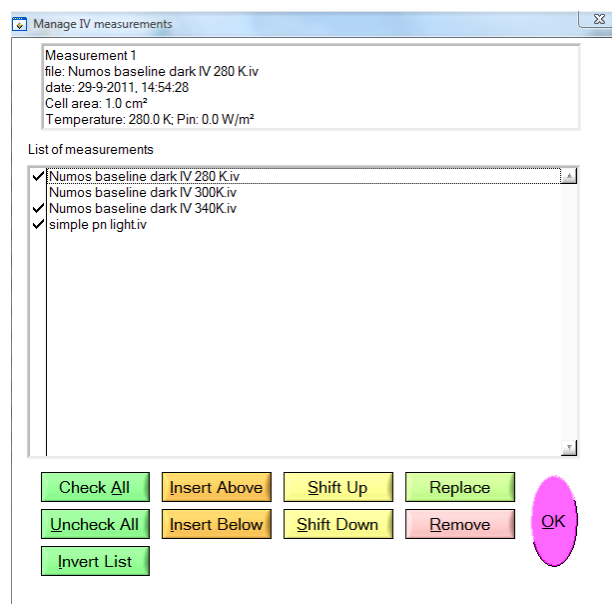
Figure 6.10   Manage measurements panel

6.5.2 Structure of a measurement file

Any ASCII-text file can be read as a measurement file. In particular, any of the simulation results which have been saved by SCAPS can be read as a measurement file. So, if you feel unsure whether your file will be read correctly, you can try to start from a SCAPS-results file.

The file extension indicates which kind of measurement it contains. The allowed extensions are '.iv', '.cv', '.cf' and '.qe'.

A measurement file is read until the first line which can be interpreted as measurement data is found. This is a line starting with at least two numeric values. All lines above are considered to be part of the header (which might be mere comments or contain information about the working point conditions) All lines below the first data line are considered to contain measurement data. As a result, any line which can not be interpreted as data which occurs below the first data line will be discarded and can thus not be interpreted as working point conditions. Hence, when more than one measurement is contained in a file only the working point conditions of the first measurement will be read, even though the data points of all measurements will be loaded!

6.5.2.1 The workingpoint conditions

SCAPS will try to interpret the information in the header of the file as the working point conditions of the file. When any of these words is found <u>at the beginning of a line</u>, the numerical value which follows is saved. None of these words is case-sensitive. An overview of the code-words used is listed below.

Table 6.1   Workingpoint code words

code-word(s)	unit	Default value	remarks
area; cell area; cellarea;	cm^2	1.0	See below

temperature; temp	K	300	
bias	V	0.0	Only for *C-f* and QE
frequency; freq; frekw	Hz	10^6	Only for *C-V*
Incident power; Pin; P in; P_in Incident light; Light power	mW/cm^2	100	Only for *I-V* and QE
dark	#	*light*	Whenever this word is found, the incident power is set to zero Only for *I-V* and QE
Units			See §6.5.2.2

- The area is used to scale the current, capacitance and conductance values to an area of 1 cm^2.

- Alternative units for the area are supported. Adding any of the words ["cm2", "cm^2", "cm^2"]; ["mm2", "mm^2", "mm^2"] or ["m2", "m^2", "m^2"] to the area statement will change the unit of the area read to cm^2; mm^2 or m^2. If none of these words are found the unit of the area is cm^2.

- In a similar way as for the area, different units for the incident power are supported: mW/cm2; mW/cm^2; mW/cm^2; W/m2; W/m^2; W/m^2; mW/m2; mW/m^2; mW/m^2; W/cm2; W/cm^2; W/cm^2.

- The temperature and bias voltage are only used to calculate admittance spectra

- The incident power is used to calculate the efficiency.

- When the incident power is zero (or when the code-word dark has been found) the *I-V* parameters (thus: η, J_{sc}, V_{oc} and *FF*) will not be calculated.

6.5.2.2 The measurement data

When a first data line is found, all subsequent lines will be interpreted as data lines. Up to five numeric values will be read. The meaning of these numbers varies according to the measurement type as listed below.

Table 6.2   Data in the columns of a measurement file

measurement	#1	#2	#3	#4	#5
I-V	*V* (V)	*I* (mA)*			
C-V	*V* (V)	*C* (nF)*	*G* (S)*	*W* (μm)	$N_{apparent}$ (cm^{-3})
C-f	*f* (Hz)	*C* (nF)*	*G* (S)*		
QE	*λ* (nm)	*QE* (%)			

- The properties marked with a * will be scaled with the area, when no area has been read, an area of 1 cm^2 is assumed.

- The depletion width *W* and the apparent doping density $N_{apparent}$ can be read from the measurement-file, but they can also be calculated from *C* and *V* by SCAPS. This choice is governed by a switch button on the *capacitance analysis panel*.

Measurement data:

☐ nW calculated by SCAPS
▨ nW read from file
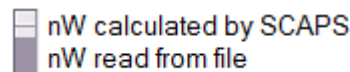
   The default units for the measurement data are given in the table. It is however possible to change them. In order to do this, a line starting with the code-word "units" has to be added in the header. Next to this code-word a list of properties with their unit can be specified.

   Examples: units: I: A   V: mV

              Units: C: pF   V: V   G: mS

   A list of property-names which can be changed for every measurement type are given in the Table below:

I-V	V:	I:	J:		
C-V	V:	C:	G:	W:	N:
C-f	f:	C:	G:		
QE	lambda:	QE:			

Pay attention! The semicolon ':' is obligatory.

6.6 Saving results

When clicking the save or show button, the following properties are saved in an ASCII file which can be read using e.g. notepad or MS Excel. More options to keep track of and save properties are available in the recorder facility, see Chapter 8.

Table 6.3   Overview of the properties which can be saved on the EB panel

property	remark
x(μm)	
Ec(eV)	Conduction band energy
Fn(eV)	Quasi-Fermi level for electrons
Fp(eV)	Quasi-Fermi level for holes
Ev(eV)	Valence band energy
n(/cm3)	Free electron density
p(/cm3)	Free hole density
rho(defect) (/cm3)	Charge density in defects
net doping (/cm3)	$N_D - N_A$
rho(/cm3)	$p - n + N_D - N_A$ + charge in defects
E(V/cm)	Electric field
jn(mA/cm2)	Electron current density
jp(mA/cm2)	Hole current density
jn_tunnel(mA/cm2)	Electron tunnel current density
jp_tunnel(mA/cm2)	Hole tunnel current density
jtot(mA/cm2)	Total current density
generation(#/cm3.s)	Generation rate
recombination(#/cm3.s)	Recombination rate
cumulative generation (mA/cm2)	Cumulative generation rate (summation from the side where the light is incident, to the opposite side)
cumulative recombination (mA/cm2)	Cumulative recombination rate (once with summation from left to right, and once summed from right to left). Also interface and contact recombination taken into account
jn1(mA/cm2)	Interface recombination current for electrons (left side of the interface)
jn2(mA/cm2)	Interface recombination current for electrons (right side of the interface)
jp1(mA/cm2)	Interface recombination current for holes (left side of the interface)
jp2(mA/cm2)	Interface recombination current for holes (right side of the interface)
jn1[with tunnel](mA/cm2)	Same as jn1 but now with tunneling
jn2[with tunnel](mA/cm2)	Same as jn2 but now with tunnelling
jp1[with tunnel](mA/cm2)	Same as jp1 but now with tunnelling
jp2[with tunnel](mA/cm2)	Same as jp2 but now with tunneling
js(mA/cm2)	Total surface recombination current density
pi(C/cm2)	Surface charge density

A note on "cumulative" generation and recombination

The cumulative generation is always calculated starting from the side where the light is incident. Thus:

$$G_{\text{cumulative}}(x) = \int_0^x G(x')\,dx' \qquad \text{illumination from left, thus:}$$

$$G_{\text{cumulative}}(0) = 0 \tag{27}$$

$$G_{\text{cumulative}}(d) = G_{\text{total}}$$

or

$$G_{\text{cumulative}}(x) = \int_x^d G(x')\,dx' \qquad \text{illumination from right, thus:}$$

$$G_{\text{cumulative}}(0) = G_{\text{total}} \tag{28}$$

$$G_{\text{cumulative}}(d) = 0$$

The cumulative recombination is offered in two directions of summation (integration): from left to right, and from right to left:

$$R_{\text{cumulative,L}\to\text{R}}(x) = \int_{0^-}^x R(x')\,dx' \qquad R_{\text{cumulative,R}\to\text{L}}(x) = \int_x^{d^+} R(x')\,dx' \tag{29}$$

The notation 0^- means that the left contact is included, and d^+ that the right contact is included. At an interface, $R_{\text{cumulative}}$ takes a discontinuity (jump) equal to the interface recombination. With these definitions,

$$R_{\text{cumulative,L}\to\text{R}}(0) = R_{\text{left contact}} \qquad R_{\text{cumulative,L}\to\text{R}}(d) = R_{\text{total}}$$

$$R_{\text{cumulative,R}\to\text{L}}(0) = R_{\text{total}} \qquad R_{\text{cumulative,R}\to\text{L}}(d) = R_{\text{right contact}} \tag{30}$$

Table 6.4 Overview of the properties which can be saved on the Generation recombination panel when saving the generation data. All properties are split up per layer, per defect in this layer and per charge state in this defect. (first index = layer, second index = mechanism, third index = defectlevel (0 is the most positively charged and hence the closest to the valence band))

property	remark
x (µm)	
Total recombination (#/cm3.s)	Total recombination rate
SRH recombination (#/cm3.s)	Shockley-Read-Hall recombination rate (at defects)
Geh (#/cm3.s)	Direct band-to-band excitation of electron-hole pairs by light (this equals zero in dark)
Radiative recombination (#/cm3.s)	Radiative recombination rate
Auger recombination (#/cm3.s)	Auger recombination rate
IPV Gn	Gross IPV electron effect: direct optical excitation from defect to conduction band
net Gn	Net electron generation: IPV Gn + thermal generation - thermal recombination
Gnth	Thermal electron generation (thermal emission of electron from defect to conduction band)
Rnth	Thermal electron recombination (capture by defect of conduction band electron)

IPV Gp	Gross IPV hole effect: direct optical excitation from defect to valence band
net Gp	Net hole generation: IPV Gp + thermal generation - thermal recombination
Gpth	Thermal hole generation (thermal emission of hole from defect to valence band)
Rpth	Thermal hole recombination (capture by defect of valence band hole)

Table 6.5   Overview of the properties which can be saved on the IV-panel.

property	remark
v(V)	voltage
jtot(mA/cm2)	Total current density
jbulk(mA/cm2)	Bulk recombination current density
jifr(mA/cm2)	Interface recombination current density
js_n(mA/cm2)	Back contact recombination current density
js_p(mA/cm2)	Front contact recombination current density
j_SRH(mA/cm2)	Defect recombination current density
j_Radiative(mA/cm2)	Radiative recombination current density
j_Auger(mA/cm2)	Auger recombination current density
Voc (V)	Open circuit voltage (only under illumination)
Jsc (mA/cm2)	Short circuit current density (only under illumination)
FF (%)	Fill factor (only under illumination)
eta (%)	Efficiency (only under illumination)
V_MPP (V)	Voltage at the maximum power point (only under illumination)
J_MPP (mA/cm2)	Current density at the maximum power point (only under illumination)

Table 6.6   Overview of the properties which can be saved on the AC-panel.

property	remark
x (μm)	
jn.re(mA/cm2)	Real part of the small signal electron current density
jn.im(mA/cm2)	Imaginary part of the small signal electron current density
jp.re(mA/cm2)	Real part of the small signal hole current density
jp.im(mA/cm2)	Imaginary part of the small signal hole current density
jdispl.re(mA/cm2)	Real part of the small signal displacement current density
jdispl.im(mA/cm2)	Imaginary part of the small signal displacement current density
j.re (mA/cm2)	Real part of the small signal current density
j.im (mA/cm2)	Imaginary part of the small signal current density
psi.re(#kT/q)	Real part of the small signal potential
psi.im (#kT/q)	Imaginary part of the small signal potential
Fn.re (#kT)	Real part of the small signal electron quasi-Fermi level
Fn.im (#kT)	Imaginary part of the small signal electron quasi-Fermi level
Fp.re (#kT)	Real part of the small signal hole quasi-Fermi level
Fp.im (#kT)	Imaginary part of the small signal hole quasi-Fermi level

Table 6.7   Overview of the properties which can be saved on the CV-panel.

property	remark

v(V)	voltage
C(nF/cm2)	Capacitance
G(S/cm2)	Conductance
W(μm)	Depletion width
Napp(/cm3)	Apparent doping density
jtot(mA/cm2)	Total current density
jbulk(mA/cm2)	Bulk recombination current density
jifr(mA/cm2)	Interface recombination current density
jopp_n(mA/cm2)	Back contact recombination current density
jopp_p(mA/cm2)	Front contact recombination current density

Table 6.8 Overview of the properties which can be saved on the Cf-panel.

property	remark
f(Hz)	Frequency
C(nF/cm2)	Capacitance
G(S/cm2)	Conductance
Z.re(ohm.cm2)	Real part of the impedance
Z.im(ohm.cm2)	Imaginary part of the impedance
Z.magn(ohm.cm2)	Magnitude of the impedance
Z.phase(°)	Phase angle of the impedance

Table 6.9 Overview of the properties which can be saved on the AS-panel.

property	remark
f(Hz)	Frequency
C(nF/cm2)	Capacitance
Et(eV)	Activation energy
-w*dC/dw(nF/cm²)	Scaled derivative (see [18])
Nt(cm^-3/eV)	Calculated defect density

Table 6.10 Overview of the properties which can be saved on the QE-panel.

property	remark
lambda(nm)	Wavelength
QE(%)	Quantum efficiency
photon energy (eV)	Photon energy

Chapter 7: Batch calculations

When you want to explore the influence of one or a few parameters to the solar cell characteristics, you can take profit of the batch option. When you click 'Batch set-up', a panel opens where you can choose which parameter to vary, over which range, and in which mode (Lin, Log or custom). You can also define more than one parameter, and vary all of them (in a nested way or 'simultaneous'). Now, up to nine batch parameters can be defined, but be modest to start. A batch calculation is launched when 'calculate: batch' is clicked. After a batch simulation all parameters on the panels are reset as they were before the calculation.

7.1 The batch set-up panel

The batch set-up panel allows you to vary up to nine different parameters, when you want to vary more you can also vary entire definition files (§7.2) or exploit the script facilities (Chapter 10). An example of the batch set-up panel is shown in

Figure 7.1   The batch set-up panel, illustrating most of the options

- All (most of the) parameters present in the currently defined structure can be varied. Also working point conditions can be varied.

- When checking 'simultaneous', this parameter will be varied together with the parameter above (and thus in the same number of steps). If 'simultaneous' is not checked, the parameters are varied in a nested way.

- Parameters can be varied in a linear, a logarithmic or a custom defined way.

- Parameters which are files can be varied by entering a list of file names: generation files, spectrum files, filter files, grading files, optical capture corss-section files, initial state workpoint files.

- Some parameters can only take two values (on/off), e.g. illumination.

- When changing any of the illumination parameters (ND filter, spectrum file…) as a batch parameter, the illumination is automatically switched on.

- When changing one of the effective mass parameters, the accompanying tunnelling mechanism is automatically switched on.

- It is possible to save/load the parameters on the batch panel in a '*.sbf'-file which is a standard ASCII-file.

- It is possible to print the batch panel in order to remember your settings.

- [SCAPS 3.3.04, october 2016] It is possible to suppress (or not) the updating of the screen (drawing of all curves in the Energy Bands panel) during batch execution: check the appropriate box in the Batch Set-up Panel. This is speeding up the batch execution somewhat, though less than we hoped for ☹. This option can also be set in a script, with a `set batch_display_mode command` (see Section 10.4.6).
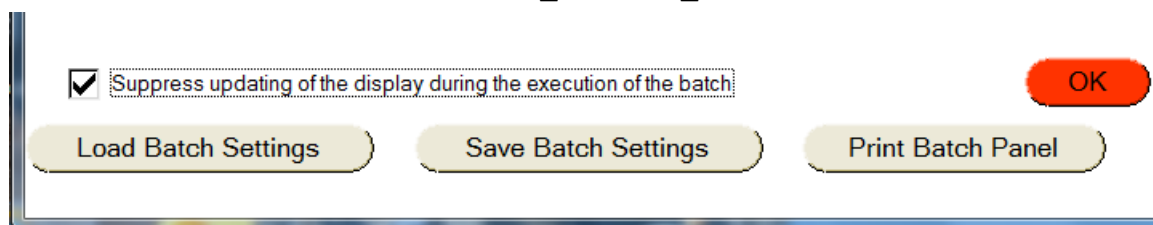
7.1.2 Custom defined values

When checking the custom list option, a Set-button appears which when clicking it opens a panel which allows you to give a enter a list of parameter values.
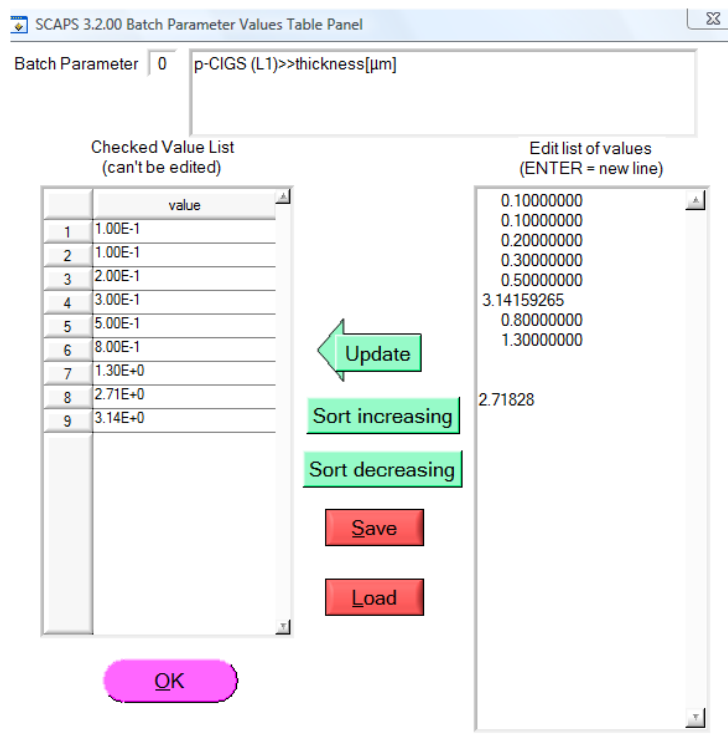
Figure 7.2   Setting custom batch values

- Parameters can be typed or copy/pasted in the right list interactively. Afterwards you should press 'Update' to allow SCAPS to interpret your typing work. Data which can not be interpreted as a number will be ignored.

- The values in the left list are the values SCAPS will use. These can be sorted.

- Parameter value lists can be saved and loaded. The resulting files are standard ASCII files with extension '*.bdf'. This allows you to make custom lists with any other program (MS Excel, Origin, Matlab…) and load them in SCAPS. The layout of this file is rather tolerant. You just add the parameter values one below the other. All lines which can not be interpreted as only one number are ignored as being

comment. We strongly recommend the user to add some comments to the file, so that she/he remembers what data are present in the file, even after not using it for a while.

Some parameters are files rather than numbers, e.g. a spectrum file. Then you have no choice but varying the parameter in a custom way. Clicking the Set-button will open a panel where you can set a list of files. Files can be added/removed/replaced. The order of the file-list can be changed. The list of files can be saved/loaded as well. This is again a standard ASCII files with extension '*.bdf'. In the bdf-file you then have to list the filenames of these files preceded by "file:". All lines in the bdf-file which are not preceded by this will be treated as comment. The listed files should be placed in the appropriate SCAPS folder: For example, the spectra and the generationfiles which you list should be placed in the "spectrum" and "generation"-file of your SCAPS-folder respectively.

7.2 Varying entire definition files

SCAPS offers the opportunity to vary entire definition-files in a batch calculation. This option is activated by checking the 'vary definition files'-checkbox. A list of files can then be set by clicking the 'edit/load list of definition files'-button. This list of files is treated in a similar way as any other parameter which has a file-nature rather than a numeric-nature, see §7.1.2.

When definition files are varied in a batch, only those parameters which are present in ALL of these definition files can be varied further as a batch parameter.

7.3 Varying parameters of the initial state work point

There are two ways to vary the initial workpoint variables in the batch set-up: all together, by specifying a list of .wp2 files (Figure 7.3), or each parameter separately (Figure 7.4).
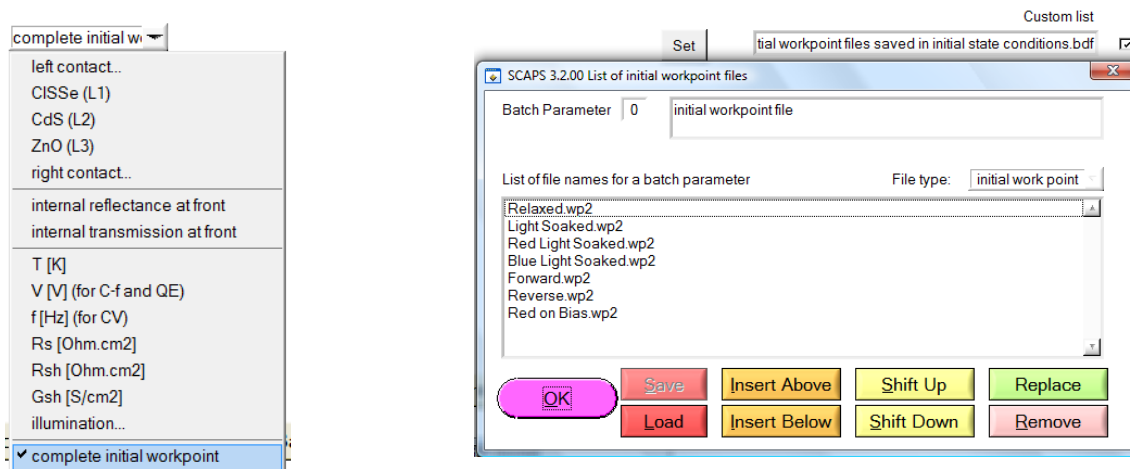
Figure 7.3   Selecting all parameters of the initial state workpoint as a batch parameters; al list of .wp2 files has to be specified, where each .wp2 file was made by saving the initial state workpoint in a file.
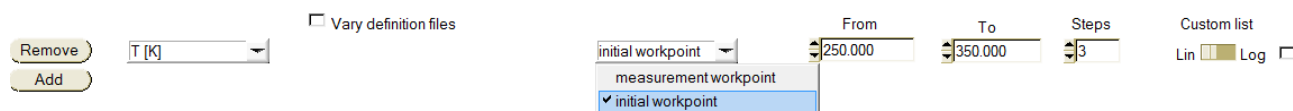
Figure 7.4   Selecting a single parameter of the initial workingpoint as a batch parameter.

7.4 During calculation…

It might happen that you launched a (big) batch calculation, and all of a sudden you realize you made an error in the set-up. Instead of waiting until all calculations are performed or instead of aborting SCAPS, you can interrupt the batch calculation by keeping the SHIFT-button pressed. The calculation will then stop (but only after it has finished the calculation of the current parameter value)

You might want to launch a big batch calculation overnight, but when you return next morning, you see that the calculation stopped at step 3 due to a convergence error ☹ which gives a pop-up window that should be acknowledged. Inventive users have used pieces of wood to keep the ENTER-button pressed (which solved the problem ☺), but there exists a better way out. On the numerical panel you can ask SCAPS to write error messages to a log-file rather than to inform on the screen.

| Convergence failure messages | Inform on screen and wait for ackowledgm ▼ |
| Output list after a convergence | Truncate output list   ▼ |
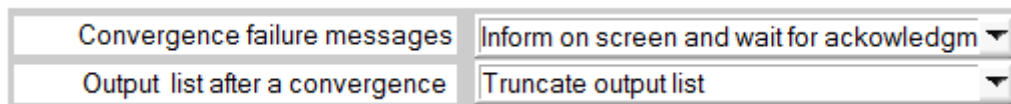
Figure 7.5   Error handling on the numerical panel

The batch facility is a powerful tool, but it also enables you to let things go wrong. Be aware that you are more able to set irrealistic parameter values. For example you can set a layer width equal to zero (which will abort SCAPS ☹). When varying files, take care that all files exist in the correct folders.

Every new calculation in a batch starts a the starting point, as explained in §5.1.2, this can however be changed to speed up the calculations. Do take care however to add the voltage/frequency/resistance as the last (bottom-most) parameter in the batch then.

Chapter 8: Recorder calculations

In a regular single shot or batch calculation, the detailed panels are only available for the last measurement point. To be able to see them as a function of the batch parameters you can launch a record calculation. You should first select the properties which you want to keep track of by clicking 'Record set-up'. By clicking 'calculate: recorder', a recorder calculation is launched. Cell parameters are varied according to the Batch set-up, and all simulations (and only those) are performed which are needed to determine the asked properties. This means the selected measurements on the action panel are ignored!

If this option is used with a little bit of imagination, you can device all kind of measurement simulations with SCAPS.

8.1 Setting a recorder

With the Recorder facility, you can do a batch calculation and register or record selected cell properties as a function of the batch parameter(s). For example, you can record cell efficiency as a function of some doping density, $\eta(N_A)$, or whatever, there are really many possibilities.

The list of properties to be recorded is made in the Record Set-up, see Figure 8.1. With the help of the five choice-menus (type-property-layer-defect-level) the user can chose a property and add it to the list on the left side of the panel using an Insert or Replace button. There are eight types of properties to be recorded: *I-V* characteristics, General properties, Cell definition, Interfaces, Energy band panel, Generation panel, Occupations and AC panel.
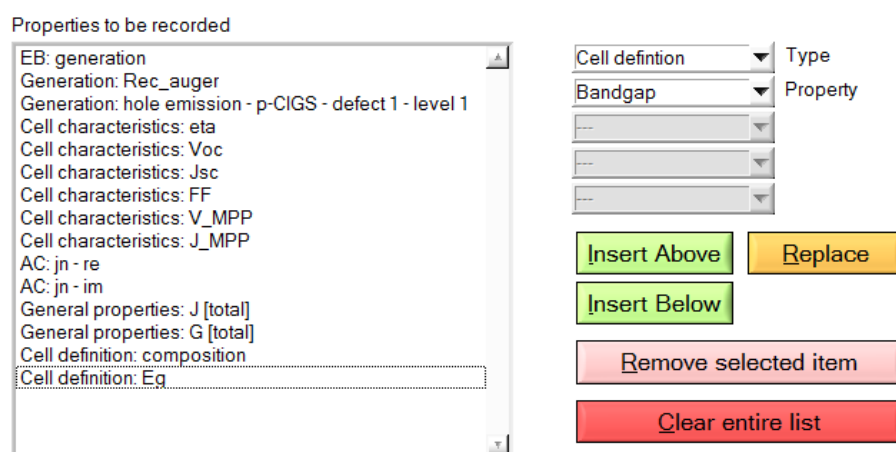
Figure 8.1   Setting up a recorder.

Pleas note carefully that it is also possible to record the cell definition properties! This allows you to see how SCAPS interpreted the parameters you set (in the cell definition or in the batch). This might be very helpful when dealing with graded variables. However, in recent SCAPS versions, this is easier done with the

green Graph View button in the Cell Definition Panel, once at least one calculation has been done; an equilibrium calculation, that is at V and in dark, is sufficient, see Section 3.5.7.

You are able to record properties which are not available in the simulations, e.g. the occupation of defect 3 when there is only one defect. Of course SCAPS can not record what is not available. However, this will not lead to an interruption of the calculations.

Recorder settings can be loaded and saved in a similar way as batch settings, only the extension of the files are now '*.srf'.

