Simiconductor: manual

Jori Liesenborgs
jori.liesenborgs@uhasselt.be

December 17, 2014

Chapter 1

Introduction

1.1 Goal

`Simiconductor` is a program to find the equilibrium situation of the following set of drift-diffusion equations:

$$\vec{\nabla} \cdot \left(\varepsilon_{\text{rel}} \vec{\nabla} \phi \right) = -\frac{e}{\varepsilon_0} (p - n + b) \tag{1.1}$$

$$\vec{J_n} = -D_n \vec{\nabla} n + \mu_n n \vec{\nabla} \phi \tag{1.2}$$

$$\vec{J_p} = -D_p \vec{\nabla} p - \mu_p p \vec{\nabla} \phi \tag{1.3}$$

$$\frac{\partial n}{\partial t} = G - R - \vec{\nabla} \cdot \vec{J_n} \tag{1.4}$$

$$\frac{\partial p}{\partial t} = G - R - \vec{\nabla} \cdot \vec{J_p} \tag{1.5}$$

When equilibrium has been reached, $\frac{\partial n}{\partial t}$ and $\frac{\partial p}{\partial t}$ are equal to zero. In these equations $n(\vec{x})$ and $p(\vec{x})$ are electron and hole number densities at a specific location respectively, and is $\phi(\vec{x})$ the electrostatic potential. In principe, these are the only simulation properties that vary during a simulation, although it is possible that other quantities (like R for example) also depend on the values of n, p or ϕ. The vectors $\vec{J_n}$ and $\vec{J_p}$ represent the **number** currents, which are actually only intermediate quantities since their values are used in the final two equations.

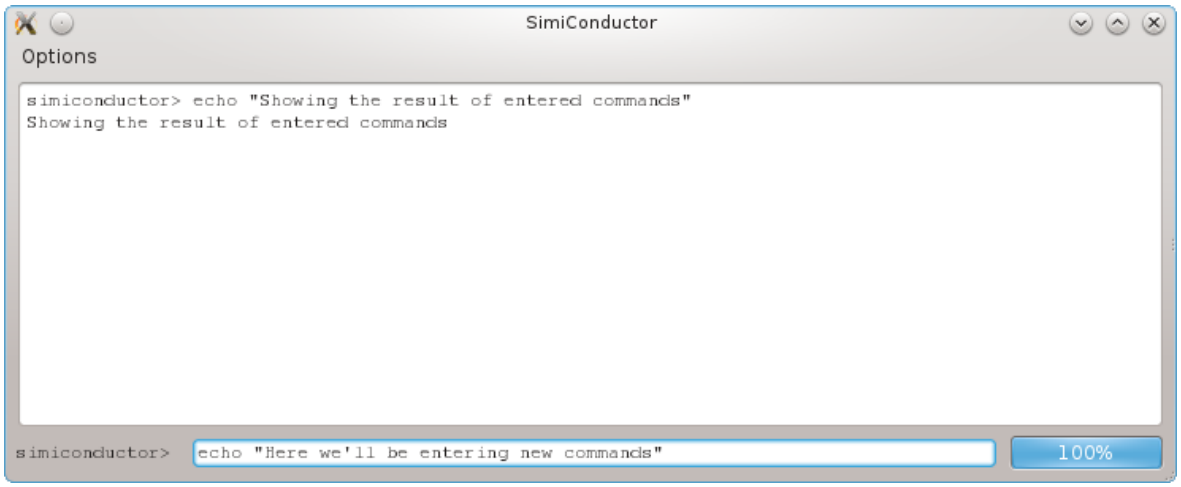
Other quantities which are used in these equations are:

- ε_0 and ε_{rel}: these are the vacuum permittivity ($8.85418781762 \times 10^{-12}\ F\ m^{-1}$) and the relative permittivity of the medium. The total permittivity equals $\varepsilon_0 \varepsilon_{\text{rel}}$.
- b: a fixed background charge density, expressed as a number density which in this case can be negative. This is necessary when simulating a p-n junction for example.
- D_n and D_p: diffusion constants of electrons and holes.
- μ_n and μ_p: mobilities of electrons and holes.
- G: rate with which electron-hole pairs are generated.
- R: recombination rate.