8.2 Recorder calculations

Clicking the 'Calculate: recorder'-button leads to a batch recording, which is only available when a batch is set. In this case the action list as it is set on the action panel will be ignored. SCAPS will determine which calculations need to be done in order to record the asked properties. For example when a property of the AC-panel is to be recorded SCAPS will perform a *C-f* simulation at the frequency determined by the working point. When a cell characteristic is to be recorded an *I-V* simulation will be performed starting from 0 V stopping at the open circuit voltage.

SCAPS doesn't a priori know the open circuit voltage. Hence it will perform an *I-V* simulation from 0 V up to a predefined maximum voltage with the 'stop at Voc' option on. This predefined maximum voltage can be set on the numerical panel and has a default value of 2.00V. The accuracy of the determination of the cell characteristics is determined by the increment voltage used in the *I-V* simulation. SCAPS uses the value which is present on the action panel unless it is larger than a predefined minimum value. This value can also be set on the numerical panel and has a default value of 0.05V.

| Maximum voltage for the recording of Solar cell characteristics (V) | ⇕ 2.00 |
| Minimum voltage increment for the recording of Solar cell characteris | ⇕ 0.05 |

Figure 8.2   Recorder settings on the numerical panel

8.3 Analysing the recorder results

The results of a recording can be accessed through the 'Recorder Results' button on the action panel (or on the EB- or AC-panel). Recorded properties of the type *IV characteristics*, *General properties* or *Interfaces* are immediately plot as a function of one of the batch parameters. Record properties of the other types are plot as a function of the position in the cell (the mesh). The results can also be saved to a file, see Figure 8.3. For each type of property type a different file is saved. The user can chose which files are to be made using the checkboxes seen in Figure 8.3. The show option has the same meaning as everywhere in SCAPS. But as not all property types can be showed in one window, only the last one (= the lowest checkbox checked on Figure 8.3) will be showed.

☑ IV characteristics
☐ General properties
☑ Cell definition
☐ EB panel
☐ Interfaces
☑ Generation
☐ Occupations
☐ AC panel

[save]

[show]

Figure 8.3   Saving and showing recorder results

Chapter 9: Curve fitting

The purpose of curve fitting is to vary one or more parameters in the cell definition to obtain a fit between one or more measured curves and the simulation. The numerical algorithms are taken from [7]. If there is only one parameter to fit: the golden section search ([7], p. 401). When there are 2 to 9 parameters to fit: the downhill simplex method, or Nelder-Mead method ([7], p. 408).

9.1 General principles

The parameters to fit are the parameters set in the batch set-up, together with their range and linear or logarithmic character. Not all batch parameters are usable as curve fitting parameters: parameters that are file names (e.g. definition files, spectrum files, absorption files), and parameters of an on/off nature (e.g. 'illumination on/off', 'tunnelling on/off') are not usable. The curve fitter can only be set up when at least one parameter allowable for curve fitting has been defined in the batch set-up.

The measurements to be fit should be selected in the curve fitting set-up. Currently, a maximum of 9 measurements is possible. A fixed parameter can be attached to each measurement. This can be useful when you want e.g. to fit 4 dark *I-V* measurements at 4 different temperatures, or 5 illuminated *I-V* curves at 5 illumination intensities. A limited selection of parameters is offered by the user interface as a fixed parameter of a measurements: only those which can be controlled in an experiment (as we think: we do e.g. not believe that one can make different samples that differ by their electron capture constant c_n of some defect…).

9.2 Setting up the curve fitter

The curve fitter can be set by clicking the 'Curve fit set-up'-button on the action panel. This button is however only available when a batch is set. Once the curve fitter is set, it can be launched by clicking the 'calculate: curve fitter'-button.

As already mentioned, the curve fitting set-up takes the allowable parameters from the batch set-up, complete with their range and linear/logarithmic nature. In the curve fitting set-up, the starting value of this parameter should be set, see Figure 9.1: by default it is the central value in the range (taking into account the linear or logarithmic character). You can also start from the actual value of a parameter in the present cell definition.

Parameters to be fitted to the measurements (These can only be changed in the Batch Set-up Panel !)		Minimum	Maximum	Lin / Log	Start Value	Start Value = Actual Value ? All	None
p-CIGS (L1)	thickness[µm]	1.000E+0	5.000E+0	linear	3.140E+0	☑	
n-CdS (L2)	shallow donor density	1.000E+14	1.000E+18	log.	1.000E+16	☐	

Figure 9.1 Curve fitting parameters taken over from the batch set-up; setting the start value to the central value in the range, or to the actual value in the cell definition.

Addition SCAPS 3.3.03, february 2016. The 'simultaneous' option in the batch settings is now also used by the curvefitter. The first parameter of a group of 'simultaneous' parameters is treated as a master curvefit

parameter, and is varied independently by the curvefitter. The other simultaneous parameters are treated as slaves of that master; each time the master is given a next value in the curvefitting process, the slave values follow automatically. Thus, a group of one master + several slaves is treated as only one curvefit variable.

When a parameter was defined as a table list in the batch set-up, you did not have the option to set its linear or logarithmic character (the batch calculation does not need it, it just takes the values from the list). But the curvefitter needs to know the linear or logarithmic character. Therefore, you have the possibility to set the lin or log property of a parameter in the curvefit set-up,when this parameter was defined as a table list in the batch set-up; even when this curvefit parameter was a slave parameter, you can set its lin/log property then.

The slave parameter values, that must always follow the master values, are found from interpolation. As usuall in SCAPS, the lin/log property of a variable affects the interpolation.

To assist you with the curvefit set-up, groups of curvefit variables (thus one master and one or more slaves) are set in an own color, see Figure 9.2. The first curvefit variable is a group of two thicknesses d_1 (the master) and d_2 (the slave); the second variable is a group of 3 variables: N_{A3} (the master) and μ_{n3} and μ_{p3} (the slaves). With these settings, $d_1 + d_2 = d = 1$ μm for all values of master d_1. This is thus a way to vary the thickness of two layers, while the total thickness remains constant. In the second group, doping-depend mobilities $\mu_n(N_A)$ and $\mu_p(N_A)$ are set. The values of $\mu_n(N_A)$ and $\mu_p(N_A)$ are for each value of master N_A interpolated between the table values set in the batch set-up; note that we have set here the lin/log property of N_A to 'logarithmic'.
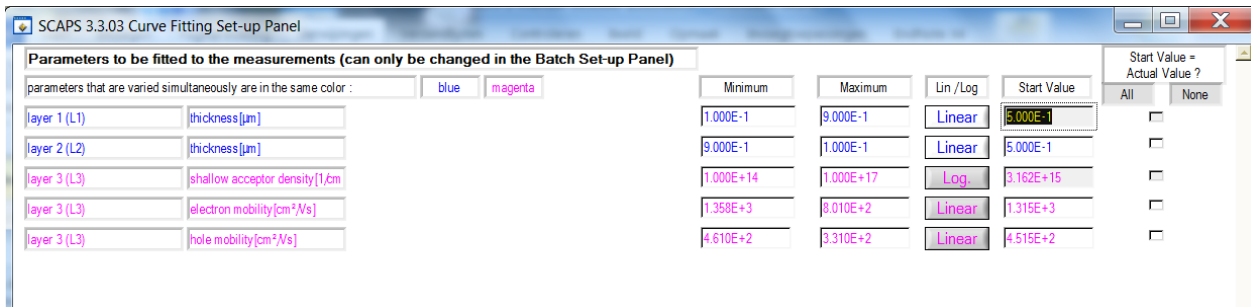
Figure 9.2   Example of a batch and curvefit parameter set-up. There are two independent curvefit variables: one is a group with two thicknesses; and the other is a group with one doping density N_A and two mobilities μ_n and μ_p.

Then the measurements should be set, see Figure 9.3. If 'show' is clicked, the measurement is shown in the simulation panel (*I-V*, *C-V*, *C-f* or *QE*).
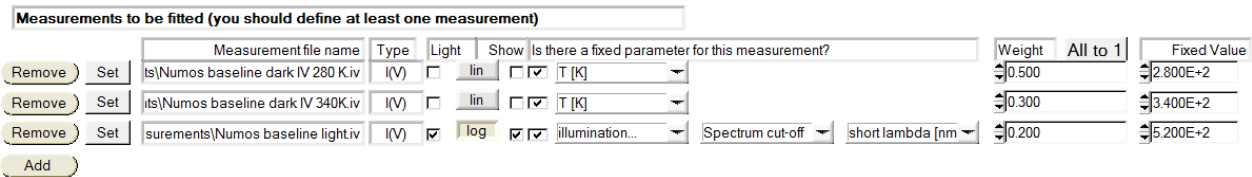
Figure 9.3   Measurement set-up in the curve fitter: the measurement type and illumination, the linear/logarithmic character, and a possible fixed parameter; the (relative) weight of a measurement is available from SCAPS≥3.3.02 on.

The correspondence between a simulation and a measurement is expressed by the χ^2 sum ('*Chi square*'), that we define here as:

$$\chi^2 = \frac{\sum_i \left(y_{\text{meas}} - y_{\text{calc}}\right)^2}{\sum_i \left(y_{\text{meas}}\right)^2} \text{ (linear) or } \chi^2 = \frac{\sum_i \left(\log|y_{\text{meas}}| - \log|y_{\text{calc}}|\right)^2}{\sum_i \left(\log|y_{\text{meas}}|\right)^2} \text{ (logarithmic)} \tag{31}$$

where y_{calc} is the simulated property and y_{meas} the measured property (thus: *I, C* or *QE*), and only the points *i* are considered from the overlap in the simulation and measurement range of *x* (thus: *V, f* or *λ*). In these expressions, the simulations are linearly or logarithmically interpolated at the x_{meas} values. Our normalisation of χ^2 ensures that the χ^2 of all measurements are dimensionless and are scaled to 1: a fit is good if $\chi^2 \ll 1$. It also allows that the χ^2 values of the individual measurements/simulations can be compared and summed. Please note that the definition Eq. (31) of χ^2 deviates somewhat from the more standard definition as e.g. given in the *Recipes*:

$$\chi^2_{\text{Recipes}} = \sum_i \frac{\left(y_{\text{meas}} - y_{\text{calc}}\right)^2_i}{\sigma^2_i} \qquad \text{(the \textit{Recipes}, p. 660)} \tag{32}$$

where σ_i is the standard deviation of measurement point *i*. This definition Eq. (32) is also dimensionless, but it does not show the 'scaled to unity' property as discussed above, and it can cause numerical inaccuracies or problems if the measurement is thought to be nearly perfect or even perfect ($\sigma \to 0$ or $\sigma = 0$). Also, attributing an individual standard deviation to each measurement point, even if it were known, is laborious for the user, and (more important ? ☺) very laborious for the programr. For this reason we prefer our own definition (31) of χ^2 above the mathematically more standard definition (32).

We use χ^2_{tot} as the figure of merit of the whole curve fitting action:

$$\chi^2_{tot} = \frac{\sum\limits_i w_i \chi^2_i}{\sum\limits_i w_i} \to \frac{1}{n} \sum_i \chi^2_i \text{ if all } w_i = 1 \text{ (or no weights set, SCAPS} \le 3.3.01) \tag{33}$$

where χ^2_i are the χ^2 values of the individual measurements, *n* is the number of measurements, and w_i the relative weight that the user attributed to each measurement in the Curvefit Set-up Panel (Figure 9.3). In SCAPS versions $\le 3.3.01$, no weight could be set, and Eq. (33) then reduces to its most right-hand member. A user can interpret the weight w_i of measurement *i* as a parameter for the standard deviation, if all measurement point in a measurement had the same standard deviation (then $w \approx \left(\sum y^2_{\text{meas}}\right)\!\!\big/\sigma^2$).

Some numerical parameters should also be set in the curve fitting set-up, see Figure 9.4.

max #steps [25]   initial step size [0.1000]   minimal step size (stop criterion) [1.00E-10]   minimal Chi-square (stop criterion) [1.00E-7]

Load CurveFitting Settings   Save Curve Fitting Settings   Print Curve Fitting Panel   OK   ☐ write the fitted parameters in the cell delinition?

Figure 9.4   Numerical parameters in the curve fitting set-up

The step size of a parameter variation is relative to the range of this parameter (taking into account the linear or logarithmic nature of this parameter). Do not expect a perfect curve fitting after 25 steps only: 100 steps or so is more realistic, … but takes more time. The curve fitter walks around though the parameter space, and stops when any of the stop criteria is met (maximum number of iterations exceeded, or step size small enough, or total χ^2_{tot} low enough). The parameter set then obtained is only written in the cell definition when the button at the bottom right is clicked. When you visualise the results of the curve fitting (§9.4), you will be able to judge the quality of the curve fitting, and you will get another chance to write the 'fitted' parameters in the cell definition. The curve fitting settings can be saved and loaded, and are also included in the 'save/load all SCAPS settings' actions.

9.3 Defining groups of simultaneous curve fitting parameters
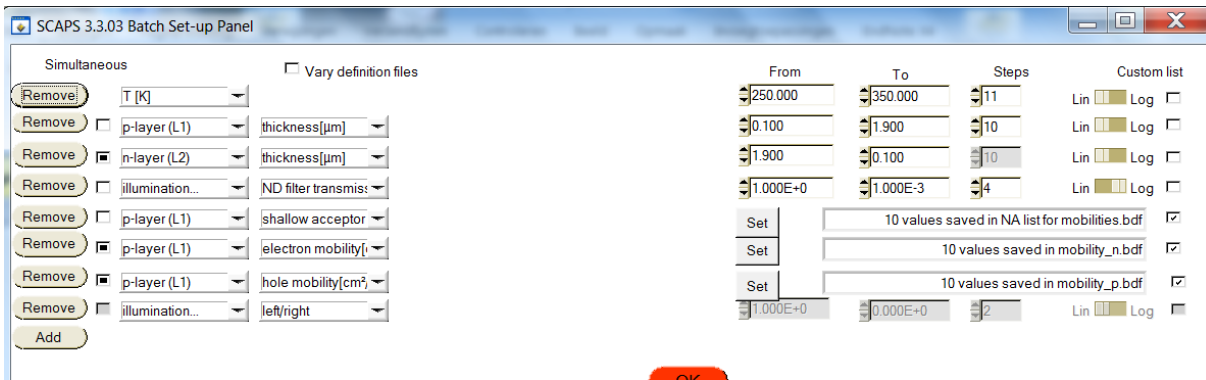
Figure 9.5   Example of a batch set-up. The settings of d_1 and d_2 in parameters 2 and 3 is such that always $d_1 + d_2 = d = 2$ μm. Parameters 5, 6 and 7 contain lists that define a relationship $\mu_n(N_A)$ and $\mu_p(N_A)$. In this batch set-up panel, there is no possibility to set the linear or logarithmic character of these variables N_A, μ_n, μ_p, as this is not relevant for the batch calculation. When passing to the curve fitting set-up, you will have to set the lin/log character of these variables. Parameter 8 is of the on/off type, and thus will not be used as a curve fitting parameter.

The 'simultaneous' option in the batch settings is now also used by the curve fitter. A group of simultaneous batch parameters can also form a group of simultaneous curve fitting parameters (remember that not all possible batch parameters are allowed as curve fitting parameters: parameters of the on/off type (e.g. illumination on/off), and parameters that are a list of files (e.g. spectrum files) are discarded by the curve fitter. Figure 9.5 is an example of a batch set-up, with detailed comments in the figure caption.

The first parameter of a group of 'simultaneous' parameters is treated as a master curve fit parameter, and is varied independently by the curve fitter. The other simultaneous parameters are treated as slaves of that master; each time the master is given a next value in the curve fitting process, the slave values follow automatically. Thus, a group of one master + several slaves is treated as only one curve fit variable.

So finally there are two types of curve fitting variables: independent ones, consisting of one variable only; and groups of dependent ones, consisting of one master and one or more slave parameters. To assist you with the curve fit set-up, independent variables are set in black as before, but groups of dependent curve fit variables are set each in an own colour, see Figure 9.6.
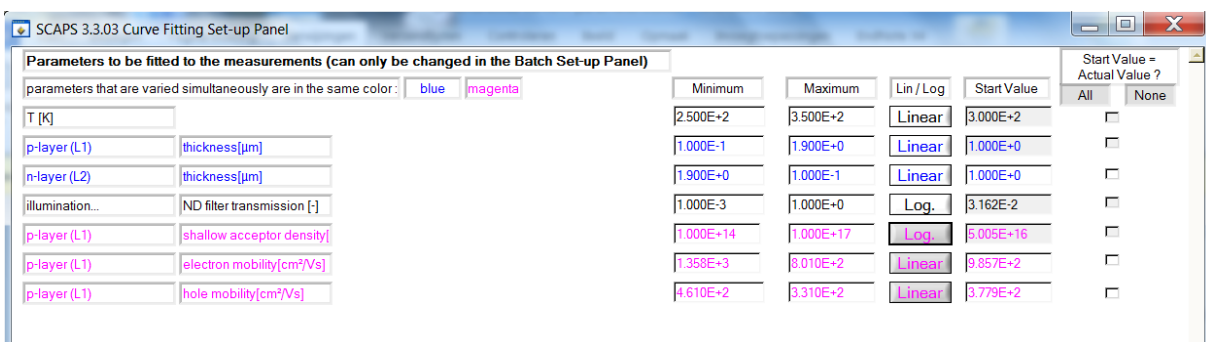
Figure 9.6   Curve fitting parameters derived from the batch set-up of Figure 9.5. There are two groups of dependent variables, the blue group and the pink group. Two parameters (temperature T and ND filter transmission) are independent ones. The illumination left/right parameter is not in the curve fitting parameter list. You can set the lin/log character of the variables that were derived from a list in the batch set-up, thus N_A, μ_n and μ_p of layer 1.

The slave parameter values, that must always follow the master values, are found from interpolation. As usual in SCAPS, the lin/log property of a variable affects the interpolation. When a parameter was defined as a table list in the batch set-up, you did not have the option to set its linear or logarithmic character (the batch calculation does not need it, it just takes the values from the list). But the curve fitter needs to know the linear or logarithmic character. Therefore, you have the possibility to set the lin. or log. property of a

parameter in the curve fit set-up, when this parameter was defined as a table list in the batch set-up; even when this curve fit parameter was a slave parameter, you can set its lin/log property then, see Figure 9.5

When the curve fitter varies d_1, it simultaneously varies d_2; and when it varies N_A, it simultaneously varies $\mu_n(N_A)$ and $\mu_p(N_A)$; here we have given N_A a logarithmic character, and μ_n and μ_p a linear character, meaning that the $\mu(N_A)$ data are first 'plotted' in a μ vs. $\log(N_A)$ plot, and then linearly interpolated. Our example thus illustrates a way to warrant a constant total thickness over two layers, and a way to take the proper dependencies of some materials parameters into account.

9.4 Analysing the curve fitting results

The curve fitting results can be accessed by clicking the 'Curvefitting results'-button on the action panel, which opens the curve fitting results panel, that allows you to select a variety of graphs to visualise and evaluate the entire curve fitting, see Figure 9.7.
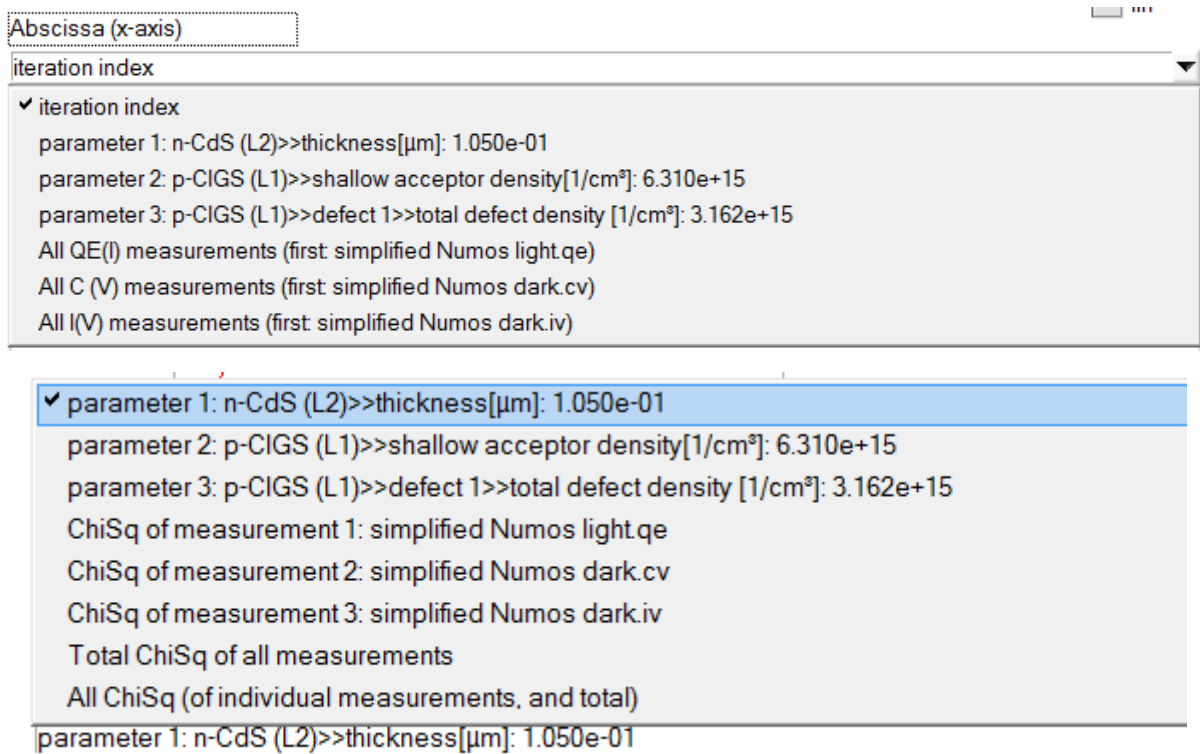
Figure 9.7   Choices of abscissa (x-axis, top) and ordinate (y-axis, bottom) to visualise the curve fitting results.

The examples shown in Figure 9.8 to Figure 9.11 illustrate the Numos exercise 1 example. The file Numos CIGS baseline.def first has been simplified to simplified Numos CIGS baseline.def, by taking a single energy defect level wherever a band of defects was presents: this is speeding up the calculations substantially. All settings, including the batch settings and the curve fitting settings, are saved in 'simplified Numos curvefitting.scaps'. This calculation took 24 minutes on my computer… The iteration was stopped after 73 steps because the required accuracy ($\chi^2_{tot} < 10^{-7}$) was met. Please note:

- do not expect a decent curve fit after e.g. 10 iteration steps only! In the example here, 50 – 60 steps seem to be necessary (Figure 9.8, Figure 9.9).
- do not expect such a very nice fit in a real situation! Here, the measurements were 'constructed', that is, calculated with SCAPS. Thus, an exact fit is possible. With real measurements, it can very well be that χ^2 never gets below 10^{-2} or 10^{-3}, meaning that your measurements can not be fitted very well with the present cell definition.

- it is also clear that you should not exaggerate with the number of parameters and the number of measurements. Though 9 is (now) the allowed maximum for both, I would recommend to use much more modest numbers, e.g. ≤ 3, certainly to start with.

- *The recipes* state that it is a good idea to start a curve fitting over again starting with the parameters obtained from a previous curve fitting calculation. You can do so by saving the fitted parameter values in the cell definition, and running the curve fitting calculation again starting from the 'actual values', possibly with a smaller initial step, and/or with a narrower range for the parameters.
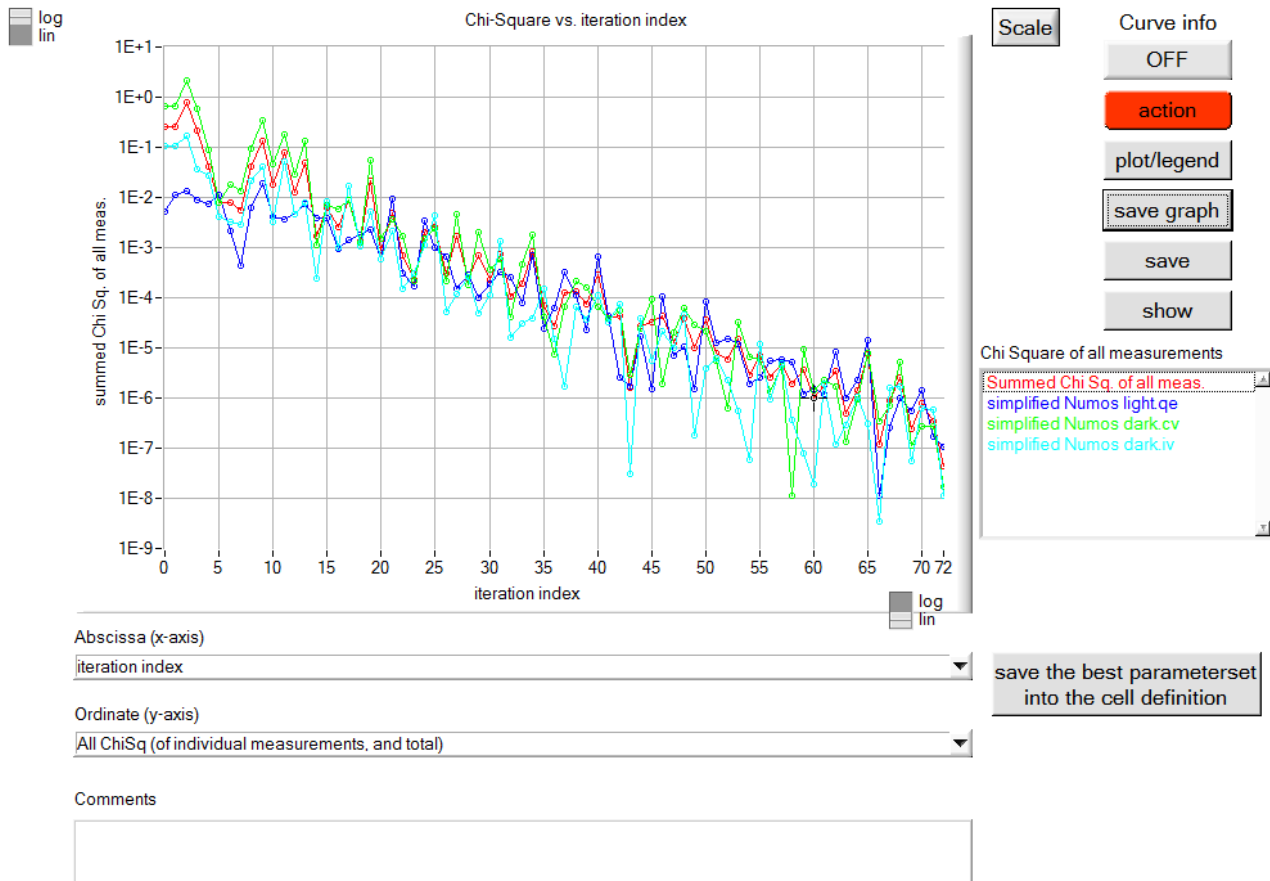
Figure 9.8   Some examples of curve fitting results: Numos exercise 1, with a simplified problem file. The evolution of all χ^2 values with iteration index. When you are satisfied with the results, you can insert the fitted parameter values in the cell definition by clicking the grey button.
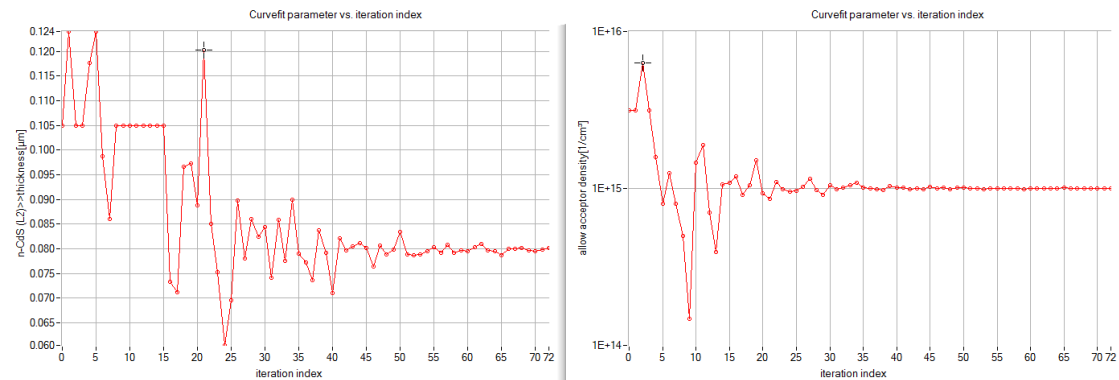
Figure 9.9   Some examples of curve fitting results: Numos exercise 1, with a simplified problem file. The evolution of the parameters d_{CdS} and $N_{A, CIGS}$ with the iteration index
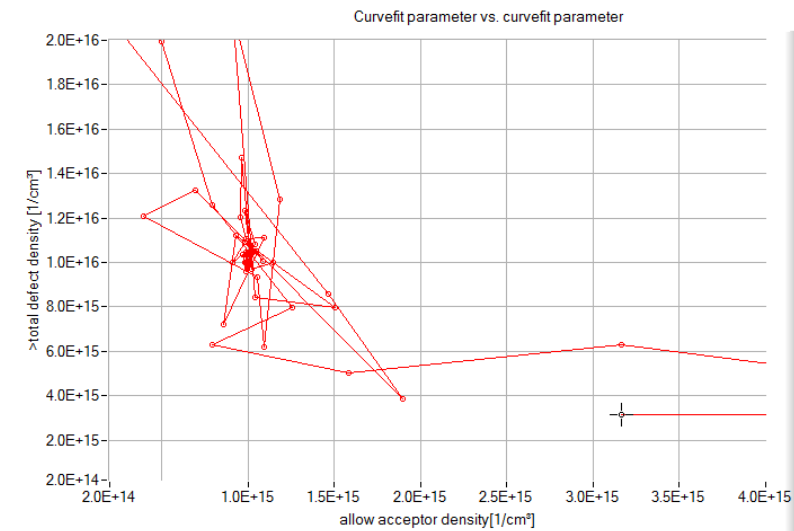
Figure 9.10   Some examples of curve fitting results: Numos exercise 1, with a simplified problem file. Walking in the parameter space to find optimum values for $N_{A,\ CIGS}$ and $N_{t,\ CIGS}$.
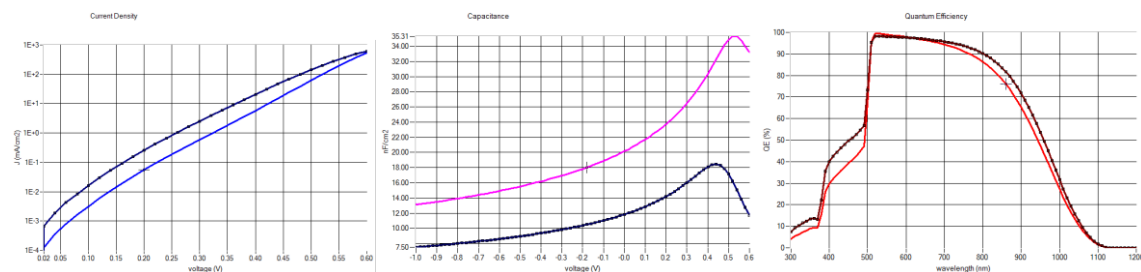
Figure 9.11   Some examples of curve fitting results: Numos exercise 1, with a simplified problem file. The measurements (dots), the initial calculation (simplified Numos CIGS baseline.def) and the calculation after 73 curve fitting iterations: *I-V* (left), *C-V* (middle) and *QE* (right).

9.5 Curve fitting résumé

1. Set up a rough SCAPS model for your cell
2. Verify that scaps is reading your measurements correctly (units, normalisation per area,…): display your measurements and check if all is OK.
3. Refine the model, by hand. As there are probably > 100 parameters to input in your model, you can't leave this to the computer, you should do it yourself. Consider the scaps curve fitting facility only as a fine-tuning.
4. Define the parameters you want to fit in the batch set-up, together with their range and linear/logarithmic nature. Do a quick batch calculation (only very few steps per parameter) to check if your measurement indeed could be fitted by the parameters you have selected
5. Set-up the curve fitter: the parameters were already set in the batch set-up; now only specify the starting value (mid-range or actual value). You have already checked that the measurements are available and readable by scaps. Now select the measurements to curve-fit, set their linear/logarithmic nature, and set one fixed parameter for a measurement if necessary.
6. First use a modest number of iteration steps, e.g. < 10, and run the curve fitting calculations, to see if everything is going as expected.
7. If it looks promising, set a higher number of steps, run the curve fitter, be patient and wait, and hope that the computer will have solved your problem. Do not be too disappointed when the result is less than you hoped for…

Chapter 10: Scripting

SCAPS is and has been designed to be a user-interactive program. The most important computer should be based on neurons instead of on transistors. It is important that the user understands what is physically happening rather than performing more simulations than (s)he ever could analyse. Nevertheless, SCAPS also provides the possibility to write a script and run it.

There are several levels of sophistication for the SCAPS script. One can use it to create a personalized version of SCAPS, to automate actions within the user interface, to use SCAPS in symbiosis with another program, to run SCAPS without mouse-clicks…

An overview of the SCAPS script language is given in §10.4, its use is explained in the preceding sections.

10.1 Running SCAPS externally

Normally SCAPS is started by clicking an icon on the desktop. Internally in Windows, a command line attached to this icon is executed. This command line just contains the full path of the SCAPS .exe file (thus scaps2902.exe or scaps3002.exe or so). Now you can add extra arguments to this command line: a list of filenames to be loaded/executed before SCAPS starts: these can be one or more of the definition, action,…, script files listed above; also a spectrum file and a generation file can be given. The order of execution is: first .def, then .act, .scaps, .spe, .gen, .sbf, .srf, and finally .script. By doing so, you can ensure that SCAPS starts in the condition that you prefer, not in the condition fixed by the SCAPS developers: good news for those who had to set e.g. the wavelength range appropriate for CIGS each time again and again.

Here are several ways to start SCAPS from a command line, and to edit this line:

- Make, e.g. with Notepad, a batch file with extension .bat (this stems from the very old MS-DOS times, but is still supported in Windows). Write the command line in one line, e.g.

  ```
  scaps2903.exe all CIGS.scaps AM0.spe
  ```

  A .bat file is run by double clicking it; you can also make a shortcut to it on the desktop. To edit the .bat file, right-click the name and select 'edit', or directly open from Notepad.

- Right-click the normal SCAPS-icon on your desktop, select 'properties', and edit the third line ('target'). This is well suited when you do this once and for all. When you would alter the command line more frequently, the previous method might feel more comfortable.

- Some programs (e.g. Matlab) provide options to run external programs. Of course you can benefit from these options to run SCAPS or a SCAPS script within such a program environment. Example '*.m*'-files (for Matlab users) which allow communication with SCAPS are available for users on request.

10.2 Running a script

A script-file is a normal text file, that contains commands that are equivalent to a mouse-click. During execution of the script, SCAPS will not respond to user interventions (mouse or keyboard). After execution of the script, SCAPS returns to it's normal interactive mode, or is switched off (to be set with a `set quitscript…` script command). [SCAPS 3.3.04, october 2016] Also, the updating of the screen (e.g.

drawing of all curves in the Energy Bands panel) can be suppressed with a `set script_display_mode…` script command.

When no script file was found in the command line, SCAPS remains the same interactive program as it has always been. A few SCAPS scripts are distributed with SCAPS to serve as an example.

There are several levels of sophistication to use scripts.

10.2.1 Automate mouse clicking

The basic use is that you write down all actions you would do in the interactive mode, and then can leave the lab. One advantage of doing so would be that you could split up your to-do list in several smaller batch jobs instead of one gigantic batch job (by giving several `load batchsettings` commands), and that you save the results in between (by giving `save results` or `save graphs` commands). This is safer when there is a risk that your computer (or SCAPS ☹) would hang up underway; also, all results are waiting for you when you come back to the lab.

This automatisation of interactive mouse clicking can be initiated from a batch file, an external program, but also from within SCAPS. Hereto set up a script using the 'script set-up'-button and run it using the 'execute script'-button.

Figure 10.1   Running a script from within the SCAPS user interface.

10.2.2 Run external programs within SCAPS

A more sophisticated use is that you start and run an own (or another) program somewhere during the execution of the script, by giving a `runsystem` command. For example, you could load a problem, do just the equilibrium calculation, and save the results of the generation panel to a file; this also contains the mesh. You could then open an own program, read this mesh-file and use it to do an own calculation of the optical generation, and save this in a file with the format of a SCAPS generation file. When your function returns, the script is continued. You could then load this generation file, and do all calculations you want. This procedure is more convenient then the full-manual method that several SCAPS users have intensely used.

In this way, the communication between scaps and your own program is only via the file system: both SCAPS and your program read and write files, but do not communicate directly with each other. Also, reading and especially writing files in a SCAPS format might be cumbersome for a programr.

10.2.3 Running dynamically linked libraries

For advanced users, a more direct communication method between SCAPS and an own program has been implemented. The user program should have been compiled as a *dynamically linked library*, and should be placed in the SCAPS mother directory as a .dll and a .lib file. Also, these filenames, the name of the dll-program(up to now there are only two), and the definition (the argument list) of this function is fixed. This functionality seems to address well skilled programmers only. However, one such dll, with a flexible functionality, is distributed with SCAPS, and using it does not require high level programmer skills: see the description of the `rundll` command below, and the examples distributed with SCAPS. In contrast, we are confident that the basic use of SCAPS scripts can facilitate the simulation work of a broad class of SCAPS users.

10.3 The script editor

A script-file can be written in any text editor, e.g. notepad. A typing error is however quickly made. In order to avoid this, it is strongly recommended to use the script editor, which is launched when clicking the 'script set-up'-button on the action panel, see Figure 10.1, which opens the panel shown in Figure 10.2.
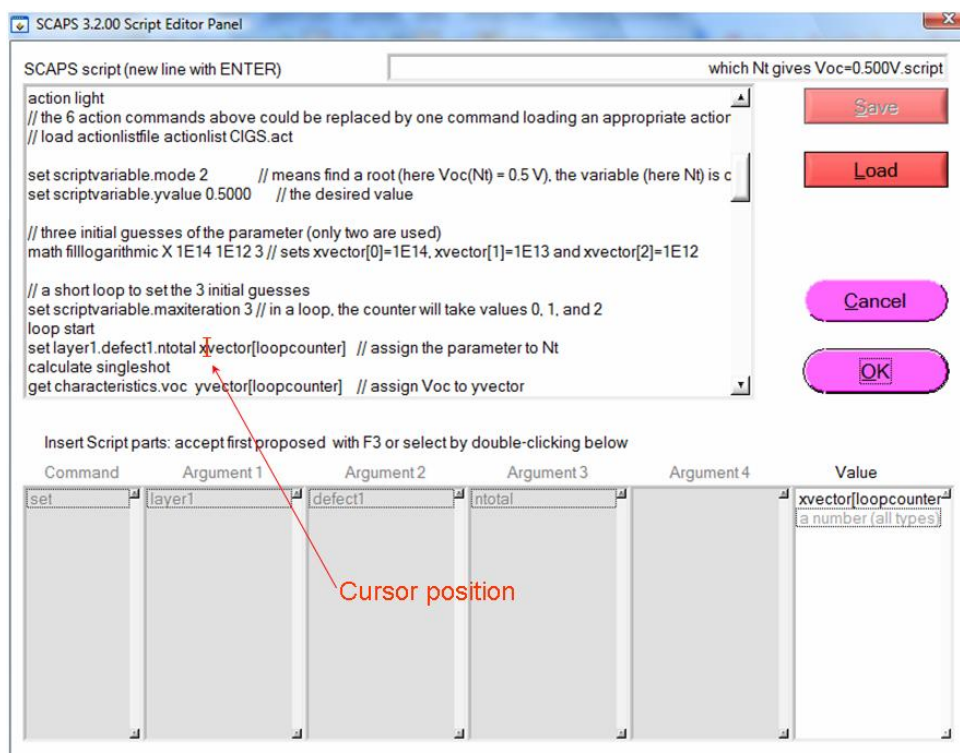
Figure 10.2   Screen shot of the SCAPS script editor.

- When placing the cursor in an existing line of a script (as shown in Figure 10.2), the components of this command line (thus: the command, the arguments and the value) are shown in the six blocks at the bottom of the panel. When typing a new line in a script, the parts of the command line available so far are proposed in these blocks, and can be selected and placed in the editor box of the script, either by double-clicking on the desired argument or by pressing F3.
- Existing script files can be saved and loaded.

A tip:

When developing a script, frequently insert a `show scriptvariable` statement. The execution of a script stops at such statement, and you can inspect the script variables to see if all is going as you intended. When everything is checked and OK, you can remove or out comment these `show` statements.

10.4 The SCAPS script language

10.4.1 General

The SCAPS-directory, this is where the scaps.exe file resides, is noted as 'SCAPS\'.

A comment line in a script is a line that cannot be interpreted as a command line. E.g. any line starting with a punctuation character is treated as comment. You can also add comment at the end of a command line. The Script Editor will recognize such in-line comment when it starts with a double punctuation, except ']]' (thus e.g. '//' or '!!' or '>>' are OK …).

   All command lines in a script consist of up to three parts:

```
command argument value
```

where `command` and `argument` are reserved words, and `value` is free with some restrictions, depending on the command line. The three components of the command line are separated by whitespace (spaces, tabs,..), but should be on one line. They are not case-sensitive (upper case or lower case letters do not matter). The possible commands are given in Table 10.1.

Table 10.1   Available script commands

load	action	set	math	calculate
save	clear	get	loop	run
		extract	show	rundll
			plot	runsystem

Whilst processing a script, SCAPS internally maintains a few variables, as specified in Table 10.2. The user can use these variables in `set` and `get` commands, and some are used internally in a `loop`. Also, these variables are passed to an external dll function, that can be made by the user.

In this list (and in this entire manual) the notation {m} should be replaced by x, y, z, u, v, w, in order to get script variables like xvector, wvector, uvalue, ny , nv, zname, uindex...SCAPS ≥ 3.3.09, december 2020: {m} can be any letter of the Latin alphabet (without accents), thus any letter from (a, b, c, d, e, f, g, h, i, j, k, l, m, n, o, p, q, r, s, t, u, v, w, x, y, z). Hence, also variables like na, nm, kindex, pvalue, qvector, bname... are accepted.

Table 10.2   SCAPS script variables

name	C-type	default value	max value
{m}value	double	0	
{m}vector	array of double	0	
n{m}	int	0	
{m}valuename	character string	empty	max size 256 bytes
{m}vectorname	character string	empty	max size 256 bytes
{m}index	int	0	
loopcounter	int	0	
maxiteration	int	25	
looperror	double	1E30	
maxerror	double	1E-3	
status	int	0	
mode	int	0	
filename	character string	empty	max size 256 bytes

10.4.2 Load commands

Syntax:

```
load argument value
```

Where `load` is the reserved command word, `argument` can take 8 reserved values, and `value` is a filename, without path. The filename can contain spaces. The files are supposed to reside in their default directories. There is (exceptionally) some freedom allowed in the name of the argument: just writing `definition`, `action`, `batch`, `record`, `allscaps`, `spectrum` or `generation` will also do.

Table 10.3   SCAPS load commands

command	argument	value	default-directory
load	definitionfile	a filename	scaps\def
load	actionlistfile	a filename	scaps\def
load	initialworkpointfile	a filename	scaps\def
load	batchsettingsfile	a filename	scaps\bdf
load	recordersettingsfile	a filename	scaps\bdf
load	allscapssettingsfile	a filename	scaps\def
load	spectrumfile	a filename	scaps\spectrum
load	generationfile	a filename	scaps\generation

load definitionfile.filenamelist listfile[index]

Here, listfile is a file that contains a list of filenames, in this example, a list of deffiles. The index can be a number, or one of the SCAPS script variables loopcounter, maxiteration or mode. This format works for the load statements definitionfile to generationfile. Example:

load generationfile.filenamelist my genfilelist.gen[loopcounter].

load	singleshotbatch		scaps\bdf

The last argument (load singleshotbatch) is slightly deviating from the others as it does not take a value. The purpose of this command is to work together with the command get recorder. When load singleshotbatch is called the batch settings file *singleshotbatch.sbf* is loaded. This file sets a batch calculation with one calculation at the working point temperature. So it enables you to perform a recording of a singleshot calculation. This option is very useful as a lot of properties can only be accessed in the script through performing a record calculation and taking the value via get recorder. In this way you can access e.g. the electrical field distribution in the structure and do calculations with it. The temperature in this batch is set to the working point value when the command load singleshotbatch is called. Hence when you vary the temperature afterwards you should repeat the command again.

With SCAPS 3.3.10, March 2021, a few more load commands are introduced. They are used to load the temporary definition and generation files that were generated by the new command math split_tandem, without having to know the exact names of these files (see next Chapter)

command	argument	value	default-directory
load	definitionfile.temporary.tandem_cell	a filename	scaps\def
load	definitionfile.temporary.top_cell	a filename	scaps\def
load	definitionfile.temporary.bottom_cell	a filename	scaps\def
load	generationfile.temporary.tandem_cell	a filename	scaps\generation
load	generationfile.temporary.top_cell	a filename	scaps\generation
load	generationfile.temporary.bottom_cell	a filename	scaps\generation

10.4.3 Save commands

Syntax:

```
save argument value
```

Where `save` is the reserved command word, `argument` has a compound syntax; the first part can take 3 reserved values (`settings, results` or `graphs`). The `value` is a filename, without path. The filename can contain spaces. The files are supposed to reside in their default directories.

Commands in blue are new in SCAPS version 3.3.02, june 2015

Table 10.4   SCAPS save commands

command	argument	value	default-directory
save	scriptvariables	a filename	scaps\results
save	scriptvariables.xyzuvw	a filename	scaps\results

SCAPS ≥ 3.3.09, december 2020: From now on, `save scriptvariables` will save all 26 indices, values and vectors. The new command `save scriptvariables.xyzuvw` acts as the traditional(SCAPS ≤ 3.3.08) command `save scriptvariables`: only xindex,… windex; xvalue,… wvalue and xvector…wvector are saved, in the order x, y, z, u, v and w. This is to support users that have developed e.g. MatLab tools that import and process the traditional file exported by `save scriptvariable` (SCAPS ≤ 3.3.08)

command	argument	value	default-directory
save	settings.definitionfile	a filename	scaps\def
save	settings.actionlistfile	a filename	scaps\def
save	settings.batchsettingsfile	a filename	scaps\bdf
save	settings.recordersettingsfile	a filename	scaps\bdf
save	settings.allscapssettingsfile	a filename	scaps\def
save	results.eb	a filename	scaps\results
save	results.genrec	a filename	scaps\results
save	results.SCAPSgenerationfile	a filename	scaps\generation contains only an $G(x)$ table, in the format of a SCAPS generation file (SI units for G)

```
save settings.definitionfile.filenamelist listfile[index]
```

Here, listfile is a file that contains a list of filenames, in this example, a list of deffiles. The index can be a number, or one of the SCAPS script variables `loopcounter`, `maxiteration` or `mode`. This format works for the save statements `settings.definitionfile` to `results.SCAPSgenerationfile`.

save	scriptvectors.asSCAPSgenerationfile	SCAPS 3.3.07 of 4-5-2018 (and SCAPS 3.3.08 of 18-5-2020 for the first command "…as…generation…"). The `save scriptvectors` command takes as value `AB filename` or `AB listfile[…]`. Here A and B are scriptvectors (only use the letters x, y, …, w; SCAPS ≥3.3.09: all letters
save	scriptvectors.asSCAPSfilterfile	
save	scriptvectors.asSCAPSabsorptionfile	
save	scriptvectors.asSCAPSoptcapt_n_file	
save	scriptvectors.asSCAPSoptcapt_p_file	
save	scriptvectors.asSCAPSfilterfile.filenamelist	
save	scriptvectors.asSCAPSabsorptionfile.filenamelist	

save	`scriptvectors.asSCAPSoptcapt_n_file.filenamelist`	
save	`scriptvectors.asSCAPSoptcapt_p_file.filenamelist`	
save	`scriptvectors.asSCAPSgradingfile.grading(composition_y)`	
save	`scriptvectors.asSCAPSgradingfile.grading(abs_position_x)`	
save	`scriptvectors.asSCAPSgradingfile.grading(rel_position_x_d)`	
save	`scriptvectors.asSCAPSgradingfile.filenamelist.grading(composition_y)`	
save	`scriptvectors.asSCAPSgradingfile.filenamelist.grading(abs_position_x)`	
save	`scriptvectors.asSCAPSgradingfile.filenamelist.grading(rel_position_x_d)`	

a… z are allowed). If saved as a filter, absorption or optical capture file, vector A is assumed to contain wavelengths in nm. If saved as a grading file or as a generation file, A contains the position within a layer (left = 0; x (in μm) if "absolute" (then a line "position: absolute" is added to the file) and x/d if "relative" (then "position relative" is added, see Section 3.5.5.1 page 15.). The user should have checked that the A and B contain the appropriate information, SCAPS cannot know that. If no ".filenamelist" appears in the argument, the information in A and B is saved in a file with name `filename` (old files with this name are overwritten!). If the argument contains ".filenamelist" appears, then the last part of the value is a list file specification, as explained above.

| save | `scriptvectors.asSCAPSspectrumfile` | |

From SCAPS 3.3.12 subversion 4-6-2025 on: the first scriptvector should contain wavelength in nm, the second photon flux in #/cm2/nm/s (thus, as calculated by the script). Only give one letter (a to z) for the script vectors, see above

save	`results.ac`	a filename	scaps\results
save	`results.iv`	a filename	scaps\results
save	`results.cv`	a filename	scaps\results
save	`results.cf`	a filename	scaps\results
save	`results.qe`	a filename	scaps\results
save	`results.recorder`	a filename	scaps\results
save	`graph.eb.wholepanel`	always .png !	scaps\results
save	`graph.eb.energybands`	always .png !	scaps\results
save	`graph.eb.carrierdensities`	always .png !	scaps\results
save	`graph.eb.currents`	always .png !	scaps\results
save	`graph.eb.ftraps`	always .png !	scaps\results
save	`graph.ac.wholepanel`	always .png !	scaps\results
save	`graph.ac.currents.amplitude`	always .png !	scaps\results
save	`graph.ac.currents.phase`	always .png !	scaps\results

save	graph.ac.potentials.amplitude	always .png !	scaps\results
save	graph.ac.potentials.phase	always .png !	scaps\results
save	graph.genrec.wholepanel	always .png !	scaps\results
save	graph.genrec.genrec	always .png !	scaps\results
save	graph.genrec.ftraps	always .png !	scaps\results
save	graph.iv.wholepanel	always .png !	scaps\results
save	graph.iv.iv	always .png !	scaps\results
save	graph.iv.recombination	always .png !	scaps\results
save	graph.cv.wholepanel	always .png !	scaps\results
save	graph.cv.cv	always .png !	scaps\results
save	graph.cv.gv	always .png !	scaps\results
save	graph.cv.Mott-Schottky	always .png !	scaps\results
save	graph.cv.dopingprofile	always .png !	scaps\results
save	graph.cf.wholepanel	always .png !	scaps\results
save	graph.cf.cf	always .png !	scaps\results
save	graph.cf.gf	always .png !	scaps\results
save	graph.cf.Nyquist	always .png !	scaps\results
save	graph.cf.G(f)/f-f	always .png !	scaps\results
save	graph.qe.wholepanel	always .png !	scaps\results
save	graph.qe.qe	always .png !	scaps\results
save	graph.recording.wholepanel	always .png !	scaps\results
save	graph.recording.resultsgraph	always .png !	scaps\results
save	graph.grading.wholepanel	always .png !	scaps\results
save	graph.grading.gradinggraph	always .png !	scaps\results

SCAPS 3.3.10, March 2021: new save commands to save temporary tandem cell components, with their appropriate (fixed) filename)

save	definitionfile.temporary.tandem_cell	a filename	scaps\def
save	definitionfile.temporary.top_cell	a filename	scaps\def
save	definitionfile.temporary.bottom_cell	a filename	scaps\def
save	generationfile.temporary.tandem_cell	a filename	scaps\generation
save	generationfile.temporary.top_cell	a filename	scaps\generation
save	generationfile.temporary.bottom_cell	a filename	scaps\generation

10.4.4 Action commands

Syntax:

```
action argument value
```

Where `action` is the reserved command word, `argument` can take the values in Table 10.5, and `value` is a numerical value or a script variable or a filename, without path. The filename can contain spaces. The

files are supposed to reside in their default directories. Some values can take two values only (0 or 1). There is a (very) limited degree of freedom in the exact arguments. E.g. instead of `iv.checkaction`, you can also write `iv.doiv` or `iv.iv`. Instead of `batch.checkaction`, you can also write `batch.dobatch` (as in the user interface of SCAPS < 2.10); and alike with `recording.dorecord`. When the value of these commands is omitted, the value 1 is assumed (giving a clear meaning to the form `doiv`, `docv`,…, `dobatch`…).