1.2 The user interface

In the `simiconductor` program, the equations above are solved numerically for the region specified, which can be either one or two dimensional. To look for the equilibrium solution, the equations are discretized by dividing the region in a number of pixels. To prepare a simulation, i.e. to specify the fixed quantities (like D_n) on each grid point and to specify initial conditions for the variable quantities (like n), one must supply a sequence of commands. These commands can be entered manually by the user in an interactive manner, or can be read from a file that has been prepared beforehand.

If the program is executed with a file name as an argument, e.g. `simiconductor mijnbestand.txt`, the program will read the file line by line and will try to interpret each line as a command. Without specifying a filename, the program is started in the interactive mode. Such a graphical user interface (GUI) can look as follows:



A very short file could look like this:

```
math/def nm 1e-9
sim1/new 64 100*nm
sim1/grid/set n 1 64 1e25
```

The first part of every line, like `math/def` or `sim1/new`, describes which command you'd like to execute. Following this command name, possibly a number of parameters for the command will be present. In the first command a variable with name `nm` is defined and assigned the value 10^{-9}. This way it is possible to use `nm` to represent one nanometer. The second command creates a new simulation grid, where `sim1` indicates that a one dimensional simulation is requested. The parameters of this command specify that 64 discrete points will be used which span a length of 100 nanometers. The last command initializes the electron number density `n` to $10^{25}m^{-3}$ for point 1 to point 64, i.e. for the entire simulation grid.

To get a list of all possible commands, you can simply enter the command `help`. To obtain more detailed information about a specific command, specify the command name as the parameter to the `help` command. For example, the command line `help sim1/phi/set` will produce the following output:

```
USAGE:
sim1/phi/set potentialdifference initgrid
```

```
DESCRIPTION:
Sets the potential difference to be used in the simulation. Note that this
includes the possible internal potential difference of the simulated device.

ARGUMENTS:
(1) potentialdifference (Real)
Difference in potential between right and left pixel in the simulation.

(2) initgrid (Bool) (Default: 'no')
If this flag is set, not only the potential at the edges will be set, but also
the potential on other grid points, using a linear interpolation.
```

The 'usage' section shows that this command in principle needs two arguments, which are detailed further in the 'arguments' section. As you might except, the 'description' section provides a few sentences which describe what the command does. In the text describing each argument you can see that the first argument, `potentialdifference`, requires an value of type `Real`. This simply means that at the corresponding location you'll need to specify a real number.

The second argument on the other hand is of type `Bool`, indicating that you can enter `true` or `false`, or even `yes` or `no`. Only these four possibilities are allowed for a `Bool` parameter. In the documentation of this `initgrid` parameter, you can also see that it has a default value. When a default value is present, it means that the use of this default value can be specified by entering an asterisk ('*') at that location. The command

```
sim1/phi/set 0.8 *
```

then means the same thing as entering

```
sim1/phi/set 0.8 no
```

In case after a certain parameter every remaining parameter has a default value, you don't need to specify '*' values for all of them to use these defaults. Instead, you can simply omit them entirely. In case of the previous example, the line

```
sim1/phi/set 0.8
```

would also be allowed and would have the same result.

The parameters for a command can have the following types:

- `Real`: A real number.
- `Integer`: An integer number. In case a real number is specified, the value will be rounded.
- `Bool`: A boolean value, which can be `yes`, `no`, `true` or `false`.
- `String`: This parameter should be a string, a sequence of characters. Because a space is used to separate different parameters, you cannot simply specify a sequence of characters which contain one or more spaces. If this is the case, if spaces are needed in the string, you can use quotes to indicate what belongs together as one parameter, for example:

```
sim1/save "My file.dat"
```

can be used to save the current simulation state to the file named `My file.dat`.

- `Choice`: In this case you can specify one particular value from a number of allowed values. The values that are allowed will depend on the command that was specified, but using the `help` command will provide you with a list of allowed values. For example, the command `help sim1/grid/set` will produce the following output:

```
USAGE:
sim1/reg/set property regionname value

DESCRIPTION:
Set the value of a property of the simulation in a specific named region.

ARGUMENTS:
(1) property (Choice)
Specifies the property to set.
Valid choices:
 - n    : electron number density
 - p    : hole number density
 - bg   : fixed background number density
 - g    : generation rate for electron-hole pairs
 - rf   : recombination rate will be rf*(n*p-ni*ni)
 - dn   : diffusion constant for electrons
 - dp   : diffusion constant for holes
 - nmob : electron mobility
 - pmob : hole mobility
 - eps  : relative permittivity
 - ni   : intrinsic electron number density

(2) regionname (String)
Name of the region in which this value should be set.

(3) value (Real)
Value to be set in the specified region.
```

  The first parameter is of type `Choice`, and as the documentation shows the allowed values are `n`, `p`, `bg`, `g`, `rf`, `dn`, `dp`, `nmob`, `pmob`, `eps` and `ni`.