Table 10.5   SCAPS action commands

command	argument	value	remark
action	workingpoint.temperature		Kelvin
action	workingpoint.kT		Volt or eV
action	workingpoint.voltage		Volt
action	workingpoint.frequency		Hz
action	workingpoint.numberofpoints	≥ 2	
action	dark	none	overrides `light`
action	light	none	overrides `dark`
action	illumination.fromleft	none	overrides `fromright`
action	illumination.fromright	none	overrides `fromleft`
action	generationfrominternalmodel	none	overrides `generationfromfile`
action	lightflux_specified		alternative for the above command
action	spectrumfile	filename	`scaps\spectrum` equivalent with `load spectrumfile`
action	spectrumcutoff.on	none	overrides `spectrumcutoff.off`
action	spectrumcutoff.off	none	overrides `spectrumcutoff.on`
action	spectrumcutoff.shortlambda		Nm
action	spectrumcutoff.longlambda		Nm
action	intensity.ND		
action	intensity.T		%
action	generationfromfile	none	overrides `generationfrominternalmodel`
action	generation_specified		alternative for the above command
action	generationfile	filename	`scaps\generation` Equivalent with `load generationfile`
action	generationfromfile.attenuation	a number	these 4 script commands have the same effect, as there is only one generation attenuation defined; the attenuation is expressed as a transmission in %
action	generationfrommodel.attenuation	a number	
action	generation.attenuation	a number	
action	generationattenuation	a number	
action	spectrum_defined.frommodel		
action	spectrum_defined.fromfile		

action	`spectrum_defined.file`	filename	equivalent with `action spectrumfile`
action	`generation_defined.frommodel`		
action	`generation _defined.fromfile`		
action	`generation_defined.file`	filename	equivalent with `action generationfile`
action	`generationmodel.constantG`		
action	`generationmodel.exponentialfromL`		
action	`generationmodel.exponentialfromR`		
action	`generationmodel.rectangular`		
action	`generationmodel.gaussian`		
action	`generationmodel.rect_exponentialtails`		
action	`generationmodel.rect_gaussiantails`		
action	`generationmodel.constantG.generation`		
action	`generationmodel.exponentialfromL.generation`		
action	`generationmodel.exponentialfromL.decaylength`		
action	`generationmodel.exponentialfromR.generation`		
action	`generationmodel.exponentialfromR.decaylength`		
action	`generationmodel.rectangular.generation`		
action	`generationmodel.rectangular.central_position`		
action	`generationmodel.rectangular.full_width`		
action	`generationmodel.gaussian.generation`		
action	`generationmodel.gaussian.central_position`		
action	`generationmodel.gaussian.standard_deviation`		
action	`generationmodel.rect_exponentialtails.generation`		
action	`generationmodel.rect_exponentialtails.central_position`		
action	`generationmodel.rect_exponentialtails.rectangle_full_width`		
action	`generationmodel.rect_exponentialtails.decaylength`		
action	`generationmodel.rect_gaussiantails.generation`		
action	`generationmodel.rect_gaussiantails.central_position`		
action	`generationmodel.rect_gaussiantails.rectangle_full_width`		
action	`generationmodel.rect_gaussiantails.standard_deviation`		
action	`spectrummodel.white_constant_flux`		
action	`spectrummodel.white_constant_power`		
action	`spectrummodel.blackbody`		
action	`spectrummodel.monochromatic_constant_flux`		
action	`spectrummodel.monochromatic_constant_power`		
action	`spectrummodel.white_constant_flux.total_flux`		

action	spectrummodel.white_constant_flux.short_wavelength
action	spectrummodel.white_constant_flux.long_wavelength
action	spectrummodel.white_constant_power.total_power
action	spectrummodel.white_constant_power.short_wavelength
action	spectrummodel.white_constant_power.long_wavelength
action	spectrummodel.blackbody.total_power
action	spectrummodel.blackbody.temperature
action	spectrummodel.blackbody.short_wavelength
action	spectrummodel.blackbody.long_wavelength
action	spectrummodel.monochromatic_constant_flux.total_flux
action	spectrummodel.monochromatic_constant_flux.central_wavelength
action	spectrummodel.monochromatic_constant_flux.full_bandwidth
action	spectrummodel.monochromatic_constant_power.total_power
action	spectrummodel.monochromatic_constant_power.central_wavelength
action	spectrummodel.monochromatic_constant_power.full_bandwidth
action	generationmodel.nx
action	spectrummodel.nwavelengths

SCAPS 3.3.05, december 2016: the spectrum and generation model commands above also work for the 'initial workpoint' (metastable defects), e.g. `action initialwp.generationmodel.gaussian.central_position`

action	initialwp.temperature		Kelvin
action	initialwp.kT		Volt or eV
action	initialwp.voltage		Volt
action	initialwp.numberofpoints	≥ 2	
action	initialwp.dark	none	overrides light
action	initialwp.light	none	overrides dark
action	initialwp.generationfrominternalmodel	none	overrides generationfromfile
action	initialwp.spectrumfile	filename	scaps\spectrum
action	initialwp.spectrumcutoff.on	none	overrides spectrumcutoff.off
action	initialwp.spectrumcutoff.off	none	overrides spectrumcutoff.on
action	initialwp.spectrumcutoff.shortlambda		nm
action	initialwp.spectrumcutoff.longlambda		nm
action	initialwp.intensity.ND		
action	initialwp.intensity.T		%
action	initialwp.generationfromfile	none	overrides generationfrominternalmodel
action	initialwp.generationfile	filename	scaps\generation
action	initialwp.generationfromfile.attenuation		%
action	iv.startV	Volt	

action	`iv.stopV`	Volt	
action	`iv.points`	≥ 2	
action	`iv.increment`	Volt	
action	`iv.doiv`	none	equivalent to action iv.checkaction 1
action	`iv.checkaction`	0 or 1	1 is the default
action	`iv.stopafterVoc`	none	
action	`iv.continueafterVoc`	none	
action	`cv.startV`	Volt	
action	`cv.stopV`	Volt	
action	`cv.points`	≥ 2	
action	`cv.increment`	Volt	
action	`cv.docv`	none	equivalent to action cv.checkaction 1
action	`cv.checkaction`	0 or 1	1 is the default
action	`cf.startf`	Hz	
action	`cf.stopf`	Hz	
action	`cf.total.points`	≥ 2	
action	`cf.points.per.decade`	≥ 1	
action	`cf.totalpoints`	≥ 2	
action	`cf.pointsperdecade`	≥ 1	
action	`cf.checkaction`	0 or 1	1 is the default
action	`cf.docf`	none	equivalent to action cf.checkaction 1
action	`qe.startlambda`	nm	
action	`qe.stoplambda`	nm	
action	`qe.points`	≥ 2	
action	`qe.increment`	nm	
action	`qe.checkaction`	0 or 1	1 is the default
action	`qe.doqe`	none	equivalent to action qe.checkaction 1

10.4.5 Clear commands

Syntax:

```
clear argument
```

With `clear scriptvariables`, all script variables (or all but 2 or 3 elements) are set to their defaults. `clear simulations` is equivalent to pressing the 'clear all simulations' button in the SCAPS action panel.

Table 10.6   SCAPS clear commands

command	argument	value	remarks
clear	`scriptvariables.all`	no value	see text above

clear	scriptvariables.allbutfirst3	no value	leaves $\{m\}$vector$[i]$ with $i = 0, 1, 2$. n$\{m\}$ are set to 3. The other script variables are not affected.
clear	scriptvariables.allbutfirst2	no value	idem, but with $i = 0, 1$
clear	scriptvariables.allbutlast3	no value	idem, but shifts elements i = n$\{m\}$-1, n$\{m\}$-2, n$\{m\}$-3 to i = 0, 1, 2 and leaves them
clear	scriptvariables.allbutlast2	no value	idem, but shifts elements i = nx-1, nx-2 (or with ny) to i = 0, 1 and leaves them
clear	simulations	no value	see text above
clear	plot	no value	clears all script graphs; identical with plot clear
clear	scriptgraphs	no value	identical with plot clear or clear plot
clear	actions	no value	unchecks all 4 actions (IV, CV, Cf and QE) and restores the workpoint settings to a default (300 K, 0 V, 1 MHz, 5 pts)
clear	all	no value	clears all simulations, all scriptvariables and all plots: equivalent to clear simulations plus clear scriptvariables.all but not clear actions

10.4.6 Set commands

Syntax:

```
set argument value
```

where set is the reserved command word, argument can take the reserved values from Table 10.8. The set command can also be used to set the script variables. The third part of the set command line is value: this is a numerical value, a script variable or a filename, without path. The filename can contain spaces. The files are supposed to reside in their default directories.

Some values can take two values only (0 or 1). When the value is a numerical value, you can specify a number, e.g. 1.25E16, or one of the internal script variables mode, loopcounter, maxiteration, $\{m\}$index, $\{m\}$value, $\{m\}$vector and n$\{m\}$. Here $\{m\}$ can be one of the letters x, y, …, w., and n$\{w\}$ is the number of elements in the corresponding $\{w\}$vector. SCAPS $\geq$ 3.3.09, december 2020: $\{m\}$ can be any of the 26 letters in (a, …, z).

The values of the internal variables $\{m\}$value, $\{m\}$vector, … can be set directly with a set-command; also, they are used and possibly changed in SCAPSUserFunction.dll (see §10.4.14). The value of n$\{m\}$ can be set directly with the set command; it is also updated in some commands: get, math and clear, see later. The allowed indices in SCAPS script vectors are listed in the Table 10.7.

When you set a new value of n$\{m\}$, the length of the corresponding vector is updated. If the new value is smaller than the previous one, data gets lost, if it is larger, the vector is extended with non-initialised (random) numbers. Before setting a script variable, you might want to re-initialise them with one of the clear commands, see later.

These conventions for the use of scriptvectors in the set and get (see further) commands are summarised in the Table below.

Table 10.7   Allowed formats for SCAPS script vector elements

script vector format	index	meaning; remarks
{m}vector	no index	Only as an argument of set scriptvariable... or as the value of get characteristics... The value of n{m} is incremented, all existing elements of {m}vector are shifted one up, and the value of the set scriptvariable... command, or the parameter to get, is placed at {m}vector[0]
{m}vector[-1]	-1	Only as an argument of set scriptvariable... or as the value of get characteristics... The value of n{m} is incremented, and the value of the set scriptvariable... command, or the parameter to get, is placed as the new last element of {m}vector
{m}vector[i]	a number	i is an integer number and should be $0 \leq i \leq n\{m\}$ -1
{m}vector[last]		For your comfort: internally, last is replaced with the appropriate n{m}-1
{m}vector[loopcounter]	a scriptvariable	
{m}vector[mode]	a scriptvariable	
{m}vector[maxiteration]	a scriptvariable	
{m}vector[{n}index]	a scriptvariable	m and n can differ: you can specify e.g. zvector[yindex]
{m}vector[{n}value]	a scriptvariable	m and n can differ: you can specify e.g. uvector[wvalue]. The value of {n}value is first rounded to the nearest integer.
{m}vector[{n}vector[…]]	a scriptvariable	Here the index … of the inner {m}vector takes one of the forms allowed in this Table. You can nest many vectors, but that should not be a reason to exaggerate

Table 10.8   SCAPS set command-arguments

command	argument	value	remark
set the script variables			
set	scriptvariable.maxiteration	integer	
set	scriptvariable.status	integer	
set	scriptvariable.mode	integer	
set	scriptvariable.looperror		
set	scriptvariable.maxerror		
set	scriptvariable.xvalue		
set	scriptvariable.xvalue		
set	scriptvariable.{m}vector[i]		$0 \leq i \leq nx$ - 1, or i = -1, or no index
set	scriptvariable.n{m}	integer	
set	scriptvariable.{m}name	character string	It sets the name of a scriptvector. length < 256. SCAPS3.3.12, October 2024: this is still serviced, but no

			longer proposed by the script editor.
set	scriptvariable.name.{m}vectorname	character string	SCAPS3.3.12, October 2024: now proposed in the script editor to give an own label to a script vector
set	scriptvariable.name.{m}valuename	character string	New in SCAPS3.3.12, October 2024: You can now give an own label to the scalar values in the Script Variables Panel and output file.
set	scriptvariable.filename	character string	length < 256
set	scriptvariable.filename.filenameli st	listfile[…]	the format of the list file with an [index] is explained under the save commands above
set	scriptvariable.filename.SCAPSpath	character string	length < 256
set	scriptvariable.filename.SCAPSpath		

The filename is completed to (or changed to) the full default SCAPS path. E.g. the command `scriptvariable.filename.SCAPSpath mycell.def` will set **filename** to (e.g.) c:\MB\SCAPS try-outs\def\mycell.def. If no value is given, the actual **filename** is completed to the SCAPS default path. This command is useful to pass a filename to another program, that might need to know the full path (e.g. the SCALSdll function). `script_display_mode…` and `batch_display_mode…` are new in SCAPS 3.3.04 (october 2016).

general set commands			
set quitscript.interactiveSCAPS		no value	the default
set quitscript.quitSCAPS		no value	
set script_display_mode.normal		no value	the default
set script_display_mode.not_suppressed		no value	same as .normal
set script_display_mode.mostly_suppressed		no value	screen updating is mostly suppressed during script execution
set script_display_mode.fully_suppressed		no value	screen updating is fully suppressed during script execution
set batch_display_mode.normal		no value	the default
set batch_display_mode.not_suppressed		no value	same as .normal
set batch_display_mode.suppressed		no value	screen updating is suppressed during batch execution
set errorhandling.toscreen		no value	
set errorhandling.appendtofile		no value	the default during script execution
set errorhandling.overwritefile		no value	

set	errorhandling.outputlist.truncate	no value	the default
set	errorhandling.outputlist.fillzeros	no value	
set	errorhandling.outputlist.fillwhite	no value	
set	external.Rs		Ωcm^2
set	external.Rsh		Ωcm^2
set	external.Gsh		Scm^{-2}
set	internal.reflection		fraction, not %
set	internal.transmission		fraction, not %

illumination set commands

set	generation.from_scriptvectors	AB	scriptvector A should contain the position x in μm, and vector B the generation $G(x)$ in #/cm³/s. SCAPS store these values in an internal generation file (*internalSCAPS.gen*), sets the illumination conditions to "generation specified", "generation from file", and sets the internal gen file a generation filename. (SCAPS 3.3.08 of 18-5-2020). This is equivalent with saving (A,B) in a generation file to be specified (with `save scriptvectors_as scapsgenerationf ile AB filename`), setting the generation conditions as should and loading the generation file just created: but now all this with one command.
set	illumination.fromleft	no value	
set	illumination.fromright	no value	
set	qe.photonflux	a scalar value	$\#.cm^{-2}s^{-1}$
set	qe.photonpower	a scalar value	Wcm^{-2}
set	qe.constant_photonflux	no value	*QE* calculation mode
set	qe.constant_photonpower	no value	*QE* calculation mode

numerical set commands (see Numerical Panel, and CV-analysis panel) (new in SCAPS 3.3.02 version 2-10-2015)

set	numerical.CV-analysis.layer		first layer = 1

set	numerical.CV-analysis.side		-1=LEFT; +1=RIGHT
set	numerical.CV-analysis.points		# points in local neighbourhood
set	numerical.CV-analysis.order_middle		polynomial order within local neighbourhood
set	numerical.CV-analysis.order_edge		polynomial order at edges of local neighbourhood
set	numerical.include_G_mesh_in_SCAPS_mesh		0 = NO; 1 =YES;
set	numerical.recalculate_mesh		0 = NO; 1 =YES;
set	numerical.recalculate_mesh.points		maximum # of mesh points introduced/deleted per iteration cycle
set	numerical.recalculate_mesh.min_ratio		see Numerical Panel
set	numerical.recalculate_mesh.max_ratio		see Numerical Panel
set	numerical.recalculate_mesh.generation_limit		see Numerical Panel
set	numerical.recalculate_mesh. recombination_limit		see Numerical Panel

set	contact.Sn		cm.s^{-1}
set	contact.Sp		cm.s^{-1}
set	contact.opticalfilter.on	no value	
set	contact.opticalfilter.off	no value	
set	contact.opticalfilter.transmission	no value	
set	contact.opticalfilter.reflection	no value	
set	contact.opticalfilter.value		fraction, not %
set	contact.opticalfilter.file	a filename	scaps/filter
set	contact.opticalfilter	0 or 1	
set	contact.workfunction		V or eV
set	contact.flatband.off	no value	
set	contact.flatband.once	no value	
set	contact.flatband.always	no value	

layer set commands: replace `layer` with `layer1, layer2, … layer7`

set	layer.remove	S
set	layer.duplicate	
set	layer.split.leftthickness	μm
set	layer.split.rightthickness	μm
set	layer.split.leftfraction	-
set	layer.split.rightfraction	-
set	layer.add_default	a layer with default

properties is added to the right of the cell structure; the layernumber in the set layer1… command is ignored. November 2021: (if no layer number specified) works also for empty cells, that do not yet have a layer

set	layer.name	character string	
set	layer.thickness		μm
set	layer.Eg		eV
set	layer.chi		V or eV
set	layer.epsilon		-
set	layer.NC		cm^{-3}
set	layer.NV		cm^{-3}
set	layer.vthn		cm.s^{-1}
set	layer.vthp		cm.s^{-1}
set	layer.mun		cm^{2}V^{-1}s^{-1}
set	layer.mup		cm^{2}V^{-1}s^{-1}
set	layer.NA		cm^{-3}
set	layer.ND		cm^{-3}
set	layer.radiative		cm^{3}s^{-1}
set	layer.Augern		cm^{6}s^{-1}
set	layer.Augerp		cm^{6}s^{-1}
set	layer.absorption.file	a filename	scaps\absorption
set	layer.absorption.A		eV$^{-1/2}$cm^{-1}
set	layer.absorption.B		eV$^{+1/2}$cm^{-1}

From SCAPS ≥ 3.3.07, january 2018: absorption model script commands. Replace `layer` with `layer1`, `layer2`,… and `absorptionA` with `absorptionB` as necessary

set	layer.absorptionAfile	a filename
set	layer.absorptionBfile	a filename
set	layer.absorptionmodelA.background	equivalent with …`background.on` (and alike for the next 5 commands)
set	layer.absorptionmodelA.Egstep	
set	layer.absorptionmodelA.Egsqrt	

set	layer.absorptionmodelA.power1	
set	layer.absorptionmodelA.power2	
set	layer.absorptionmodelA.subbandgap	
set	layer.absorptionmodelA.background.on	-
set	layer.absorptionmodelA.background.off	-
set	layer.absorptionmodelA.background.alpha_background	cm^{-1}
set	layer.absorptionmodelA.Egstep.on	-
set	layer.absorptionmodelA.Egstep.off	-
set	layer.absorptionmodelA.Egstep.alpha_constant	cm^{-1}
set	layer.absorptionmodelA.Egsqrt.on	-
set	layer.absorptionmodelA.Egsqrt.off	-
set	layer.absorptionmodelA.Egsqrt.alpha0	cm^{-1}
set	layer.absorptionmodelA.Egsqrt.beta0	cm^{-1}
set	layer.absorptionmodelA.power1.on	-
set	layer.absorptionmodelA.power1.off	-
set	layer.absorptionmodelA.power1.alpha1	cm^{-1}
set	layer.absorptionmodelA.power1.beta1	cm^{-1}
set	layer.absorptionmodelA.power1.Eg1	eV or -multiple
set	layer.absorptionmodelA.power1.exponent1	-
set	layer.absorptionmodelA.power2.on	-
set	layer.absorptionmodelA.power2.off	-
set	layer.absorptionmodelA.power2.alpha2	cm^{-1}
set	layer.absorptionmodelA.power2.beta2	cm^{-1}
set	layer.absorptionmodelA.power2.Eg2	eV or -multiple
set	layer.absorptionmodelA.power2.exponent2	-
set	layer.absorptionmodelA.subbandgap.on	-
set	layer.absorptionmodelA.subbandgap.off	-
set	layer.absorptionmodelA.subbandgap.Ee0	eV

In the layer defect commands below, give layer and defect their appropriate number, e.g.:

`set layer3.defect2.Et 0.6`

Note: as of now(7-11-2018) a defect level is not implemented in the SCAPS script: the property you set will be applied to all levels… not as it should be for multivalent defects ☹…

set	layer.defect.remove	
set	layer.defect.duplicate	inserts a defect after the defect with the given number, and with identical properties as this defect
set	layer.defect.add_default	adds a defect with default properties after

			the last defect in this layer; the defect number in the command is ignored
set	`layer.defect.singlelevel`	no value	
set	`layer.defect.uniform`	no value	
set	`layer.defect.gauss`	no value	
set	`layer.defect.CBtail`	no value	
set	`layer.defect.VBtail`	no value	
set	`layer.defect.neutral`	no value	
set	`layer.defect.singledonor`	no value	
set	`layer.defect.doubledonor`	no value	
set	`layer.defect.singleacceptor`	no value	
set	`layer.defect.doubleacceptor`	no value	
set	`layer.defect.amphoteric`	no value	
set	`layer.defect.aboveEV`	no value	
set	`layer.defect.belowEC`	no value	
set	`layer.defect.aboveEi`	no value	
set	`layer.defect.Et`		eV
set	`layer.defect.Echar`		eV
set	`layer.defect.capture_cross_section.electrons`		cm^2
set	`layer.defect.capture_cross_section.holes`		cm^2
set	`layer.defect.Ntotal`		cm^{-3}
set	`layer.defect.Npeak`		$cm^{-3}eV^{-1}$

Metastable defect set commands: replace `layer` with `layer1, ...,`

set	`layer.metadefect.E_HE`		eV
set	`layer.metadefect.E_EE`		eV
set	`layer.metadefect.E_EC`		eV
set	`layer.metadefect.E_HC`		eV
set	`layer.metadefect.E_TR`		eV
set	`layer.metadefect.aboveEV`	no value	
set	`layer.metadefect.belowEC`	no value	
set	`layer.metadefect.aboveEi`	no value	
set	`layer.metadefect.allowmeta`	0 or 1	
set	`layer.metadefect.Ntotal`		cm^{-3}

interface set commands: replace `interface` with `interface1, interface2, ... interface6`

set	`interface.name`	character string

set	interface.IBtunneling.off	no value	
set	interface.IBtunneling.on	no value	
set	interface.IBtunneling.me		--
set	interface.IBtunneling.mh		--

interface defect set commands: replace `interface` with `interface1,...` and `IFdefect` with `IFdefect1, IFdefect2, IFdefect3`. Leave the new commands, set `interfaces_all…,` without change

set	interface.IFdefect.remove	
set	interface.IFdefect.duplicate	as for bulk defects
set	interface.IFdefect.add_default	as for bulk defects
set	interface.1_default_IFdefect	SCAPS3.3.12, October 2024: removes all defects of the interface, then adds one default IF-defect
set	interface.2_default_IFdefects	SCAPS3.3.12, October 2024: removes all defects of the interface, then adds two default IF-defects
set	interface.3_default_IFdefects	SCAPS3.3.12, October 2024: removes all defects of the interface, then adds three default IF-defects
set	interface.all_default_IFdefects	SCAPS3.3.12, October 2024: removes all defects of the interface, then adds all default IF-defects (will only be relevant when the max number of IF defects would be increases to > 3)
set	interfaces_all.inactive	SCAPS3.3.12, October 2024. In all interfaces, remove all IF defects
set	interfaces_all.active_with_1_default_IFdefect	SCAPS3.3.12, October 2024. In all interfaces, remove all IF defects, then adds 1 default IF defect to each interface
set	interface.IFdefect.singlelevel	no value
set	interface.IFdefect.uniform	no value

set	`interface.IFdefect.gauss`	no value
set	`interface.IFdefect.CBtail`	no value
set	`interface.IFdefect.VBtail`	no value
set	`interface.IFdefect.neutral`	no value
set	`interface.IFdefect.singledonor`	no value
set	`interface.IFdefect.singleacceptor`	no value
set	`interface.IFdefect.abovehighestEV`	no value
set	`interface.IFdefect.aboveEVleft`	no value
set	`interface.IFdefect.belowlowestEC`	no value
set	`interface.IFdefect.aboveEileft`	no value
set	`interface.IFdefect.aboveEiright`	no value
set	`interface.IFdefect.aboveEiGap`	no value
set	`interface.IFdefect.Et`	eV
set	`interface.IFdefect.Echar`	eV
set	`interface.IFdefect.Ntotal`	cm^{-2}
set	`interface.IFdefect.Npeak`	$cm^{-2}eV^{-1}$
set	`interface.IFdefect.capture_cross_section.electrons`	cm^2
set	`interface.IFdefect.capture_cross_section.holes`	cm^2
set	`interface.IFdefect.tunneling.on`	no value
set	`interface.IFdefect.tunneling.off`	no value
set	`interface.IFdefect.tunneling.me`	--
set	`interface.IFdefect.tunneling.mh`	--

10.4.7 Get commands

Syntax:

```
get argument variable
```

Here, variable is one of the internal script variables.

When you ask for a scalar property, you can use {m}value or {m}vector[index]: the actual value of the variable will then be overwritten with the result of the get action. Here index is one of the allowed formats for indices in the SCAPS script. Other scalar script variables that can be used are looperror and maxerror.

When you ask for a vectorial properties, like a full *I-V* or *QE* curve, these are placed in two vectors: e.g. *I* in {m}vector. and *V* in {n}vector. If no vectors are specified, xvector and yvector are assumed: thus `get cv` is identical to `get cv xy`. Also note that only the result of the last simulation is acquired: the last single shot simulation, or the last simulation in a batch run.

The `get` command updates {m}name as well.

The purpose of the `get` command is that the script file, or the program launching the script file (e.g. from within SCAPS, from MatLab, another C-program, Windows script or MS-DOS command language…) would have access to variables such as V_{oc} , J_{sc}, η, … or even arrays as $J(V)$, … in a more convenient way then having to retrieve them from a SCAPS output file.

Also, these internal variables can be passed to and updated by the SCAPSUserFunction, that is under the control of the SCAPS user, see §10.4.14.

Table 10.9 SCAPS get commands

command	argument	value and remarks
get calculated solar cell characteristics		
get	characteristics.eta	a scalar script value: {m}value or {m}vector[i] where the index i should be in the range $0 \le i \le$ n{m}-1. Using $i = -1$ means that the value is appended at the end of {m}vector, and that n{m} are incremented with one. Using {m}vector[0] {m}vector (thus without index) means that the size n{m} is incremented with one, all elements of the vectors are shifted one position up, and the value returned by characteristics… is placed at {m}vector[0].
get	characteristics.voc	
get	characteristics.jsc	
get	characteristics.ff	
get	characteristics.vmpp	
get	characteristics.jmpp	
get general calculated characteristics		
get	iv	Two letters should be passed for the value, corresponding with two vectors. The abscissa is saved in the first, the ordinate in the second. (e.g. get cf zw, saves the frequency in the vector zvector and the capacitance in the vector wvector). If the value is omitted, xy is assumed. The sizes n{m} are set automatically, and also the names {m}name are set.
get	cv	
get	gv	
get	cf	
get	gf	
get	qe	
get	gx	

SCAPS 3.3.10, March 2021. New get commands to get properties that are a function of both position x and wavelength λ: the light flux in the cell $\Phi(x, \lambda)$ in photons/cm².s, the *eh* generation rate $G(x,\lambda)$ in #/cm³.s, and the optical absorption constant of the materials $\alpha(x, \lambda)$, in 1/cm².

1- Example:

```
get lightflux.all_internalwavelengths.at_position LF 0.25
```

You can get these properties at a single position, either to be specified by the position x (μm) (in the example: 0.25 μm) or by the mesh index i. Hint: you can specify interesting positions by first calling commands like get.layer2.leftindex, get layer3.rightx, get celllength… Then you get these properties as a function of wavelength λ, either at all internal wavelengths (this will be in most case the wavelengths where the spectrum is defined (file or model)), or at the wavelengths you have specified in a scriptvector (in the example, in *lvector*). The result (Φ, G or α; here Φ) is written in scriptvector *fvector* (in the example)

2. Example:

```
get generation.at_wavelength.all_mesh_positions XG lvector[loopcounter]
```

You can also get these properties as a function of position x (no user choice, the x positions of the internal mesh are filled in the scriptvextor xvector (in this example). These properties are at one

specified wavelength (in nm). You can specify any number (e.g. 532.0) or scriptvariable like the scalar lvalue or any script vector format (in this example: lvector[loopcounter]). The property (in this example *G*) will be filled in gvector.

3. Example:

```
get incidentlightflux.all_internalwavelengths.just_outside_cell lf
```

The incident light flux will be placed in fvector, the wavelengths in lvector. When you opt for *all_internalwavelengths*, these will be placed in lvector. When you opt for *all_userwavelengths*, you should first have filled lvector with the λ values (in nm) of your choice. When the light was set to be incident from left, the light fluxes are at position 0, thus $\Phi(0^-)$ for *just_outside_cell* and $\Phi(0^+)$ for *just_inside_cell*. And when the light was incident from the right, the incident fluxes are $\Phi(L^+)$ for *just_outside_cell* and $\Phi(L^-)$ for *just_inside_cell*, when *L* is the total cell length or thickness. The ratio between inside and outside is the transmission of the front contact filter:

$$\Phi(\text{inside},\lambda) = T_{\text{front}}(\lambda)\Phi(\text{outside}, \lambda).$$

4. Example:

```
get SCAPSinputfile.filter LT my fancy top filter.ftr
```

You can also access the contents of SCAPS input files without having to load or set these input files in your cell definition: you can directly go from the filename (here *my fancy top filter.ftr*) to the script vectors. in this example, the wavelengths of the file are written in *lvector* (in nm), and the filter property (transmission or reflection, in %) is written in *tvector*. You have then the occasion to modify these properties, e.g. *T*(λ) in the script, and only after manipulation load or set them in your cell problem, with, in this example), e.g. by

```
save scriptvectors.asSCAPSfilterfile LT some filename.ftr
```

and then setting it as a contact filter file by

```
set rightcontact.opticalfilter.file some filename.ftr.
```

… and now the list of the new commands…

	argument	value
get	lightflux.all_internalwavelengths.at_position	position *x* in the cell (μm)
get	lightflux.all_internalwavelengths.at_meshindex	mesh index *i* in the cell
get	lightflux.all_userwavelengths.at_position	
get	lightflux.all_userwavelengths.at_meshindex	
get	lightflux.at_wavelength.all_mesh_positions	wavelength λ (nm)
get	generation.all_internalwavelengths.at_position	
get	generation.all_internalwavelengths.at_meshindex	
get	generation.all_userwavelengths.at_position	
get	generation.all_userwavelengths.at_meshindex	
get	generation.at_wavelength.all_mesh_positions	
get	absorption.all_internalwavelengths.at_position	
get	absorption.all_internalwavelengths.at_meshindex	
get	absorption.all_userwavelengths.at_position	
get	absorption.all_userwavelengths.at_meshindex	

```
get    absorption.at_wavelength.all_mesh_positions

get    incidentlightflux.all_internalwavelengths.just_outside_cell

get    incidentlightflux.all_internalwavelengths.just_inside_cell

get    incidentlightflux.all_userwavelengths.just_outside_cell

get    incidentlightflux.all_userwavelengths.just_inside_cell

get    SCAPSinputfile.generation

get    SCAPSinputfile.filter

get    SCAPSinputfile.absorption

get    SCAPSinputfile.grading

get    SCAPSinputfile.optcapt_n

get    SCAPSinputfile.optcapt_p
```

`get    measurement.iv`
 	
`get    measurement.cv`
 	
`get    measurement.gv`
 	
`get    measurement.cf`
 	
`get    measurement.gf`
 	
`get    measurement.qe`
	In the same way, you can now also get a measurement. It places the last calculated *I-V*, *C-V*,… measurement in the vectors specified (if none specified, xy is assumed)

`get    recombination.tot`
 	
`get    recombination.SRH`
 	
`get    recombination.rad`
 	
`get    recombination.aug`
	… and likewise the recombination information on the IV panel (at least one *I-V* or *C-V* calculation should have been done). The first vector will contain the voltage *V*.

`get    recv.tot`
 	
`get    recv.SRH`
 	
`get    recv.rad`
 	
`get    recv.aug`
	These commands are equivalent to the `get recombination` commands.

`get    energybands.EC`
 	
`get    energybands.EV`
 	
`get    energybands.EFn`
 	
`get    energybands.EFp`
 	
`get    energybands.n`
 	
`get    energybands.p`
 	
`get    energybands.chargeindef`
 	
`get    energybands.netdoping`
 	
`get    energybands.chargedensity`
 	
`get    energybands.E`
 	
`get    energybands.field`
	Also the output of the energy bands panel (obtained by clicking show or save in the EB-panel), can be accessed in the script. The output goes to the two vectors specified (or to xvector and yvector if none were specified). The first vector will contain the mesh *x* in μm. The second vector the specified property, e.g. the electric field *E* (note that `get energy.E` and `get energy.field` are equivalent).

get	`energybands.Jn`	Remember that at least one calculation (*I-V, C-V, C-f* or *QE*) should have been done before this information is accessible.
get	`energybands.Jp`	
get	`energybands.Jntunnel`	
get	`energybands.Jptunnel`	
get	`energybands.Jtot`	
get	`energybands.generation`	
get	`energybands.recombination`	
get	`energybands.cumulativeGeneration`	
get	`energybands.cumulativeRecombination`	
get	`energybands.Jtotal_scalar scalar`	All energy other band output comes in the form of two vectors, e.g. $E_{Fp}(x)$ or $J_{tot}(x)$. In contrast, this commands puts the value of J_{tot} in a script scalar, e.g. into xvalue, or yvector[loopcounter],…
get	`recorder`	You can get also the result of a recorder-calculation. You get the result of one recorded property, that is on position `mode` in the recording list as you have specified it. Here `mode` is the script variable. It is somewhat laborious to set up the recorder list from a script (you should set up a recorder file, and load this in the script), set the `mode` variable to the property you want, and then `get` it. But at leastyou can access any variable in SCAPS in thus way.

get mesh characteristics; **layer** should be substituted by **layer1**, **layer2**, … or **layer7** (mind that the mesh is only defined upon calculation, a workpoint calculation is enough)

get	`layer.leftindex`	the index of the leftmost point in the specified layer
get	`layer.leftx`	the position x (in μm) of the leftmost point in the specified layer
get	`layer.rightindex`	the index of the rightmost point in the specified layer
get	`layer.rightx`	the position x (in μm) of the rightmost point in the specified layer
get	`layer.middleindex`	(16-6-2024) the index of the meshpoint in the middle of the specified layer.

		layer.middle_index will also be recognised.
get	layer.middlex	(16-6-2024) the position (in µm) of the middle of the specified layer. layer.middle_x will also be recognised.
get	numberoflayers	the number of layers in the actual cell definition
get	celllength cellength	the total cell length *x* (in µm) of the actual cell definition; both celllength and cellength will work ☺

From SCAPS 3.0.02 on (may 2011), the scalar cell parameters that are available in `set` are made available in `get`. When your cell has graded properties, the parameters that you can `set` or `get` relate to the 'pureA' material (when grading is a function of composition) or to the left side of a layer (when grading is a function of position) (see the SCAPS2.8 add-on manual on grading). The units and remarks are as for the corresponding `set` commands.

Table 10.10   SCAPS get commands (cell definition related)

contact get commands: replace contact with either leftcontact or rightcontact	
get contact.Sn	get contact.opticalfilter.file
get contact.Sp	get contact.opticalfilter
get contact.opticalfilter.on	get contact.workfunction
get contact.opticalfilter.off	get contact.flatband.off
get contact.opticalfilter.transmission	get contact.flatband.once
get contact.opticalfilter.reflection	get contact.flatband.always
get contact.opticalfilter.value	

SCAPS 3.3.10, March 2021: you can also get the optical contact filter property $T(\lambda)$ or $R(\lambda)$ and place it directly into scriptvectors (λ in nm in the first scriptvector, T or R in % in the second). This can be at all internal λ values (the λ values where the spectrum was specified), or at a specified user list of λ values (then interpolation between the internal λ values at each user λ is used). Examples:

`get contact.opticalfilter.all_internalwavelengths LT` (λ in *lvector*, T or R in *tvector*)

`get contact.opticalfilter.all_userwavelengths KR` (λ in *kvector*, T or R in *rvector*)

layer get commands: replace layer with layer1, layer2, … layer7	
get layer.thickness	get layer.NA
get layer.Eg	get layer.ND
get layer.chi	get layer.radiative

```
get    layer.epsilon                          get    layer.Augern

get    layer.NC                               get    layer.Augerp

get    layer.NV                               get    layer.absorption.file

get    layer.vthn                             get    layer.absorption.A

get    layer.vthp                             get    layer.absorption.B

get    layer.mun                              get    layer.composition

get    layer.mup
```

defect get commands: replace `layer` with `layer1, …,` and `defect` with `defect1, defect2` or `defect3`

get `layer.defect.energydistribution`	returns an integer that encodes for single, uniform, Gauß, …
get `layer.defect.chargetype`	returns an integer that encodes for neutral, single donor, ….
get `layer.defect.referencelevel`	returns an integer that encodes for above E_V, below E_C, above E_i
get `layer.defect.Et`	
get `layer.defect.Echar`	
get `layer.defect.capture_cross_section.electrons`	
get `layer.defect.capture_cross_section.holes`	
get `layer.defect.Ntotal`	
get `layer.defect.Npeak`	

Metastable defect get commands: replace `layer` with `layer1, …,`

get `layer.metadefect.referencelevel`	returns an integer that encodes for above E_V, below E_C, above E_i
get `layer.metadefect.E_HE`	
get `layer.metadefect.E_EE`	
get `layer.metadefect.E_EC`	
get `layer.metadefect.E_HC`	
get `layer.metadefect.E_TR`	
get `layer.metadefect.Ntotal`	
get `layer.metadefect.allowmeta`	

About the Script and **grading**: The SCAPS script does not fully support grading, as of now (1-9-2013,version 3.2.01). The properties that you `set` or `get` are either the property of the 'pure A material' (composition $y = 0$) if the property is not

graded, or graded as a function of composition *y*, or the property at the left side of the layer, if the property is graded as a function of position *x*. You cannot access or set any other grading property (thus e.g. the value of the pure B material, bowing parameters, characteristic exponential decay lengths…). However, from SCAPS 3.2.01 (2-9-2013) on you can set a grading file for all the properties above that can be graded. The grading profile for this property is then automatically set to 'graded from file'. To do so, use e.g.

```
set layer1.Eg.file mypersonal_Eg_gradingfile.grd
```

or: `set layer3.composition.file myfavourite_composition_gradingfile.grd`.

The extension *.grd* is necessary, and the program assumes that the file resides in the *…/grading* subdirectory, unless you specify a full file path. (And make sure that the file exists and can be found and read correctly: do a single shot calculation before you launch complicated scripts ☺).

interface get commands: replace `interface` with `interface1, interface2, … interface6`	
`get   interface.IBtunneling.off`	`get     interface.IBtunneling.me`
`get   interface.IBtunneling.on`	`get     interface.IBtunneling.mh`

interface defect get commands: replace `interface` with `interface1,…` and `IFdefect` with `IFdefect1, IFdefect2, IFdefect3`	
`get   interface.IFdefect.energydistribution`	returns an integer that encodes for single, uniform, …
`get   interface.IFdefect.chargetype`	returns an integer that encodes for neutral, single donor, …
`get   interface.IFdefect.referencelevel`	returns an integer that encodes for above E_V left, above highest E_V, below lowest E_C, …
`get   interface.IFdefect.Et`	
`get   interface.IFdefect.Echar`	
`get   interface.IFdefect.Ntotal`	
`get   interface.IFdefect.Npeak`	
`get   interface.IFdefect.capture_cross_section.electrons`	
`get   interface.IFdefect.capture_cross_section_holes`	
`get   interface.IFdefect.tunneling.on`	
`get   interface.IFdefect.tunneling.off`	
`get   interface.IFdefect.tunneling.me`	
`get   interface.IFdefect.tunneling.mh`	

Some `action` commands and some other `set` commands now also have their corresponding `get` command	
`get   action.workingpoint.frequency`	`get   action.generationfromfile.attenuation`
`get   action.workingpoint.temperature`	`get   action.initialwp.temperature`
`get   action.workingpoint.kT`	`get   action.initialwp.kT`

```
get   action.workingpoint.voltage           get   action.initialwp.voltage

get   action.spectrumcutoff.shortlambda     get   action.initialwp.spectrumcutof
                                                  f.shortlambda

get   action.spectrumcutoff.longlambda      get   action.initialwp.spectrumcutof
                                                  f.longlambda

get   action.intensity.ND                   get   action.initialwp.intensity.ND

get   action.intensity.T                    get   action.initialwp.intensity.T

get   action.Pin.from_lamp

get   action.Pin.after_cutoff

get   action.Pin.after_cutoff_and_ND

get   action.Pin.in_cell (same as above)

get   action.Jideal.in_genfile

get   action.Jideal.after_attenuation

get   action.Jideal.in_cell (same as above)

get   action.initialwp.generationfromfile.attenuation

get   external.Rs

get   external.Rsh

get   external.Gsh

get   internal.reflection

get   internal.transmission

get   numerical.qe.photonflux

get   numerical.qe.photonpower
```

The command `get recorder` gets the data from the record results, and hence allows to access almost any property in script mode. The recorded property is selected by the value of the script variable `mode` (the first property in the record setting list is accessed when mode = 0, the next when mode = 1…). If the recorded property is a scalar value as a function of the batch calculation (e.g. the open circuit voltage) the abscissa consists of the numbers of the batch calculations. If the recorded property is a vector (e.g. the conduction band profile) the abscissa value is the mesh. In this case only the recorded vector of the last batch calculation is copied to the script variable. In this view, performing a batch with only one calculation using `load singleshotbatch` is very recommended, see §10.4.2.

10.4.8 The extract command

SCAPS 3.3.07 of 4-5-2018. There is only one extract command in the script:

```
extract layerx ABCD
```

Here x is the layer number (1 to 7 presently; the user should have checked that this layer exists). A to D represent script vectors (use only the first letter x, y, … z). C is supposed to contain the mesh position x in µm over the whole cell, and D a cell property (obtained e.g. with a get command). The the command extracts the information (A,B) of layer x from the whole cell information in (C,D). Also, the vectors A and B get the same name as C and D, with the word "layer x" added to it.

To do the same, several commands were used in earlier versions. As an example, find $n(x)$ in layer 2:

[after a calculation]

```
get energybands.n xy
```
// now xvector contains x and yvector $n(x)$ over the whole cell

```
get layer2.leftindex xindex
```
// now xindex points to the leftmost point of layer2

```
get layer2.rightindex yindex
```
// now yindex points to the rightmost point of layer 2

```
math extract ZXxy
```
// the contents of xvector from xindex to yindex is copied (with an offset) into zvector: $zvector_i = xvector_{i+xindex} - xvector_{xindex}$, for i=0 to (and including) i=yindex-xindex.

```
math extract UYxy
```
// the contents of yvector from xindex to yindex is copied into uvector: $uvector_i = yvector_{i+xindex}$, for i=0 to (and including) i=yindex-xindex.

From SCAPS 3.3.07 of 4-5-2018, this can be replaced with:

[after a calculation]

```
get energybands.n xy
```
// now xvector contains x and yvector $n(x)$ over the whole cell

```
extract layer2 ZUXY
```
// now zvector contains x in layer2 only, and uvector contains $n(x)$ in layer2 only

… thereby saving 3 commands and, more importantly, 2 integer variables (xindex and yindex).

10.4.9 Math commands

Syntax:

```
math argument value
```

The math commands allows to perform mathematical operations on the script vectors. The argument is followed by a list of one to four letters form the set {a, …, z}. Uppercase or lower case do not matter; however, for clearness in the description below, we will use upper case letters when vectors are meant, and lower case letters otherwise.

If a variable is a vector, e.g. Y, it is interpreted as Yvector. If a variable is a scalar, e.g. z, it is interpreted as zvalue. If a variable is an index, e.g. w, it is interpreted as windex.

Some operations are on vectors. Then operations are performed element by element and can be performed 'in place' (e.g. A ← A+B) where the original content of A is lost.

Table 10.11   SCAPS math commands

command	argument	value remark
A,B and C represent a vector variable		
a, b, c represent a scalar variable		
i represents an index variable		
math	add	ABC $A_i = B_i + C_i$. (element-by-element) Vector operation
math	multiply	ABC $A_i = B_i * C_i$. Vector operation

math	subtract	ABC $A_i = B_i - C_i$. Vector operation
math	divide	ABC $A_i = B_i / C_i$. Vector operation
math	exp	AB $A_i = \exp(B_i)$. Vector operation
math	log	AB $A_i = \ln(B_i)$. Vector operation
math	square	AB $A_i = B_i^2$. Vector operation
math	sqrt	AB $A_i = \sqrt{B_i}$. Vector operation
math	power	ABc $A_i = B_i \wedge c$. Vector operation, A and B are vectors, but c is a scalar
math	abs	AB $A_i = \|B_i\|$. Vector operation
math	change_sign	AB $A_i = -B_i$. Vector operation
math	reciproque	AB $A_i = 1/B_i$. Vector operation
math	vectoradd	ABC $A_i = B_i + C_i$. Vector operation. Identical to add
math	vectormultiply	ABC $A_i = B_i * C_i$. Vector operation Identical to multiply
math	vectorsubtract	ABC $A_i = B_i - C_i$. Vector operation Identical to subtract
math	vectordivide	ABC $A_i = B_i / C_i$. Vector operation Identical to divide
math	vectorexp	AB $A_i = \exp(B_i)$. Vector operation Identical to exp
math	vectorlog	AB $A_i = \ln(B_i)$. Vector operation Identical to log
math	vectorsquare	AB $A_i = B_i^2$. Vector operation. Identical to square
math	vectorsqrt	AB $A_i = \sqrt{B_i}$. Vector operation. Identical to sqrt
math	vectorpower	ABc $A_i = B_i \wedge c$. Vector operation, A and B are vectors, but c is a scalar. Identical to power
math	vectorabs	AB $A_i = \|B_i\|$. Vector operation, identical to abs
math	vectorchange_sign	AB $A_i = -B_i$. Vector operation. Identical to change_sign
math	vectorreciproque	AB $A_i = 1/B_i$. Vector operation, identicalt] to reciproque

The vector and scalar scale and offset commands are new in SCAPS 3.3.07 of 4-5-2018

math	vectorscale	AB x $A_i = xB_i$. x is a scalar number in any format (a number, or a script(vector) variable). Example: `math vectorscale jj svalue`. Then each element jvector[i] will be replaced with svalue*jvector[i] for all elements of jvector

math	vectoroffset	AB x	$A_i = x + B_i$. x is a scalar number in any format (a number, or a script(vector) variable). Example: `math vectoroffset uv svector[last]`. Then uvector is created (or changed) as uvector[i] = vvector[i] + svector[last] for all elements i of vvector; last is the last element of svector. There is a space preceding x, and x has a general scriptvariable format, e.g. avector[loopcounter], or rvalue…
math	vectorsign	AB	$A_i = \text{sign}(B_i)$, thus +1, 0 or -1 when B_i >0, -0 or <0 respectively. A and B are single letters (and e.g. pq refers to pvector and qvector. Since SCAPS 3.3.12 of 25-1-2025.
math	scalarscale	ab x	$a = x * b$. x is a scalar number in any format (a number, or a script(vector) variable). [the difference with `scalaradd abc` below is that in `scalaradd`, c can only be one of the scalar script values xvalue,…, wvalue]
math	scalaroffset	ab x	$a = x + b$. x is a scalar number in any format (a number, or a script(vector) variable). [the difference with `scalaradd abc` below is that in `scalaradd`, c can only be one of the scalar script values xvalue,…, wvalue]

Examples:

To multiply all elements of xvector with 10^6, the traditional script was:

`math fillConstant  Y 1E6 nx` // creates yvector with the same number of elements as xvector, all equal to 1E6: $\text{yvector}_i = 10^6$, and ny = nx

`vectormultiply ZYX` // the result is in zvector: $\text{zvector}_i = \text{yvector}_i * \text{xvector}_i$ (and the size is nz = nx)

Now it is easier with one single new command (and saving a vector variable)

`vectorscale ZX 1E6` // the result is in zvector: $\text{zvector}_i = 10^6 * \text{xvector}_i$

… where the multiplicator can take formats such as: 1E6, xvalue, yvector[0], yvector[-1], yvector[loopcounter], …

math	scalaradd	abc	$a = b + c$. Scalar operation
math	scalarmultiply	abc	$a = b * c$. Scalar operation
math	scalarsubtract	abc	$a = b - c$. Scalar operation
math	scalardivide	abc	$a = b / c$. Scalar operation
math	scalarexp	ab	$a = \exp(b)$. Scalar operation
math	scalarlog	ab	$a = \ln(b)$. Scalar operation
math	scalarsquare	ab	$a = a^2$. Scalar operation
math	scalarsqrt	ab	$a = \sqrt{a}$. Scalar operation
math	scalarpower	abc	$a = b \wedge c$. Scalar operation.
math	scalarchange_sign	ab	$a = -b$. Scalar operation

math	scalarabs	ab	a = |b|. Scalar operation
math	scalarreciproque	ab	a = 1/b. Scalar operation
math	scalarsign	ab x	a = sign(b). Scalar operation. Since SCAPS 3.3.12 of 25-1-2025.
math	integrate	ABC	$A(B) = \int_0^B C(B')\,dB'$. Vector operation
math	differentiate	ABC	$A(B) = \dfrac{dC(B)}{dB}$. Vector operation
math	interpolate	aAbB	When A is considered as a function of B, thus $A_i = A(B_i)$, it returns by interpolation $a = A(b)$

SCAPS 3.3.03, february 2016. Four more interpolation functions: interpolate. linlin,. linlog, .loglin and. loglog

math	interpolate.linlin	aAbB	Identical with `math interpolate`
math	interpolate.linlog	aAbB	First "plot" data in a A vs. log(B) plot. Then apply `math interpolate`
math	interpolate.loglin	aAbB	First "plot" data as log(A) vs. B Then apply `math interpolate`
math	interpolate.loglog	aAbB	First "plot" data as log(A) vs. log(B) Then apply `math interpolate`
math	closestindex	iaA	Returns the index i for which the element A_i is closest to a
math	extract	ABcd	c and d are indices. The elements c up to and including d of vector B are copied into vector A, that now gets dimension d-c+1; the previous contents and dimension of A is lost. The operation can be 'in place': $A=B$ is allowed. This function is useful to pick out the information of one layer from the full x-dependence, when the indices c and d are obtained with get layer.leftindex and get layer.rightindex.
math	increment	i	The index i is incremented with one. When i is one letter from {x, y, …, w}, the index is interpreted as xindex, yindex,…, or windex. But you can also use loopcounter, maxiteration, status, mode, or one of the words xindex… written in full.
math	decrement	i	The index i is incremented with one. i is a SCAPS script index, see the above statement for the valid format.
math	min	aA	$a = \min(A_i)$
math	max	aA	$a = \max(A_i)$

math	sum	aA	$a = \sum_i A_i$
math	average	aA	$a = \dfrac{1}{n_A} \sum_{i=0}^{n_A-1} A_i$
math	median	aA	$a = A\left(i = n_A/2\right)$
math	sumofsquares	aA	$a = \sum_i A_i^2$
math	averageofsquares	aA	$a = \dfrac{1}{n_A} \sum_{i=0}^{n_A-1} A_i^2$
math	product	aA	$a = \prod_i A_i$
math	geometricaverage	aA	$a = \left(\prod_i A_i\right)^{1/n_A}$
math	chi_square	aBC DE	where B contains X_{measured} and C contains Y_{measured}; and D contains $X_{\text{calculated}}$ and D contains $Y_{\text{calculated}}$. Then chi_square is calculated as: $$\chi^2 = \frac{\sum_i \left(y_{\text{meas}} - y_{\text{calc}}\right)^2}{\sum_i \left(y_{\text{meas}}\right)^2}$$ The sum is taken at the measurement points $x_{\text{meas},i}$ that fall within the range of the calculated x_{calc} points. y_{calc} is linearly interpolated between two calculated points $x_{\text{calc},j}$ and $x_{\text{calc},j+1}$ around the measured point $x_{\text{maes},i}$. Yhe χ^2 sum is normalised: dimensionless, and should ever become small compared to 1.
math	chi_square_log		The same as chi_square but first the logarithm of (the absolute value of) all y_{maes} and y_{calc} is taken.
math	push	ABC	A = [B , C] A is a concatenation of B and C. B is placed on top of C
math	constant	ABc	A = c; Watch out: c is a scalar, A gets the same length as B. B is only used to set the length of A. AAc is allowed as well.
math	linear	AB	A = [1, 2 ,3…]; A gets the same length as B. B is only used to set the length of A. AA is allowed as well.
math	rangeLin	A	The first point A[0] and the last point A[nA-1] of the vector A are conserved. The points in between are

			scaled in a linear way between those end points.
math	rangeLog	A	The first point A[0] and the last point A[nA-1] of the vector A are conserved. The points in between are scaled in a logarithmic way between those end points.
math	characteristics.eta	aBC	Vector B contains the voltage data, vector C the current density data of a calculated *IV* curve. Scalar a will then contain η or V_{oc} or J_{sc} or *FF* or V_{mpp} or J_{mpp} (*V* and *J* at the maximum power point). If you acquired the *IV* curve with `get iv`, then you can better use `get characteristics.eta…` (the scalar that should receive e.g. η in get.characterisctics can have the general SCAPS script format, like xvalue, yvector[0], zvector[loopcounter]…).
math	characteristics.Voc		
math	characteristics.Jsc		
math	characteristics.FF		
math	characteristics.Vmpp		
math	characteristics.Jmpp		
			Here, with math.characteristics, the result can only be stored in the scalars xvalue, yvalue,…. However, this function is useful if the *IV* curve was processed in the script (from series or parallel connection, all kinds of corrections made to IV curves, …)
math	characteristics.all	ABC	Works the same as the previous commands, but the results are all stored in vector A. The order is fixed: A[0] = V_{oc}, A[1] = J_{sc}, and then *FF*, η, V_{mpp} and J_{mpp}.

The five math commands below are special: they require a composed value. The first part of the `fill…` commands is a vector: a letter from {x, y, z, u, v, w} (noted here as A), that stands for the corresponding script variable xvector, yvector,…. The next parts of the value must be separated by whitespace (space of tab) from the first part and from each other. They can be a number, or a SCAPS script variable. The first part of the `force_in_range` command is a scalar: : a letter from {x, y, z, u, v, w} (noted here as a), that stands for the corresponding script variable xvalue, yvalue, … The format of the `series` and `parallel` commands are explained below.

math	fillConstant	A constant n	n is the number of points
math	fillLinear	A startvalue stopvalue n	n is the number of points
	fillinear.FixedNumberOfPoints		
math	fillLinear.FixedIncrement	A startvalue stopvalue Δx	Δx is the increment
math	fillLogarithmic	A startvalue stopvalue n	n is the number of points
	fillLogarithmic.FixedNumberOfPoints		
math	fillLogarithmicPerDecade	A startvalue stopvalue n	n is the number of points per decade
	fillLogarithmic.PerDecade		

math	fillLogarithmic.FixedMultiplyer	A startvalue stopvalue q	q is the multiplier
math	force_in_range	a minvalue maxvalue	the value of the scalar a is forced in the range [minvalue, maxvalue]
math	clip_above_top	AB x	SCAPS>3.3.12 of 24-1-2025: A and B (one letter only) refer to two script vectors; x (preceded by a space) is a script scalar in general format. It copies B into A, but replaces the elements > x with x (for …above_top), or the elements < x with x for … below_bottom
math	clip_below_bottom	AB x	
math	coerce_in_interval	AB x y	SCAPS>3.3.12 of 24-1-2025. A and B are two scriptvectors (one letter only), x and y (preceded by a spave) two general format scalars. Copies B into A, but forces all elements to be in the range [x, y]. Works like force_in_range, but is for vectors, not for scalars, and it does not work necessarily "in place", that is, the output vector A can be different from the input vector
math	series	ABCDEF n	The script vectors CD and EF contain the *IV* curves of two cells (voltage in C and E, current density in D and F). The *IV* data of the series connection is placed in the script vectors AB (voltage in A, current density in B); these vectors Avector and Bvector will contain *n IV*-points.
math	parallel	ABCDEF n x y	The script vectors CD and EF contain the *IV* curves of two cells (voltage in C and E, current density in D

and F). The area of cell 1 is multiplied with the scalar weight factor x, the area of the second cell with the weight factor y (only $x+y=1$ really makes sense). The *IV* data of the parallel connection is placed in the script vectors Avector and Bvector (voltage in A, current density in B);there will be n *IV*-points.

Examples:

math `fillinear` x 1 3 5 results in xvector $= [1.0,1.5, 2.0, 2.5, 3.0]$

math `fillinear.fixednumbeofpoints` y 2 8 4 reults in yvector $= [2, 4, 6, 8]$

math `fillinear.fixedincrement` z 1 5 2.0 results in zvector $= [1, 3, 5]$

math `fillinear.fixedincrement` z 1 6 2.0 results in zvector $= [1, 3, 5, 6]$. Note that the last increment is not the 'fixed' one (=2), but that the stop value (=6) is respected instead.

math `filllogarithmic` u 10 1000 5 results in uvector $\approx [10, 31.6, 100, 316, 1000]$

math `fillogarithmic.perdecade` v 10 2000 2 results in vvector $\approx [10, 31.6, 100, 316, 1000, 2000]$

math `filllogarithmic.fixedmultiplyer` w 1 20 2.0 results in wvector $= [1, 2, 4, 8, 16, 20]$. Note that the last multiplier is not the 'fixed' one (=2), but that the stop value (=20) is respected instead.

Since SCAPS 3.3.12 of 22-1-2025, some conditional math commands were added, these are the if-commands below.

math	`if.positive`	x y z	x y and z (separated by a space) refer to 3 scalar scriptvariables with the general format. Action: if($x > 0$), then y←z (the value of z is assigned to y). And alike for the command that check if($x<0$) and if($x=0$).
math	`if.negative`	x y z	
math	`if.zero`	x y z	
math	`if.greater`	x y z u v	x y, z, u and v (separated by a space) refer to 5 scalar scriptvariables with the general format. Action: if($x>y$), then z←u , and if not, then z←v (thus, an `if then else` statement is implemented). And alike to check $x<y$ and $x=y$.
math	`if.lower`	x y z u v	
math	`if.equal`	x y z u v	

SCAPS 3.3.10, March 2021: Four new math-commands are introduced to support the study of tandem cells. The details of their working principles, and application examples are given in a next Chapter.

math	`split_tandem_cell.only_split` n_1 n_2
math	`split_tandem_cell.adapt_contact_filters` n_1 n_2
math	`split_tandem_cell.adapt_generation` n_1 n_2
math	`split_tandem_cell.with_electronically_inactive_parts` n_1 n_2

The summary of their working is:

o the actual cell problem is saved as *temporary tandem cell definition file.def*
o this "tandem cell" is split between layers n_1 and n_2 into a top cell and a bottom cell (n_1 +1 = n_2 is required)
o some strategy for this is applied (e.g. 'adapt generation').
o these single cells are saved as *temporary top cell definition file.def* and *temporary bottom cell definition file.def*.
o when applicable, other relevant files (filter files, generation files) are saved with an appropriate 'temporary' name.

10.4.10 Loop commands

Syntax:

```
loop argument variable
```

On encountering a `loop start` command line, the internal script variables are set to: loopcounter = 0 and looperror = 1.0E30 (or the value of looperror that was set before).

The next script commands are executed until `loop stop` is met. Then, if loopcounter < maxiteration-1 <u>and</u> looperror > maxerror, the internal script variable loopcounter is incremented, and the script is retaken from the preceding `loop start` command. Thus, when the error condition is never met, loopcounter will successively be set to 0 … maxiteration-1, thus maxiteration values. The internal variables maxiteration and maxerror can be set with `set loop.maxiteration` and `set loop.maxerror` at any time, see §10.4.6.

There is no `set` command to set the internal script variable loopcounter. The variable loopcounter is internally set to zero on starting a loop, and then incremented with one each times the loop is run. The variable looperror can be set directly or be returned by the dll program SCAPSUserFunction.dll (§10.4.14), that should be set-up by the user (one example of such dll is distributed with the SCAPS installation). Two of the `loop` commands are equivalent with a set command:

E.g. `loop maxiteration 20` is equivalent to `set scriptvariable.maxiteration 20`

E.g. `loop maxerror 1E-6` is equivalent to `set scriptvariable.maxerror 1E-6`

Table 10.12   SCAPS loop commands

command	argument	value	default-directory
`loop`	`start`	no value	
`loop`	`stop`	no value	
`loop`	`maxiteration`		min=5; max=100; default = 25.
`loop`	`maxerror`		min=1E-8; max=1E25; default=1E-5

10.4.11 Show command

There is only one show command. The command line to do this is:

```
show scriptvariables
```

These are shown on the screen (see §Figure 10.3), if `errorhandling.toscreen` is set, or to the standard error file, if `errorhandling.appendtofile` or `errorhandling.overwritefile` are set. This command is very useful in debugging your script files. Also, you can stop the execution of a script when the script variables are shown on the screen. When scripts are nested, you exit all scripts upon clicking

'stop script execution'. You can comment out the `show` commands once the script is OK. The `show scriptvariables` panel is also available from the action panel (the SCAPS main panel) after execution of a script. The `show` command does not show all values of (very) long vectors, if the output is directed to the standard error file. In order to access these, you should use `save scriptvariables`.
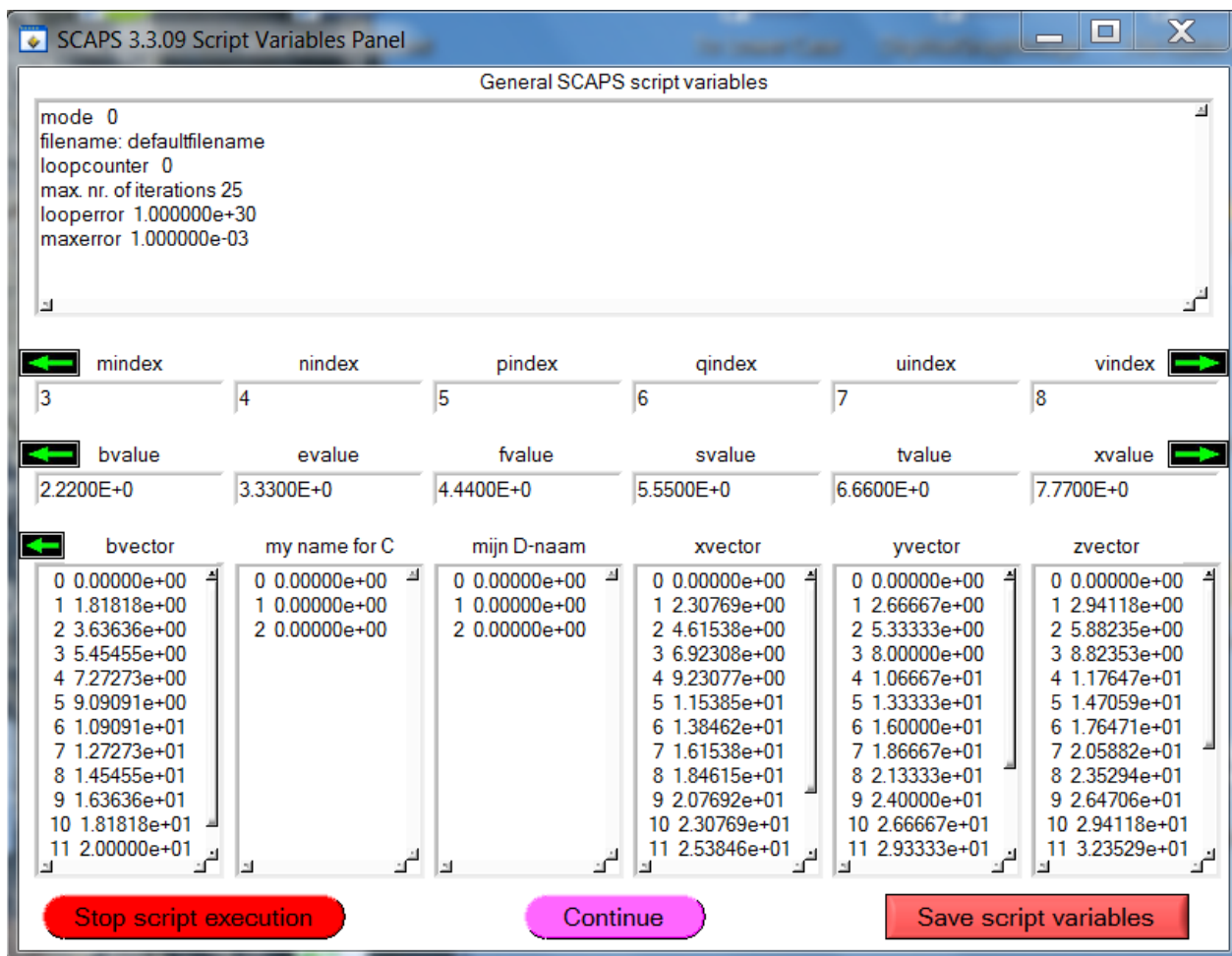

Figure 10.3 The script variables panel. When the panel is displayed after a `show scriptvariables` command, the script is interrupted and can now be aborted or continued. SCAPS ≥ 3.3.09, december 2020: a few changes since up to 26 index, value and vector variables could need to be shown: only the variables that were addressed in the script (either as input or output) are shown, the others are hidden; there are left and right arrows to let you scroll if there were more than 6 variables of a type (index, value or vector) that were addressed.

10.4.12 Plot commands

Syntax:

```
plot argument value
```

The plot command works in a similar way as the math command §10.4.9.

Table 10.13   SCAPS plot commands

command	argument	value	remark
A,B and C represent a vector and should be chosen out of the set {x,y,z,u,v,w}			
plot	draw	AB	Plot A on the abscissa and B on the ordinate
plot		AB	Identical to `plot draw`
plot	drawversusindex	A	Plot the index i on the abscissa and A_i on the ordinate

plot	clear	Clear the plots drawn by the script. Identical to `clear plot`

Whenever a `plot draw` command was met, a graph is added to the 'script plot panel', see Figure 10.4. This option allows you to make personalized graph of any data in SCAPS ☺. This panel is shown immediately after the script calculation is finished and can also be accessed from the action panel.
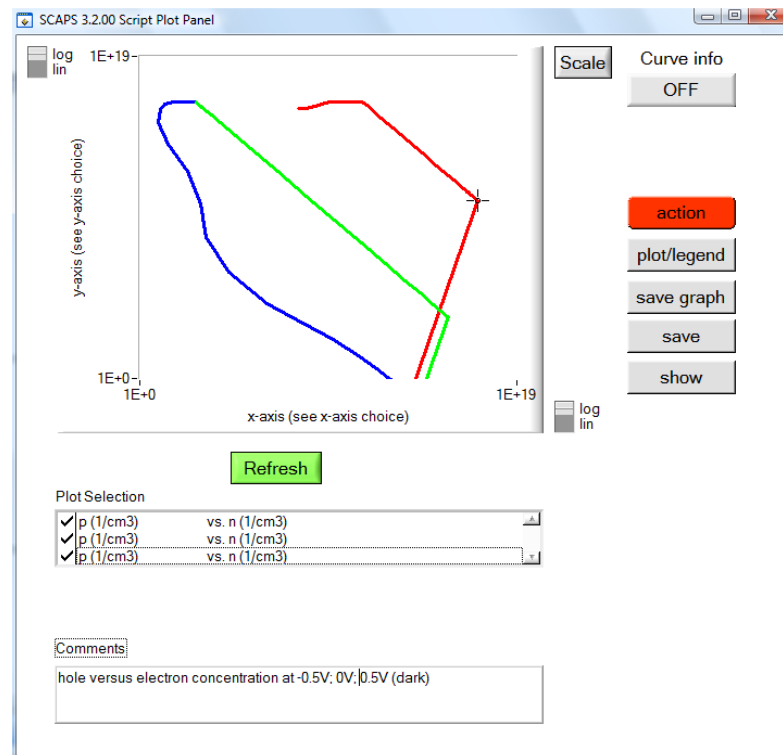

Figure 10.4   The script plot panel

10.4.13 Calculate commands

Syntax:

`calculate argument`

This is equivalent with pressing one of the "Calculate"-buttons in the interactive user interface. If no argument is present the command gets interpreted as `calculate singleshot`

command	argument	remark
calculate	singleshot	this argument can be omitted
calculate	batch	
calculate	recorder	
calculate	equilibrium_only	(SCAPS 3.3.10, march 2021) dark and $V=0$ only; this is just enough to define the mesh and all cell properties. After completion the action list is restored to its original state (workpoint and illumination settings, IV, C-V… settings)
calculate	short_circuit_only	(SCAPS 3.3.10, march 2021) light and $V=0$ only; this is just enough to calculate the illumination parameters (lightflux, generation, …). After completion the action list is restored to its original state (workpoint and illumination

settings, *IV*, *C-V…* settings)

10.4.14 The run commands

10.4.14.1   Running SCAPSUserFunction.dll or SCAPSUserFunctiondll26

These function are run by

`rundll scapsuserfunction` or	`rundll26 scapsuserfunction` or
`run dll scapsuserfunction` or	`run dll26 scapsuserfunction` or
`run dll`	`run dll26`

Traditionally (SCAPS ≤ 3.3.08), only one user dll is recognized in SCAPS, named SCAPSUserFunction.dll.

With SCAPS ≥ 3.3.09, a second user dll is recognised, named SCAPSUserFunction26.dll.

The format of these commands allows possible later addition of more dll's).

   This dll is the method that SCAPS is using to implement two-way communication with the user. When you do not (want to) know how to write an own program and make a dll (dynamic link library) of it, you are restricted to use only the SCAPSUserFunction.dll and/or SCAPSUserFunction26.dll as delivered with SCAPS, or not to use loops in a SCAPS script. The following information is for SCAPS users with programming skills. By writing their own SCAPSUserFunction.dll, they now can realize the following (in the formulation of an external SCAPS user):

   "I would need the possibility to do a simulation, evaluate the result with an external program and let it adjust the problem definition for the next simulation, do a simulation, and so on…"

… well, this external program should be named SCAPSUserFunction, and be present as a dll file in the scaps/lib directory. When implemented in C or C^{++}, this function must comply with the function definition:

```
int DLLIMPORT SCAPSUserFunction (int mode, double *xvalue, double
*yvalue, double *svector[6], int sn[6], double *looperror, char
*filename);
```

and:

int DLLIMPORT SCAPSUserFunction26 (int mode, int sindex[26], double svalue[26], double *svector[6], int sn[26], double *looperror, char *filename);

   The keyword DLLIMPORT might be dependent on the development environment; here it is for LW/CVI of National Instruments.

   The meaning of the other items is:

- SCAPSUserFunction: the name of the traditional dll. The user must provide a SCAPSUserFunction.dll and SCAPSUserFunction.lib with this name, in the scaps/lib directory. This function only accepts and possibly changes the SCAPS script variables xvalue, yvalue, xvector, yvector, zvector, uvector, vvector and wvector; and also mode, looperror and filename… but no indices are passed (such as xindex, yindex), and no other values of vectors. SCAPS ≥ 3.3.09: this function still works, and is implemented as explained below, even if there are now more variables defined

- SCAPSUserFunction26: the name of the new dll. It passes all scriptvariables: index, value and vector with letter a-z. At the moment (17-12-2020), this function is empty, it returns without having done anything.

- int SCAPSUserFunction(): the function should return an integer value, indicating the success of the function evaluation. SCAPS interprets 0 as 'success' and a negative value as a failure. This value is stored in the script-variable status, and shown in the error output (to screen or in the SCAPS error logfile).

- int mode: an integer that can be used to implement several strategies in one dll function. In the example delivered with SCAPS, mode = 1 or 2 means 'find a root' (e.g. find some N_A such that V_{oc} = 0.50 V), and mode 3 or 4 means 'find an extremum' (e.g. find some N_t such that η is maximal).

- double *xvalue, double *yvalue: (pointers to) two scalar values, passed to the function by reference, such that a new value of them can be returned by the function. Note with SCAPS 3.0.02: though there are now 6 scalar values xvalue, …, wvalue, only xvalue and yvalue are passed to the SCAPS dll. Also, the new integer variables xindex, …, windex, are not passed to the dll: thus, this dll remains compatible with earlier SCAPS versions.

- double *svector[6]: array of (pointers to) one dimensional arrays, with dimensions specified in sn[]. These vectors correspond to the vectors xvector (=0), yvector (=1), zvector (=2), uvector (=3), vvector (=4), wvector (=5). These arrays can get new values in the function that is returned to SCAPS.

- int sn[6]: the dimension of the above vectors. These are passed by value, not by reference: their value cannot be updated and returned by the function.

- double *looperror: a pointer to a scalar variable, that can be updated and returned by the function. In the SCAPS script processor, ot is treated as the internal looperror variable. Returning its value by SCAPSUserFunction.dll is the only way to change looperror in a loop. Since the script processor only checks if |looperror| < maxerror, so you can also return a negative value here.

- char *filename: a pointer to a string variable of max. 256 characters. The SCAPS script processor will treat it as a filename, that can be used to set e.g. a spectrum file, a generation file, a filter file,… with the set command.

To set up your own dll, you can use other variable names; however, the type, size and order of the variables must be exactly as specified here. Those not using C or C^{++} should use variable types of the same size (in bits) as the C types int, double, char, pointer.

The users who want to develop their own dll, or to modify the existing dll (that is easier to start with ☺), should ask us for the source code: SCAPSdll.c and SCAPSdll.h.

10.4.14.2 Executing system commands in a script

The command line to do this is:

```
runsystem systemcommand or
run system systemcommand
```

where systemcommand is something that is recognized by MS-Windows as a valid command. These can be .exe files, .bat files or WINDOWS commands. Here you can run any of your own programs (extension .exe; the arguments on the command line can be included), or any of your batch files (extension .bat).

Examples are:

```
runsystem myownopticalprogram.exe inputfile1 inputfile2 outputfile
runsystem myownwindowsbatchprogram.bat
runsystem print ivresults.iv
```

(in the last command, it is likely that Windows will need to know the full path and not only the filename…).

10.4.14.3   <u>Executing a script from within a script</u>

The command line to do this is:

```
run script scriptfile
```

where `scriptfile` is a file containing a script. You can nest script files (that is, run a script file from within another script file) as you like, but that should not be a reason for exaggeration. All the script variables are transferred from one script to the other, with the exception of some loop-variables loopcounter, looperror, maxerror, maxiterations, which are local to each script file.

Chapter 11: Script support for simulating multi-junction (tandem) cells

11.1 Introduction

The SCAPS application discussed in this document makes use of:

- SCAPS version 3.3.10 of March 2021, or more recent.

- The following files:

  Definition files that must be present in the folder [your SCAPS]\def:

  - ✓ any tandem cell (thus, a *pnpn* sequence, or an *npnp* sequence. Examples of these, delivered with SCAPS are:
  - ✓ `test tandem cell scripts illum from left.def`
  - ✓ `test tandem cell scripts illum from right.def`
  - o Script files that must be present in folder [your SCAPS]\script:
    - ✓ `test tandem split generation.script`
    - ✓ `test 2 adapt filters varying Rb.script`
    - ✓ `test tandem split fake top and bottom.script`

11.2 Problems with the direct simulation of tandem cells in SCAPS

SCAPS was originally intended to study single cells of CIGS or CdTe type. Also, the original program was conceived to work well for a *pn* structure, where the leftmost layer is *p*-type, and the rightmost layer is *n*-type, and there is only one *p-n* junction (however, several *p* layers and several *n*-layers could be present).

Indeed SCAPS runs well for such structures, even with parameters that are very different from CdTe or CIGS: it has been successfully applied to CZTS, perovskite, c-Si, a-Si, bulk heterojunction, … solar cell structures. Unfortunately, SCAPS often fails to converge when either more then one *pn* junction is present (e.g. a *pnp* or *pnpn* structure), or when there are too strong reverse junctions at a contact. The convergence could be improved (a little) by setting tunnel mechanisms on, or by playing with convergence settings (e.g. more iterations; adaptive mesh,...) ... but often these tricks do not remedy the basic convergence problem.

This SCAPS problem is well known, and persisting since the very birth of SCAPS. However, it never was on our priority list to work on tandem cells, as we wanted to enhance SCAPS with more physics of single thin film cells, and we never worked on tandems, let be (real) multiple junctions. Recently, some work was done on the convergence problem, but so far without any tangible success.

In recent years, SCAPS users are increasingly interested in simulation of multi-junction cells, mainly tandem cells based on perovskite materials. To serve these users, even when direct simulation of tandem cells is most often not possible in SCAPS, we developed script support to split a tandem structure into partial cells (thus, a top cell and a bottom cell), to study the partial cell under 'tandem illumination conditions', and to assemble the tandem cell again, this is, to place the two partial cells electrically in series. This chapter can be read as an application note on this topic.

11.3 Basic strategy: calculate top and bottom cell separately, place them in series afterwards

For the time being (and perhaps a little bit longer), those wanting to study tandem cells should remedy themselves with tricks: you can calculate the top and bottom cell separately, and series connect them yourself in an external program or with a SCAPS script. Of course, this is by far not as user friendly as a direct calculation within SCAPS...

In the study of the partial cells (thus, top and bottom) the user should take care that these partial cells are studied under the illumination conditions they have in a tandem cell configuration. There is now script support for:

- splitting a tandem cell configuration into a top and bottom cell

- three strategies to study these partial cells under "tandem-light conditions"

- making the electrical series connection of these partial cells to obtain the *IV* characteristics of a 2-terminal tandem cell.

11.4 Splitting the tandem cell into top and bottom cells

Splitting a tandem cell configuration manually into a top cell and a bottom cell is easy but a little bit laborious, and maybe boring should you have to do it regularly. Therefore, this action is now automated with the script command:

```
math split_tandem_cell.only_split n1 n2
```

This script command first saves the actual cell problem in a file named *temporary tandem cell definition file.def*. Then, this "tandem cell" is split between layers n_1 and n_2 into a top cell and a bottom cell (with the requirement $n_1 +1 = n_2$). SCAPS takes account of the side where the light is incident. Thus, when light is incident from left, the top cell will contain the original layers 1 to n_1, and the bottom cell the original layers $n_2(=n_1+1)$ to N_{layers}. And similarly for light incident from the right. It might be redundant to ask the user to specify two values n_1 and n_2, but it forces her/him to think well about the tandem cell configuration. When the parameters n_1 and n_2 are omitted (or when only one is specified), SCAPS will make an own guess of where to split the tandem, and notify you about this with a warning. The tandem split strategy of SCAPS is simple: it starts looking to the doping type of the leftmost layer, and goes on to the right. When a second type change ($n{\to}p$ or $p{\to}n$) is met, the split position is placed there. This algorithm will be fooled when truly intrinsic layers are present (thus, where the shallow $N_D = N_A$, exactly), or when the doping is modelled by charged, not-too-deep defects instead of by ideally shallow donors/acceptor. Therefore, it is better that you express your ideas on the split position (between layers n_1 and n_2), than to leave it to SCAPS.

The command … .only_split just does the splitting, and nothing more. The other tandem split commands (see below) also implement a strategy about the illumination/generation in the partial cells. In all cases, the partial cell definitions are saved in your [SCAPS]\def directory, with names *temporary top cell definition file.def* and *temporary bottom cell definition file.def*. You can use these def files to express your own ideas/strategies on the illumination/generation conditions. To load these files, you can use e.g.

```
load definitionfile temporary top cell definition file.def
```

but then you will have to remember (or copy) the exact filename of the partial cells. Instead, you can use:

```
load definitionfile.temporary.tandem_cell
load definitionfile.temporary.top_cell
load definitionfile.temporary.bottom_cell
```

Since the script editor is proposing the correct form of these commands, the risk of mistyping is minimised.

11.5 Strategy 1: 'adapt generation'

When using

```
math split_tandem_cell.adapt_generation n1 n2
```

SCAPS is doing the same as with …only_split, but before splitting the tandem cell, a generation profile for the tandem cell and for each of the sub-cells, the top cell and the bottom cell, is calculated. First, the *eh* generation profile $G_{\text{tandem}}(x)$ in the tandem cell is determined, and saved as file *temporary tandem cell G(x).gen* in the [SCAPS]\gen directory. Then, this generation profile $G_{\text{tandem}}(x)$ is split between a generation profile $G_{\text{top}}(x)$ for the top cell and one for the bottom cell $G_{\text{bottom}}(x)$. For example, if x_t is the position of the top/bottom split (thus the position of the right side of layer n_1 = the left side of layer n_2, when n_1 and n_2 are

the value parameters of the `math split_tandem` script command), and when light is incident from the left, then

$$G_{\text{top}}(x) = G_{\text{tandem}}(x) \qquad 0 \leq x \leq x_t$$

$$G_{\text{bottom}}(x) = G_{\text{tandem}}(x + x_t) \quad 0 \leq x \leq d_{\text{tandem}} - x_t$$

<div align="right">(34)</div>

where d_{tandem} is the total thickness of the tandem cell. These generation profiles are saved as files *temporary top cell G(x).gen* and *temporary top cell G(x).gen* in the [SCAPS]\gen directory. These generation files are however <u>not</u> stored in the definition files for the top and bottom cells, as SCAPS definition files do not store information on illumination conditions.

To use the definition and generation files of a partial cell (top or bottom), one thus should load the definition file and the generation file of that partial cell, and set the illumination condition to 'Directly specify G(x)' in the SCAPS action panel. In a script, this is automated as shown in the example below, for the top cell:

```
load definitionfile test tandem cell scripts illum from left.def
```

// we want the generation calculations to be done based on an incident spectrum

```
action lightflux_specified
math split_tandem_cell.adapt_generation 2  3 // the tandem split is between layers 2 and 3
action light
```

// to use the generation profiles constructed by `split_tandem`, the generation should be directly specified from file

```
action generation_specified
load definitionfile.temporary.top_cell
load generationfile.temporary.top_cell
```

(note that there is a script command to load a generation file for tandem, top or bottom cell, as there was such a script command for loading def files).

The generation conditions for the partial cells will be exact within the SCAPS optical model. They are only valid for the spectrum and intensity that was set in the Action Panel, and for the optical properties of the contacts ($R(\lambda)$ or $T(\lambda)$ filters), and for the internal reflection at the front side, and for the optical absorption $\alpha(x, \lambda)$ and thickness that was set in each layer. However, this advantage of exactness has its price: should the user, while studying or optimising a partial cell, change any of the optical relevant parameters (a thickness or band gap of a layer is enough), the generation profile is no longer exact, and the above procedure should be repeated: start from the tandem, set the generations $G_{\text{tandem}}(x)$, $G_{\text{top}}(x)$ and $G_{\text{bottom}}(x)$, and split the tandem.

Some users have developed much more elaborated optical models for the calculation of $G(x)$ profiles then the SCAPS optical model, that is rather coarse. Their optical calculation goes through own or dedicated programs, and a generation profile file calculated in this way can be passed to a SCAPS script manually or automated e.g. via MatLab. This is definitely a much more sophisticated and (optically) exact way to do tandem simulations – but it is more laborious…

11.6 Strategy 2: 'adapt the optical filters at the contacts'

When using

```
math split_tandem_cell.adapt_contact_filters n1 n2
```

SCAPS is doing the same as with `…only_split`, but before splitting the tandem cell, new contact filter files are defined, containing reflection $R(\lambda)$ or transmission $T(\lambda)$, for each of the sub-cells, the top cell and the bottom cell. These adapted filter files are then included in the definition files for the top cell and bottom cell.

The internal optical model of SCAPS was discussed in Section 4.3.1 and resulted in Eq. (15), that we repeat here in Eq. (35) for reading comfort.

$$N_{\text{phot}}(\lambda, x) = N_{\text{phot0}}(\lambda) \cdot T_{\text{front}}(\lambda) \cdot \exp(-x\alpha(\lambda)) \cdot \frac{1 + R_{\text{back}}(\lambda)\exp(-2(d-x)\alpha(\lambda))}{1 - R_{\text{back}}(\lambda)R_{\text{int}}\exp(-2d\alpha(\lambda))} \tag{35}$$

This equation is for a light flux $N_{\text{phot0}}(\lambda)$ incident from the left, on a cell with a uniform, but wavelength dependent optical absorption constant $\alpha(x, \lambda) = \alpha(\lambda)$. $T_{\text{front}}(\lambda)$ is the wavelength dependent transmission at the front contact (here: left), and $R_{\text{back}}(\lambda)$ is the wavelength dependent reflection at the back contact (here: right). R_{int} is the internal reflection at the front contact (here: left); no wavelength dependence for R_{int} is implemented in SCAPS. Of course, SCAPS takes account of different $\alpha(\lambda)$ functions in the different layers (set by an absorption file or an absorption model), and correctly treats grading, thus the possibility that within a layer, α is also position dependent: $\alpha(x, \lambda)$. This extension is rather trivial: simple exponential forms are replaced with a form with an integral in the exponent:

$$\exp\left[-\alpha(\lambda) \cdot x\right] \qquad \rightarrow \qquad \exp\left[-\int_0^x \alpha(x', \lambda)dx'\right]$$
$$\exp\left[-\alpha(\lambda) \cdot (d-x)\right] \qquad \rightarrow \qquad \exp\left[-\int_x^d \alpha(x', \lambda)dx'\right] \tag{36}$$

In this discussion, we will go on with the simple formulation of Eq. (35), in order not to obscure the argumentation by complicated equations. The treatment below will only be exact when there is no internal reflection in the cell, thus $R_{\text{int}} = 0$. Then Eq. (35) reduces to Eq. (37) below

$$N_{\text{phot}}(\lambda, x) = N_{\text{phot0}}(\lambda) \cdot T_{\text{front}}(\lambda) \cdot \exp(-x\alpha(\lambda)) \cdot \left[1 + R_{\text{back}}(\lambda)\exp(-2(d-x)\alpha(\lambda))\right] \tag{37}$$

This Eq. (37) holds for all x in the tandem cell, thus $0 \leq x \leq d$. Assume that the tandem split is at position x_t, thus that the top cell is ranged in $0 \leq x \leq x_t$, and the bottom cell is ranged in $x_t \leq x \leq d$, see Figure 11.1.

The light flux in the top cell can now be written as (with $d = x_t + d'$, d' is the thickness of the bottom cell):

$$N_{\text{phot}}(\lambda, x) = N_{\text{phot0}}(\lambda) \cdot T_{\text{front}}(\lambda) \cdot \exp(-x\alpha(\lambda)) \cdot \left[1 + R_{\text{back}}(\lambda)\exp(-2d'\alpha)\exp(-2(x_t - x)\alpha)\right]$$
$$N_{\text{phot}}(\lambda, x) = N_{\text{phot0}}(\lambda) \cdot T_{\text{front}}(\lambda) \cdot \exp(-x\alpha(\lambda)) \cdot \left[1 + R'_{\text{back}}(\lambda)\exp(-2(x_t - x)\alpha(\lambda))\right] \tag{38}$$
$$R'_{\text{back}}(\lambda) = R_{\text{back}}(\lambda)\exp(-2d'\alpha(\lambda))$$

Thus, we can treat the top cell on its own (thus without bottom cell attached to it), when we keep T_{front} as is, and replace $R_{\text{back}}(\lambda)$ with $R'_{\text{back}}(\lambda)$ as given in the third equation (38).

In the same way, we can write the light flux in the bottom cell (with $x' = x - x_t$) as:

$$x_t \leq x \leq d \quad \text{or} \quad 0 \leq x' \leq d'$$
$$N_{\text{phot}}(\lambda, x) = N_{\text{phot0}}(\lambda) \cdot T_{\text{front}}(\lambda) \cdot \exp(-x'\alpha)\exp(-x_t\alpha) \cdot \left[1 + R_{\text{back}}(\lambda)\exp(-2(d' - x')\alpha)\right]$$
$$N_{\text{phot}}(\lambda, x) = N_{\text{phot0}}(\lambda) \cdot T'_{\text{front}}(\lambda) \cdot \exp(-x'\alpha) \cdot \left[1 + R_{\text{back}}(\lambda)\exp(-2(d' - x')\alpha)\right] \tag{39}$$
$$T'_{\text{front}}(\lambda) = T_{\text{front}}(\lambda)\exp(-x_t\alpha(\lambda))$$

Thus, we can treat the bottom cell on its own (thus without top cell attached to it), when we keep R_{back} as is, and replace T_{front} with T'_{front} as given in the fourth equation (39).

Note that this strategy fails when $R_{\text{int}} \neq 0$. And of course, SCAPS will not use the simple exponential forms of Eqs. (38) and (39), but the more elaborate forms of Eq. (36). The new back contact reflection $R'_{\text{back}}(\lambda)$ of

the top cell alone is saved in [SCAPS]\filter as file *temporary top cell back reflection.ftr* and included in *temporary top cell definition file.def*; and similarly for the new front transmission T'_{front} of the bottom cell.
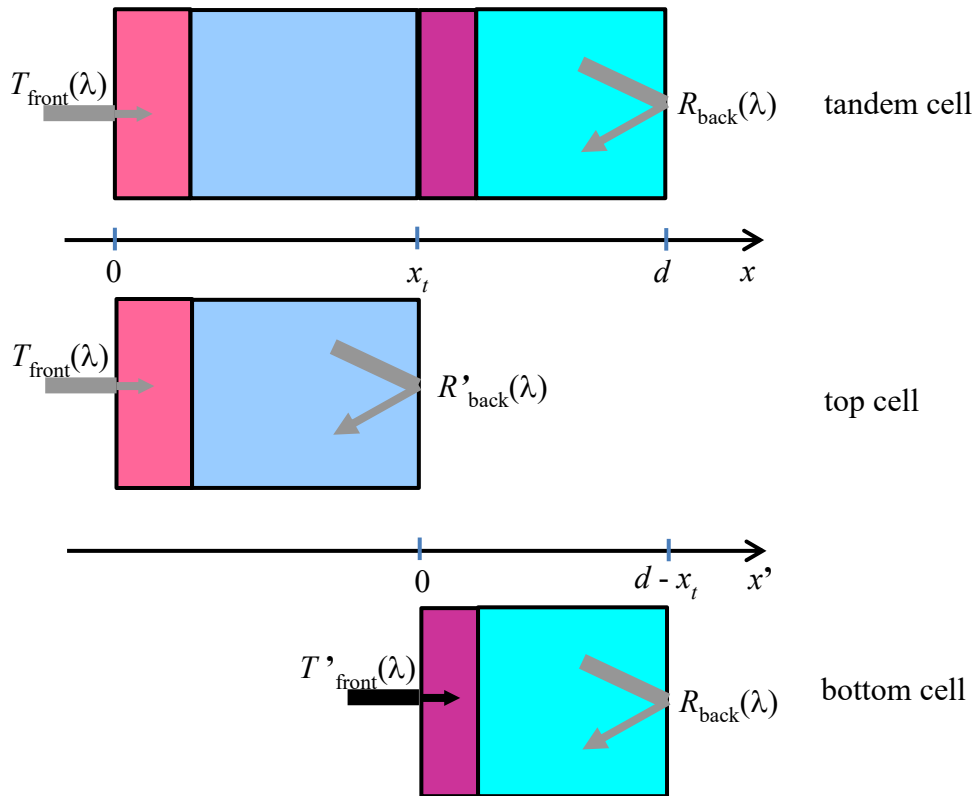

Figure 11.1 Splitting a tandem cell into a top cell and a bottom cell. The top cell has the original $T_{front}(\lambda)$, but an adapted $R'_{back}(\lambda)$. The bottom cell has an adapted $T'_{front}(\lambda)$, but the original $R_{back}(\lambda)$.

These optical conditions for the partial cells (thus the filters T'_{front} and $R'_{back}(\lambda)$) will be exact within the SCAPS optical model. They will remain valid when the settings of the spectrum and intensity in the Action Panel are changed (new spectrum model parameters, new spectrum file, new settings for neutral density filter and spectrum cut-off filters), and this is an advantage over the previous strategy ('adapt generation'). It could be a good idea to set the spectrum to a simple, smooth model (e.g. monochromatic with constant photon flux at all wavelengths) when running the script command `split_tandem.adapt_contact_filters` (then you get rid of the peaks and dips that are present in realistic spectrum files – due to our good old sun, and especially to our good old earth atmosphere). Afterwards, set the spectrum conditions to whatever you want to study. However the calculated filters T'_{front} and $R'_{back}(\lambda)$ will have to recalculated when the user is changing optical properties of the contacts ($R(\lambda)$ or $T(\lambda)$ filters), or the optical absorption $\alpha(x, \lambda)$ and thickness that was set in each layer. Also, the method cannot be applied when $R_{int} > 0$; however, when the layers reasonably absorb the light (optical α and thickness sufficiently high), the influence of R_{int} will be negligible anyway.

11.7 Strategy 3: 'replace parts of the cell with electronically inactive layers'

This is the strategy that was proposed in a SCAPS FAQ or application note "Assembling a tandem cell from two single cells in SCAPS", february 2016. It is now automatised and implemented in the script command

```
math split_tandem_cell. with_electronically_inactive_parts n1 n2
```

Now SCAPS does not really 'split' the tandem cell, but is setting up a top cell and a bottom cell that both consist of all layers of the tandem cell.

In setting up the top cell, the layers that belong to the top cell are taken as they are defined. However, the electronic properties of the bottom cell layers are changed so that they remain optically active, but are made 'electronically inactive'. Assume that light is incident from the left, so that the top cell is the left part, and the bottom cell the right part of the tandem. Also assume that the rightmost layer of the top cell (thus the layer adjacent to the bottom cell) has an effective n-type doping. To render the bottom cell 'electronically inactive', some properties are changed, of all layers of the bottom cell:

1. the layers are made (relatively) heavily doped n-type: the script sets $N_D = 10^{19}$ cm^{-3} and $N_A = 10^1$ cm^{-3}.

2. discontinuities in the conduction band edge E_C are removed, by setting the electron affinity χ of all bottom cell layers equal to the χ value of the rightmost edge of the rightmost layer of the top cell.

3. the contact velocities S_n and S_p of the right contact are set to $S_n = S_p = 10^7$ cm/s; and the flat-band feature of this contact is clicked on.

4. the layers are given a very high recombination:

   - all defects are removed

   - a new defect 1 is set up, with very high efficiency for recombination

   - this is a neutral defect, single level at mid gap

   - with $\sigma_n = \sigma_p = 10^{-12}$ cm^2, and $N_t = 10^{17}$ cm^{-3}.

Action (1) ensures that the 'electronically inactive' bottom cell does not add an extra pn junction to the top cell being set-up. Together with actions (2) and (3), it ensures that convergence will (almost) always be reached: the bottom cell acts more or less as a thick ohmic contact to the top cell. At the same time, nothing has changed to the optical behaviour of the bottom all: all layers have their $\alpha(x, \lambda)$ as set in the tandem definition, and also $R_{back}(\lambda)$ at the back contact remains unchanged. Action (4) ensures that (almost) no electrons or holes of the eh pair generation in the bottom cell will contribute to the current: they all will recombine before reaching the top cell or the right contact. Of course this strategy is adapted if the right most layer of the top cell is p-type, and/or if light is incident from the right side. The partial cells are saved automatically as `temporary top cell definition file.def` and `temporary bottom cell definition file.def`, as before.

[in our application note of 2016, we called this strategy 'top cell with *fake* bottom cell', 'bottom cell with *fake* top cell'; since the word '*fake*' has got an utmost unscientific connotation since that time, we now use the terminology: 'top cell with optically unchanged but electronically inactive bottom layer', …].

This strategy is illustrated in Figure 11.2 where the energy band diagrams of the complete tandem cell, the top-cell-with-electronically-inactive-bottom-cell, and the bottom-cell-with-electronically-inactive-top-cell are shown, all in equilibrium, this is in dark and at $V = 0$.

It is clear that this strategy is not an exact treatment of the tandem cell, but it can be a good and very convenient approximation. The advantages are clear: the top and bottom cell can be studied in arbitrary light conditions ('from illumination' and 'from generation'; all settings of intensity and spectrum cut-off). Even more, one can study a sub-cell (top or bottom) with variation of optically important parameters (thickness d, band gap E_g, optical absorption $\alpha(\lambda)$ of a layer; contact filters T_{front} and $R_{back}(\lambda)$) without having to set-up the top and bottom cells each time again (by invoking `math split_tandem_cell…`). The drawback is that it is not so easy to assess the validity of the basic approximation ('electronically inactive counterparts'). One will have to check that the result (the calculated *IV* and efficiency parameters) of the tandem cell are not very sensitive to the assumptions in this script command (the precise value of the high N_D and low N_A values in the above example, the value of the contact S_n and S_p that were used, and especially the assumptions on the high recombination effectiveness of the 'inactive counter parts', especially the value of N_t).

It might make sense to do the vast amount of the exploring work with this strategy of *inactive counterparts*, and now and then (and also finally) check with the more rigorous, but less versatile strategies of *adapt generation* or *adapt filters*.
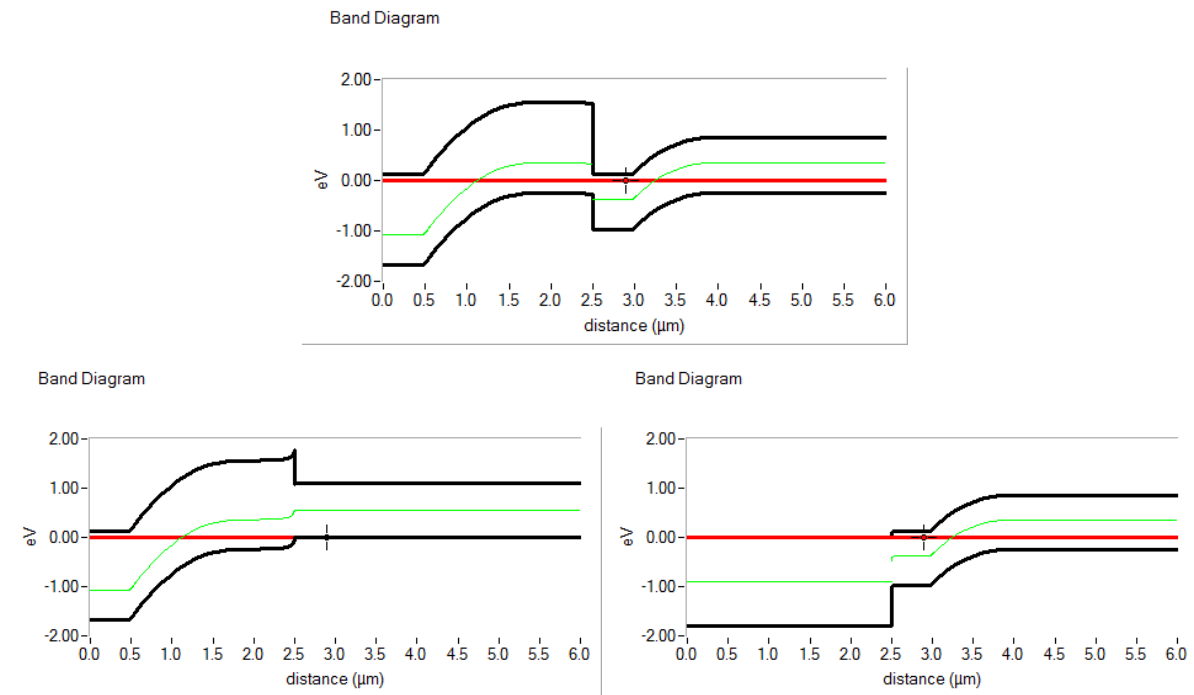
Figure 11.2   Top: Equilibrium energy band diagram of a tandem cell, illuminated from the left; the top cell (left part) is a wide band gap *np* junction, and the bottom cell is a narrow band gap *np* junction. Bottom Left: band diagram of the top-cell-with-electronically-inactive-bottom-cell; the bottom cell behaves electronically as one single p^+ layer, but remains optically active as it was in the tandem. Bottom Right: band diagram of the bottom-cell-with-electronically-inactive-top-cell; the top cell behaves electronically as one single n^+ layer, but remains optically active as it was in the tandem.

11.8 Connecting the top and bottom cells in series to a 2-terminal tandem cell

We illustrate the script for the tandem cell configuration `test tandem cell scripts illum from left.def`, and we work with the strategy 'adapt filters'. The script could start with:

load definitionfile test tandem cell scripts illum from left.def

// set the illumination conditions, and the parameters of the *IV* simulation

// (V_{start} and V_{stop}, # points, possibly 'stop after V_{oc}'…)

// split the tandem, set-up the top and bottom cell

math split_tandem_cell.adapt_contact_filters 2 3

action light // if not already done

// calculate *IV* of the top cell; store *V* in vvector and *I* in tvector (t of "top")

load definitionfile.temporary.top_cell

calculate singleshot

get iv vt

plot draw vt // *IV* of the top cell

// calculate *IV* of the bottom cell; store *V* in uvector and *I* in bvector (b of "bottom")

load definitionfile.temporary.top_cell

calculate singleshot

get iv ub

plot draw ub // *IV* of the bottom cell

// do the series (tandem) connection

// the first *IV* curve (top cell) is in (vvector, tvector); the second (bottom cell) in (uvector, bvector)

// the tandem *IV* result will be placed in wvector (the voltage) and ivector (the current density)

```
// the tandem IV curve will contain 100 points

math series WIVTUB 100

plot draw wi // the (constructed) IV curve of the tandem cell
```

To do the series connection, the common current density range of the top and bottom *IV* curves is determined. (In this example) 100 equidistant current density points are taken in this common *I*-range. For each such current density, the corresponding voltages V_{top} and V_{bottom} of the *IV* curves of top and bottom are determined by interpolation. The voltage $V_{\text{tandem}} = V_{\text{top}} + V_{\text{bottom}}$ is then attributed to this current density of the tandem cell. A few remarks are in order:

- when you want that the short circuit point of the tandem *IV* is included, thus $V_{\text{tandem}} = 0$, then in general one voltage of (V_{top}, V_{bottom}) should be positive, the other negative. It is thus advisable to set the V_{start} value of the *IV* calculation of top and bottom cells to a negative value, -0.5 V or so. Maybe some testing is required to find a suitable value for each specific problem.

- when calling e.g. `get iv vi`, the script gives suitable names to the script vectors, in this case "V (V)" for vvector and "J (mA/cm^2)" for ivector. But now you are simulating three sets of *IV* curves (top, bottom and tandem). Thus it is advisable to set suitable names explicitly by scripts commands as `set scriptvariable.iname J(top cell), mA/cm2`: this will be of great help to interpret the results (graphs, script variable tables, output tables, …)… and is definitely worth the effort of inserting many such commands.

- you can get the efficiency parameters (η, V_{oc}, J_{sc}, *FF*, V_{mpp}, J_{mpp}) of the last calculated *IV* curve with e.g.
  ```
  get characteristics.eta avector[0]

  get characteristics.voc cvector[7]

  get characteristics.jsc dvector[loopcounter]
  ```
  The result (here η, V_{oc} and J_{sc}) can be placed in any scalar scriptvariable (e.g. hvalue, xvalue,…) and in any element of a script vector, as done in the examples above. Refer to Table 10.7 for allowed formats.

- however, you cannot use these get characteristics commands to obtain the efficiency parameters of the tandem cell: the *IV* curve of a tandem cell is a constructed *IV* curve (here with `math series`), and not a directly simulated *IV* curve. Instead, you should use e.g.:
  ```
  math characteristics.eta eWI  // the η-value of the IV curve I(W) is calculated in the script
  ```
  scalar evalue
  ```
  math characteristics.voc oWI // the Voc-value of the IV curve I(W) is calculated in the script
  ```
  scalar ovalue

  To place the result in a script vector, instead of a script scalar, you can subsequently use a command like
  ```
  set scriptvariable.evector[loopcounter] evalue // evalue is placed in element
  ```
  loopcounter of evector

  (once there should come a single command to replace the two above).

11.9 Accessing the internal optical properties of a cell

Some optical properties of the cell are a function of both position *x* and wavelength λ: the light flux $N_{\text{phot}}(\lambda, x)$, the *eh* pair generation rate $G(\lambda, x)$, the optical absorption constant $\alpha(\lambda, x)$. Others are a function of wavelength λ alone: the optical transmission $T(\lambda)$ or reflection $R(\lambda)$ of the contact filters, the incident light flux just outside and just inside the cell, given by (when light is e.g. incident from the left side)

$$N_{\text{phot,just\_outside}}(\lambda) = N_{\text{phot}0}(\lambda)$$
$$N_{\text{phot,just\_inside}}(\lambda) = N_{\text{phot}0}(\lambda) \cdot T_{\text{front}}(\lambda) = N_{\text{phot}}(\lambda, 0)$$

(40)

And the total generation $G_{tot}(x)$, that is the generation $G(\lambda,x)$ integrated over all wavelengths, is a function of x alone.

All λ-dependencies are evaluated internally in SCAPS at the wavelengths where the spectrum is defined (from a file, or from a model). This also applies to the optical absorption $\alpha(\lambda)$ of a layer, and to the contact filter properties $T(\lambda)$ or $R(\lambda)$ when these are taken from an input file: internally, SCAPS does not calculate at the λ-values of for example an $\alpha(\lambda)$ file, but first interpolates these α values at the λ-values of the spectrum. Thus, the internal λ values meant here are those of the spectrum, but not those of the $\alpha(\lambda)$ … input files. All x-dependencies are evaluated internally at the mesh points.

Since SCAPS 3.3.10, march 2021, there is script support to get all these properties in script vectors, for displaying, plotting, manipulation…

One can list these properties as a function of wavelength λ, either by using all internal wavelengths, or at all wavelengths listed in a script vector (in nm) (then interpolation is used). When these properties are also a function of x, one should specify a single x value, either by specifying the mesh index, or by specifying the x-value directly, in μm. Tip: the position or index of the edges of a layer can be accessed by get commands like `get layer3.leftindex aindex` or `get layer2.rightx bvalue`…

One can also list these properties as a function of position x, only by using all internal mesh values of x. When these properties are also a function of λ, one should specify a single λ value in nm.

Examples of such get commands are thus:

`get incidentlightflux.all_internalwavelengths.just_outside_cell lo` // the result is in lvector (λ) and ovector ($N_{phot, just\_outside}$)

`get lightflux.all_internalwavelengths.at_position la 2.50` // the internal λ are set in lvector, the flux N_{phot} in avector; this flux is at $x = 2.5$ μm

`get contactright.opticalfilter.all_userwavelengths lt` // the user should have filled lvector with the desired wavelengths λ; the filter value $T(\lambda)$ or $R(\lambda)$ is set in tvector (this depends on the filter mode, that could have been set (definition file, user interface or script) to *transmission* or *reflection*)

`get absorption.at_wavelength.all_mesh_positions xa lvalue` // the mesh x-values are set in xvector, the $\alpha(x)$ values in avector; these α are at the wavelength set in lvalue (in nm)

With these new get commands, the user has more insight and control over the optical conditions in a cell.

11.10 Application examples

11.10.1 Introduction

We present here a few application examples to illustrate the new SCAPS facilities described in the above sections. The problem we treat is defined in `test tandem cell scripts illum from left.def`. It consists of a (relatively) wide band *np* cell with $E_g = 1.8$ eV and cell thickness 2.5 μm, and a (relatively) narrow band *np* cell with $E_g = 1.1$eV and cell thickness 3.5 μm. Some cell properties are summarised in Table 11.1. All layers have the optical absorption model 'power law' with exponent $n = 1.0$, with $\alpha_1 = 10^5$/cm, $\beta_1 = 0$ and with the band gap value E_g of the layer, and no sub-bandgap absorption.

In most cases of multi-junction cells, and also with the .def file under study here, any non-equilibrium calculation (this is, $V \neq 0$, or illumination, or both) will lead to convergence failure. Nevertheless, the light conditions in the cell, $N_{phot}(x, \lambda)$ and $G(x, \lambda)$ have been calculated correctly, and can be used, before the non-convergence message pops up. In a script, you can direct these annoying error messages to an error log file (script `command set errorhandling.appendtofile` or `.overwritefile`). You can also avoid these convergence failures at all by first simplifying the electronic properties of the whole tandem cell as discussed in section 11.7 above (the optical results remain unaffected, but of course any electrical result is useless).

Table 11.1   Some parameters of `test tandem cell scripts illum from left.def`.

		top cell		bottom cell	
		emitter	base	emitter	base
E_g	eV	1.8	1.8	1.1	1.1
λ_g	nm	689	689	1127	1127
d	μm	0.5	2.0	0.5	3.0
N_D	cm^{-3}	10^{17}		10^{17}	
N_A	cm^{-3}		10^{15}		10^{15}
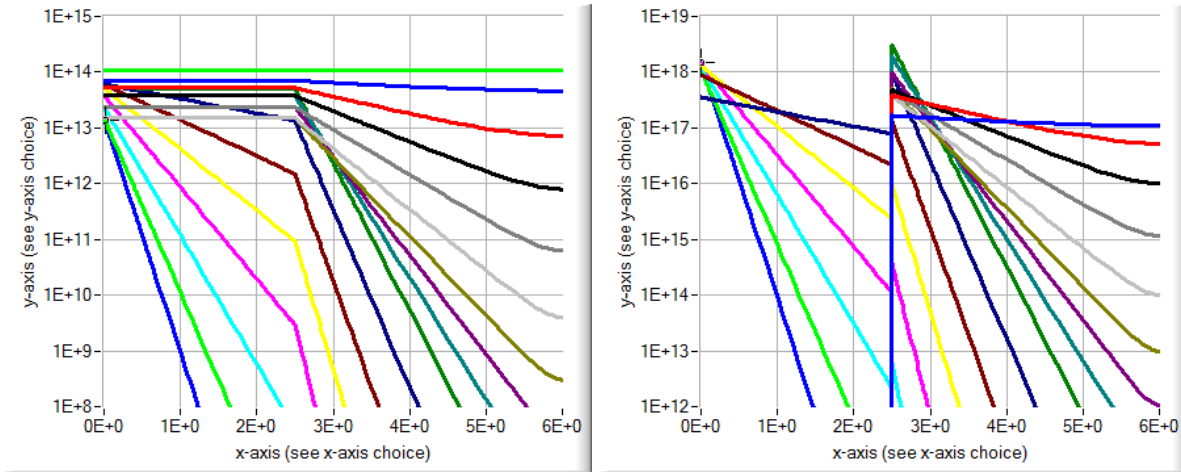
Figure 11.3   Left: light flux $N_{phot}(x; \lambda)$ (in #/cm^{2}s) and Right: generation $G(x, \lambda)$ (in #/cm^{3}s) as a function of x (in μm), and for 18 λ-values from 300 nm (blue curve bottom left) to 1150 nm (green curve on top in the N_{phot} graph left, but absent in the G graph right), step 50 nm. The tandem split (this is, between the top cell and the bottom cell) is at $x = 2.5$ μm. The light of the curves 300 nm ≤ λ ≤ 600 nm is absorbed almost completely in the top cell (0≤ x ≤2.5 μm) and quickly dies out in the bottom cell (2.5 μm≤ x ≤6.0 μm). The curves 650 nm ≤ λ ≤ 1100 nm shows that light passes without any absorption through the top cell, and is absorbed in the bottom cell, however less and less complete at longer wavelengths. The curve for 1150 nm shows that the light passes without absorption through the complete tandem cell, and does not cause any *eh* generation at all.

11.10.2 The illumination, absorption and generation conditions

Figure 11.3 illustrates the result of the script commands:

```
get lightflux.at_wavelength.all_mesh_positions xf lvector[loopcounter]
get generation.at_wavelength.all_mesh_positions xg lvector[loopcounter]
```

when `lvector` had been filled with 18 values 300, 350, …, 1150 before starting the loop.

And Figure 11.4 illustrates the result of the script commands:

```
get lightflux.all_internalwavelengths.at_position lf xvector[loopcounter]
get generation.all_internalwavelengths.at_position lg xvector[loopcounter]
```

when `xvector` had been filled with 13 values 0.0, 0.5, …, 6.0 before starting the loop.

These figures Figure 11.3 and Figure 11.4 are useful to visualise the illumination, light flux and *eh* generation rate behaviour in the tandem cell under study, as explained in the figure captions.
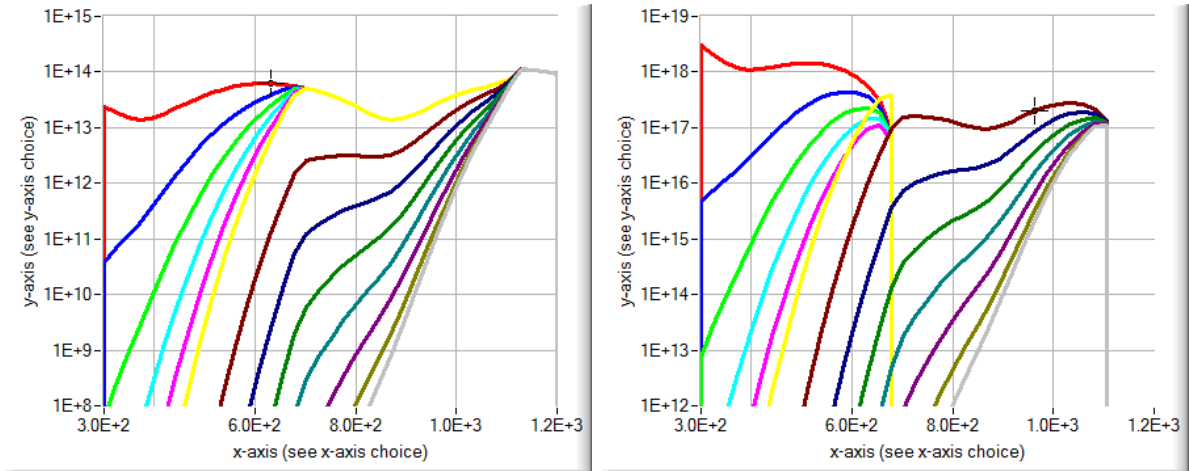
Figure 11.4 Left: light flux $N_{phot}(x; \lambda)$ (in #/cm²s) and Right: generation $G(x, \lambda)$ (in #/cm³s) as a function of λ (in nm), and for 13 x-values from 0 (red curve top left) to 6.0 μm (grey curve bottom right), step 0.5 μm. The tandem split at 2.5 μm is the yellow curve. The wavy shape of the top curves originates from the sinusoidal shape of the front transmission filter $R(\lambda)$ at the front contact (left); this has no realistic or physical relevance at all, we just used this $R(\lambda)$ to clearly see the influence of this filter… and we do see it ☺.

11.10.3 Results of the tandem simulation with 'adapt filters'

In the next illustration, we will vary the back reflection $R_b(\lambda)$ of the tandem cell from 0 to 100 %, 11 values with 10 % steps. The calculated filters ($R_{back}'(\lambda)$ that is placed at the back contact of the top cell, and $T_{front}'(\lambda)$ that is placed at the front contact of the bottom cell, are shown in Figure 11.5, right. The resulting total (= integrated over all λ) generation $G(x)$ is shown in Figure 11.5, left. The oversized figure caption explains some of the features to be observed.
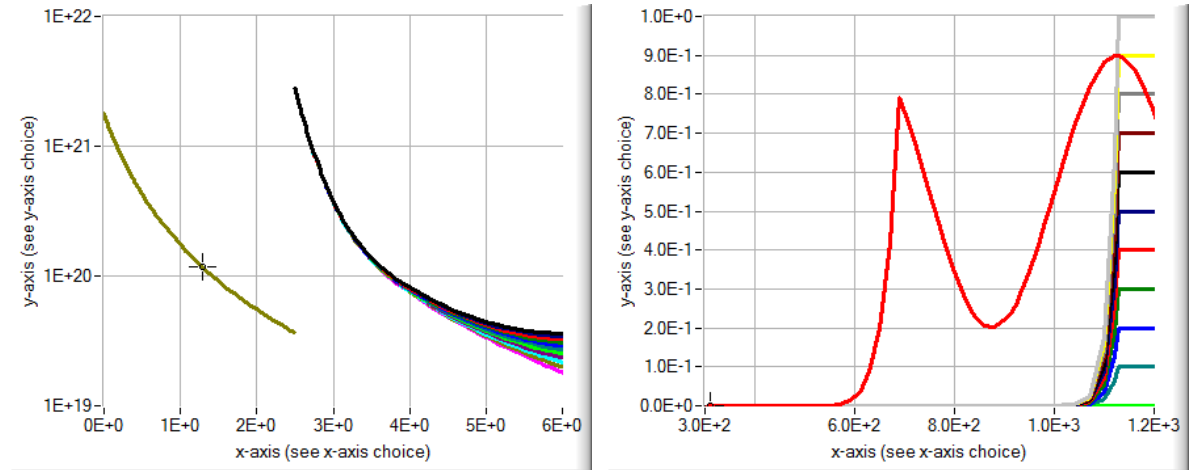

Figure 11.5 Problem file `test tandem cell scripts illum from left.def`. The back reflection R_{back} is varied from $R_{back} = 0$ to $R_{back} = 1.0$, step $\Delta R_{back} = 0.1$. Left: the *eh* generation rate $G(x)$ in #/cm³s integrated over all wavelengths of the incoming spectrum; the top cell has $0 \leq x \leq 2.5$ μm, and the bottom cell has 2.5 μm $\leq x \leq 6$ μm. Right: the adapted back reflection $R_{back}'(\lambda)$ of the top cell calculated by Eq. (38) and the adapted front transmission $T_{front}'(\lambda)$ of the bottom cell calculated by Eq. (39). The bundle $T_{front}(\lambda; R_{back})$ is seen as one curve (red); the sinusoidal shape of the front transmission filter used in this problem is followed for $\lambda > 690$ nm, but is efficiently damped for $\lambda < 690$ nm due to strong absorption in the top cell. The bundle $R_{back}(\lambda; R_{back})$ is seen as a bundle of 11 curves in the long wavelength range: $R_{back}'(\lambda) = R_{back}$ for light not absorbed at all ($\lambda > 1130$ nm), and the effect of R_{back} is quickly dying out for shorter wavelengths due to absorption of reflected light in the bottom cell. As a consequence, the generation $G(x)$ in the top cell does not depend on R_{back} at all, and $G(x)$ in the bottom cell has some weak dependence on R_{back} in the immediate vicinity of the back contact at $x = 6$ μm.

The *IV* and *QE* curves of the top and bottom cell, calculated with

```
math split_tandem_cell.adapt_contact_filters 2 3
```

and the *IV* curves of the tandem connection, calculated with (for example)

```
math series WIVTUB 100
```

are shown in Figure 11.6. The conclusions are consistent with Figure 11.5: the top cell is not sensible to R_{back}, the bottom cell only slightly.
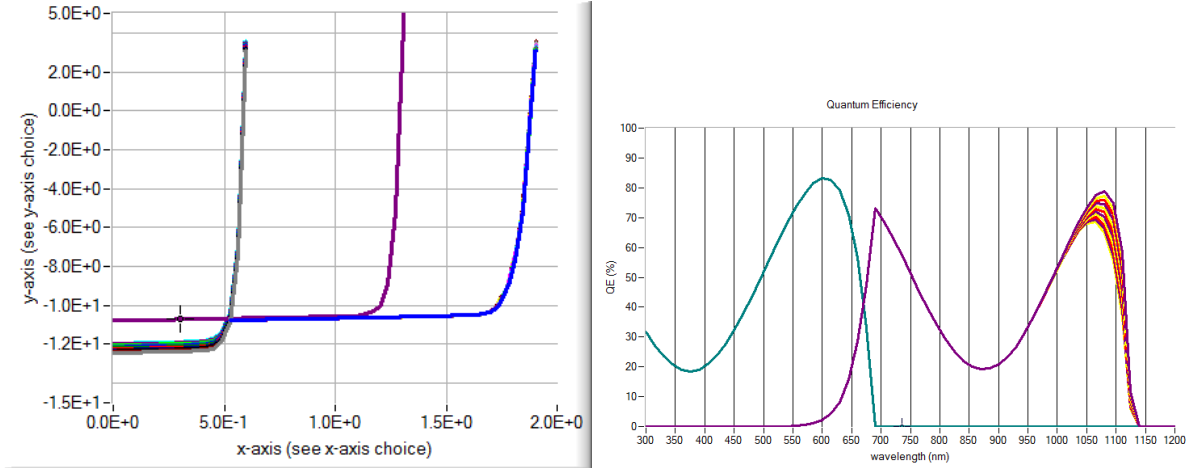

Figure 11.6   Left: the *IV* curves of the top cell (curves for all 11 R_{back} values almost coinciding, with $V_{oc} \sim 1.3$ V), the top cell (a bundle of slightly differing curves with $V_{oc} \sim 0.6$ V) and the series connection (curves coinciding with V_{oc} ~1.8 V). Right: the *QE* curves of the top cell (all curves coinciding; only > 0 for λ < 690 nm) and of the bottom cell (only > 0 for 690 nm < λ <1130 nm; the R_{back} bundle is only noticeable at long wavelengths, say λ > 1030 nm or so)

The conclusion would be different if the optical absorption of the layers would be much lower (lower absorption constant α or thickness *d*). We changed constant α_1 of the optical absorption model of all layers from $\alpha_1 = 10^5$ /cm to $\alpha_1 = 10^3$ /cm, and redid the calculations that lead to Figure 11.5 and Figure 11.6: the result is in Figure 11.7 and Figure 11.8.
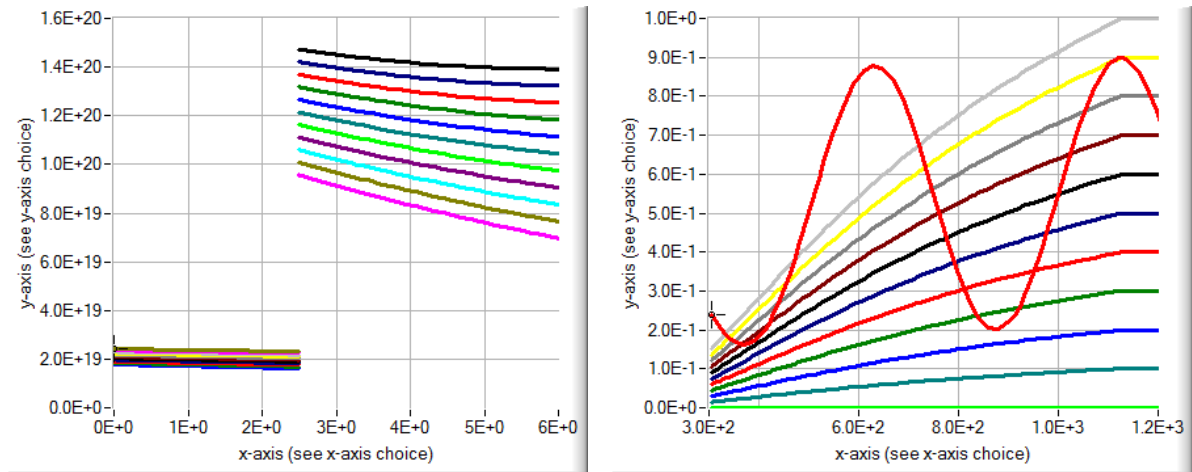

Figure 11.7   As in Figure 11.5, but for very low optical absorption (in all layers, α_1 was set to 10^3 /cm). The total generation $G_{tot}(x)$ now strongly depends on R_{back} in the bottom cell, and slightly but clearly visible in the top cell. The adapted $T_{front}'(\lambda)$ does not depend on R_{back} (in agreement with Eq. (39)), but the adapted $R_{back}'(\lambda)$ does (in agreement with Eq. (38), and much stronger with this low value for α_1: the influence is visible at all wavelengths, also for λ < 690 nm, that is almost completely absorbed in the top cell when $\alpha_1 = 10^5$ /cm.
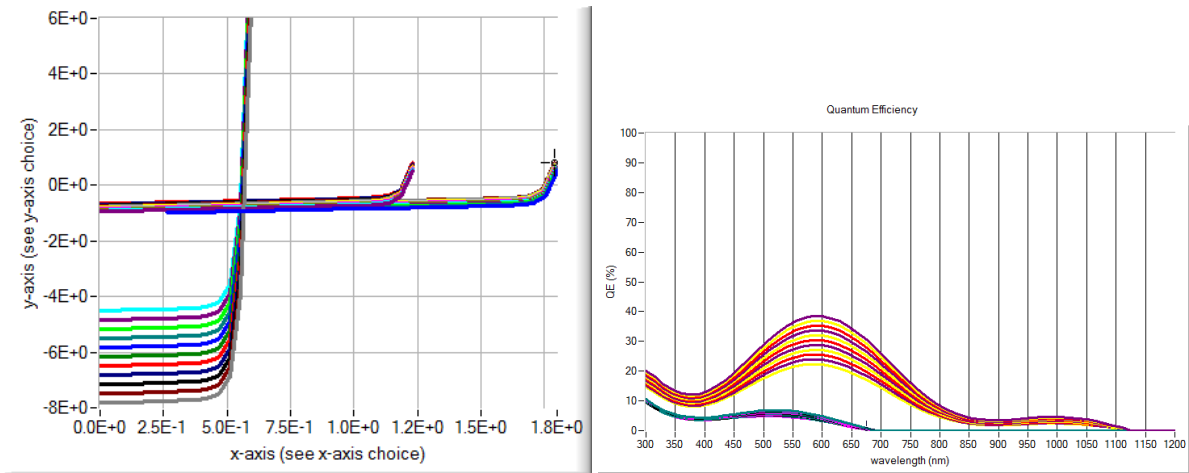
Figure 11.8 As in Figure 11.6, but for very low optical absorption (in all layers, α_1 was set to 10^3 /cm). Note that now the top cell and the resulting tandem cell have some noticeable dependence on R_{back}, and the bottom cell has a strong R_{back}-dependence.

11.10.4 Comparison of the 3 script strategies for tandem cells

With the strategy "adapt generation", the spectral response of course cannot be calculated. The results for $G(x)$ and *IV* are almost coinciding with the results obtained with strategy "adapt filters" (no illustration in a figure here).

The third strategy "electronically inactive parts…" does allow for the calculation of *QE*. The results for $G(x)$ and $QE(\lambda)$ are almost coinciding with the strategy "adapt filters" (no illustration). The *IV* results are similar, but here some deviation is visible. The *IV* curves of the top, bottom and tandem cells, calculated with the three strategies, are shown in Figure 11.9.
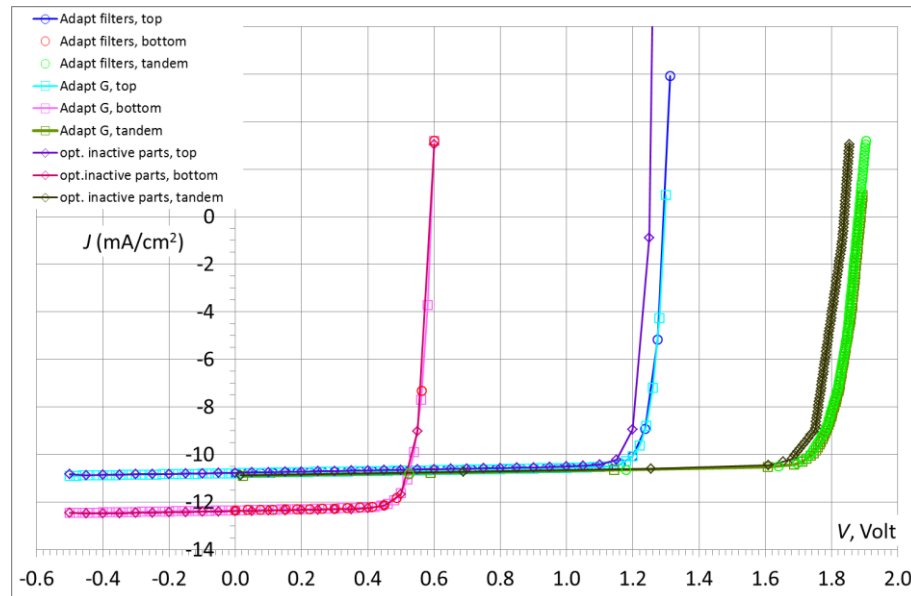

Figure 11.9 The *IV* curves of the top cell (blueish curves with $V_{oc} \sim 1.25$ V), the bottom cell (pinkish curves with $V_{oc} \sim 0.6$ V) and the tandem cell (greenish curves with $V_{oc} \sim 1.8$ V), for one value $R_{back} = 1$, and for the three script strategies. The *IV* curves for "adapt filters" and "adapt generation" almost coincide, but the *IV* curves of the top cell (and hence also the tandem cell) with strategy "electronically inactive parts" deviate somewhat (and have a V_{oc} about 0.04 V lower than calculated with the two other strategies).

To explore why the strategy "electronically inactive parts…" is not performing too well, we check the assumptions and algorithm that we used to make the bottom cell electronically inactive when studying the top cell, and the other way around. These assumptions were discussed in Section 11.7 page 139. We limit us

here to the "top cell with electronically inactive bottom cell", and to the doping and defect settings in the electronically active bottom cell (that we once called *fake bottom cell* in an Application Note – not very scientific, but shorter to type, and maybe clearer). The top cell in our example is *np* type (*n*-layer left, *p*-layer right), thus the *fake* bottom cell was set to *p*-type everywhere, with $N_A = 10^{19}$ cm^{-3}, and was given one mid-gap defect with density $N_t = 10^{17}$ cm^{-3}. These settings are internal in SCAPS when running the script command

```
math split_tandem_cell. with_electronically_inactive_parts n1 n2
```

and cannot be varied by the user. One can however load the definition file for the top cell with *fake* bottom cell set-up by the script (with `load definitionfile.temporary.top_cell`) and then, outside of the script, vary N_A or N_t of all *fake* bottom cell layers simultaneously in the batch settings. We did so once for our tandem cell example `test tandem cell scripts illum from left.def` as is, and then again for this same tandem cell with a modification: the top cell was given a Back Surface Field layer (BSF): the base layer (*p*-layer) of the top cell has $N_A = 10^{15}$ cm^{-3} over 1.5 µm, but $N_A = 10^{17}$ cm^{-3} over a 0.5 µm distance from the bottom cell to the right.

That leads to the results of Figure 11.10. The efficiency parameter most affected by the variation of N_A and N_t is V_{oc}; the influence on J_{sc} and *FF* is very minor.
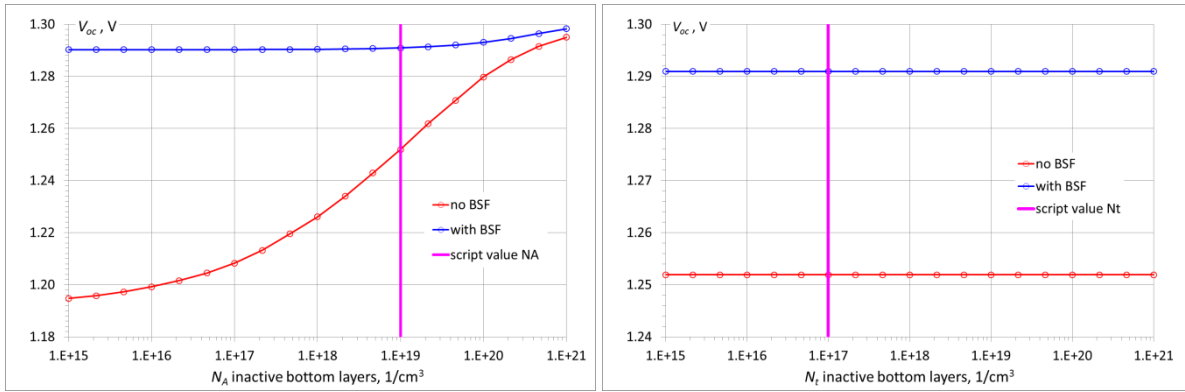

Figure 11.10 Dependence of the open circuit voltage V_{oc} of the top cell of a tandem cell with an "electronically inactive bottom cell", as constructed by the script command `math split_tandem_cell. with_electronically_inactive_parts…` Left: dependence on the acceptor density N_A set in the two layers of the electronically inactive bottom cell. Right: dependence on the defect density N_t set in the two layers of the electronically inactive bottom cell. The values of $N_A = 10^{19}$ cm^{-3} and $N_t = 10^{17}$ cm^{-3} that are set in the script are indicated by the magenta vertical line. Red curves: starting from `test tandem cell scripts illum from left.def` as is, thus without back surface field layer (no BSF). Blue curves: the top cell of this tandem structure was given a BSF layer (see text).

It is clear that "a mild variation of N_t around the set value $N_t = 10^{17}$ cm^{-3}" does not influence the results at all (Figure 11.10 right). "At all" is somewhat exaggerated: when examining e.g. the detailed $V_{oc}(N_t)$ dependence for the top cell with BSF, one sees some influence, but very, very minor, indeed negligible for all practical purposes (Figure 11.11). In contrast, a variation of N_A around its internal setting $N_A = 10^{19}$ cm^{-3} does influence the *IV* results, see Figure 11.10 left: this influence is small and maybe negligible for the cell with BSF (blue curve), but is substantial for the cell without BSF (red curve).

These examples show that:

- The strategy "partial cells with electronically inactive counterparts" is the most versatile of the three strategies: one can change the illumination conditions (spectrum, intensity) and the optically relevant parameters of the partial cell under study (absorption files or absorption model parameters, band gap, thickness; one can add layers…) without having to run each time again the `math split_tandem_cell` command in a script.

- However, the user should check the validity of the approximations used to obtain the "electronically inactive counterpart". These assumptions might be more or less valid, or totally not, depending on the

cell settings. SCAPS cannot do this analysis automatically for you, and vary the settings (in our illustration, the parameters N_A and N_t) to obtain a better implementation of the "electronically inactive" counterpart, the user is responsible for this. This illustrates again that a physical simulation program cannot substitute for physical insight ☺ ☹…

- And thus, we repeat our conclusion at the end of Section 11.7: "It might make sense to do the vast amount of the exploring work with the strategy of *inactive counterparts*, and now and then (and also finally) check with the more rigorous, but less versatile strategies of *adapt generation* or *adapt filters*".
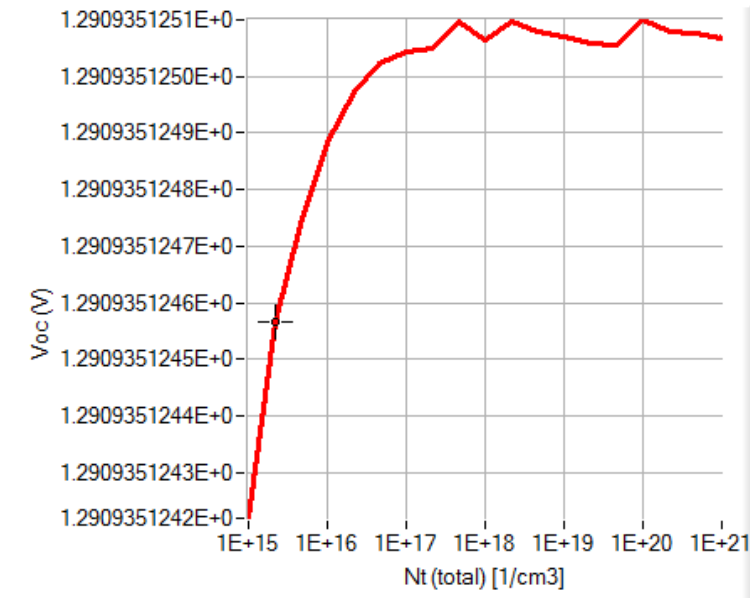

Figure 11.11 Detail of the blue curve of Figure 11.10 right: the $V_{oc}(N_t)$ dependence for the top cell with BSF. There is a very slight $V_{oc}(N_t)$ dependence, negligible in practice.

11.11 Conclusion

We can copy here the conclusion of our former Application Note on tandem cells:

- SCAPS cannot handle multi-junction structures directly, even not tandem structures.
- The way out is to calculate the top and bottom cells separately. This text gives hints and examples of how these 'top' and 'bottom' cells should be set-up.
- Three strategies, and script support for them, are developed in SCAPS 3.3.10.
- The SCAPS script also handles the series connection for you, within the SCAPS environment.
- … but it still looks very much as a Plan B… and actually it is one.

References

[1]   M. Burgelman, P. Nollet, S. Degrave, *Modelling polycrystalline semiconductor solar cells*, Thin Solid Films, 361 (2000) 527-532.

[2]   K. Decock, S. Khelifi, M. Burgelman, *Modelling multivalent defects in thin film solar cells*, Thin Solid Films, 519 (2011) 7481-7484.

[3]   M. Burgelman, J. Marlein, *Analysis of graded band gap solar cells with SCAPS*, Proceedings of the 23rd European Photovoltaic Solar Energy Conference, Valencia, 2008, pp. 2151-2155.

[4]   J. Verschraegen, M. Burgelman, *Numerical modeling of intra-band tunneling for heterojunction solar cells in SCAPS*, Thin Solid Films, 515 (2007) 6276-6279.

[5]   S. Degrave, M. Burgelman, P. Nollet, *Modelling of polycrystalline thin film solar cells : new features in SCAPS version 2.3*, Proceedings of the 3rd World Conference on Photovoltaic Energy Conversion, Osaka, 2003, pp. 487-490.

[6]   A. Niemegeers, M. Burgelman, *Numerical modelling of ac-characteristics of CdTe and CIS solar cells*, Proceedings of the 25th IEEE Photovoltaic Specialists Conference, Washington DC, 1996, pp. 901-904.

[7]   W.H. Press, S.A. Teukolsky, W.T. Vetterling, B.P. Flannery, *Numerical recipes in C*, 2nd ed., Cambridge University Press, New York, 1992.

[8]   J. Verschraegen, S. Khelifi, M. Burgelman, A. Belgachi, 21st European Photovoltaic Solar Energy Conference, Dresden, Germany, Sept. 2006, 2006.

[9]   S. Khelifi, M. Burgelman, J. Verschraegen, A. Belghachi, *Impurity photovoltaic effect in GaAs solar cell with two deep impurity levels*, Solar Energy Materials and Solar Cells, 92 (2008) 1559-1565.

[10]   S. Khelifi, J. Verschraegen, M. Burgelman, A. Belgachi, *Numerical simulation of the impurity photovoltaic effect in silicon solar cells*, Renewable Energy, 33 (2008) 293-298.

[11]   K. Decock, P. Zabierowski, M. Burgelman, *Modeling metastabilities in chalcopyrite-based thin film solar cells*, Journal of Applied Physics, 111 (2012) 043703.

[11']   M. Burgelman, K. Decock, S. Khelifi and A. Abass, *Advanced electrical simulation of thin film solar cells*, Thin Solid Films, 535 (2013) 296-301.

[12]   A. Niemegeers, S. Gillis, M. Burgelman, *A user program for realistic simulation of polycrystalline heterojunction solar cells: SCAPS-1D*, Proceedings of the 2nd World Conference on Photovoltaic Energy Conversion, Wien, 1998, pp. 672-675.

[13]   H.J. Pauwels, G. Vanhoutte, *Influence of interface states and energy barriers on efficiency of heterojunction solar-cells*, J. Phys. D-Appl. Phys., 11 (1978) 649-667.

[14]   J. Verschraegen, *Karakterisering en modellering met SCAPS van de CISCuT dunne-filmzonnecel*, dissertation Universiteit Gent. Faculteit Ingenieurswetenschappen, 2006.

[15]   S. Selberherr, *Analysis and Simulation of Semiconductor Devices*, Springer Verlag, Wien-New York, 1984.

[16]   J. Marlein, M. Burgelman, Proceedings of NUMOS (Int. Workshop on Numerical Modelling of Thin Film Solar Cells, Gent (B), 28-30 March 2007). p. 227-233 2007, 2007.

[17]   T. Walter, R. Herberholz, C. Müller, H.W. Schock, *Determination of defect distributions from admittance measurements and application to Cu(In,Ga)Se_2 based heterojunctions*, Journal of Applied Physics, 80 (1996) 4411-4420.

[18]   K. Decock, S. Khelifi, S. Buecheler, F. Pianezzi, A.N. Tiwari, M. Burgelman, *Defect distributions in thin film solar cells deduced from admittance measurements under different bias voltages*, Journal of Applied Physics, 110 (2011) 063722.