When the parameter is of type `Real` or `Integer`, it is possible that not all numbers are allowed, but should be restricted to a certain range. If this is the case, the output of the corresponding `help` command will show that there's a minimum or maximum value, or both.

When `simiconductor` is reading commands from a file, the program will exit in case no further commands can be found in the file, or when one of the commands is unsuccessful. When the program was started in the interactive mode, you need to use the `quit` command to close the program. Interrupting a calculation is possible for some commands, and if this is desired you should choose the `Interrupt calculation` menu item in the `Options` menu. If this doesn't work and you really need to force the program to exit, you can choose `Force quit` in the same menu.

1.3 Auxiliary commands

The following commands are not directly related to a simulation itself, but may nonetheless be useful:

- `restart`: start fresh; the same as if you have exited `simiconductor` and started it again.
- `exit` or `quit`: close the program.
- `import`: read and execute command lines from the specified file.
- `export`: write all commands that have been executed so far to the specified file.
- `echo`: display the specified string.
- `chdir`: change the current working directory.
- `getdir`: show the current working directory, storing it in a variable if specified.

If the graphical user interface is started, you will see a 'File' menu in which you can choose 'Select file' and 'Select directory'. When such an option is used, the resulting path will be inserted in the box where you'd normally enter commands. This can be very handy to choose a file to import, or to change the working directory.

It's possible to assign names to values using the command `math/def`. Earlier, the following example of this was already shown:

```
math/def nm 1e-9
```

After assigning a name to a value this way, you can use this name at any location you'd normally specify parameter of type `Real`. You can also use this where an `Integer` is required, but remember that the value will be rounded.

Where `Integer` and `Real` types are expected, mathematical operations can be specified as well. Take care to avoid the use of spaces however, since these are used to separate different parameters. For example, if you'd specify

```
math/def nm 5e-10 + 5e-10
```

this would result in the following error:

```
Error parsing argument list:
Too many arguments

Usage:
math/def variable expression
```

To prevent this, you can either use quotes again to indicate what belongs together as a single item:

```
math/def nm "5e-10 + 5e-10"
```

or you can omit the spaces:

```
math/def nm 5e-10+5e-10
```

A list of variables that have been given a name in this way, can be displayed using the command `math/list`. To clear the list of already defined variable names, the command `math/clearvar` can be used. If you only want to know the result of a certain mathematical operation but you don't wish to assign a name to the result, the command `math/calc` will be useful.

Using the `math/def` command it is only possible to assign names to numbers. A more general construct is available using the `pre/def` command, which allows you to assign anything that you would otherwise type to a specific name. For example, you could do the following:

```
pre/def FileName my_file.dat
sim1/save $FileName
```

Using the first command, the string 'my_file.dat' is stored in the generalized variable with name 'FileName'. The second command then uses `$FileName` to refer to the contents of this variable. The dollar sign indicates that a generalized variable name is used instead of a mathematical variable. When a command line is read from the user interface or from a file, the line is checked for the presence of generalized variables and their contents are simply substituted. Only after this substitution has taken place, all the parameters will be evaluated. The command `pre/list` shows which variables have been defined in this way, as well as their values.

Chapter 2

A simple example

2.1 Preparing a simulation

A number of natural constants which are often useful have already been assigned a name. When you run the command `math/list`, you'll see (among others) the following line:

```
CONSTANTS:
  PI   = 3.14159
  e    = 1.60218e-19
  eps0 = 8.85419e-12
  h    = 6.62607e-34
  hbar = 1.05457e-34
  kB   = 1.38065e-23
  me   = 9.10938e-31
```

In this list you will recognize the values of π, the charge of an electron, the vacuum permittivity, the Planck constant, the reduced Planck constant, the Boltzmann constant and the mass of an electron. Unless specified otherwise, the program will always use SI units.

As a first step, we'll define a number of additional names to make the rest of the commands easier to read: the thickness of the device, the temperature, the relative permittivity, the charge carrier mobilities and the corresponding diffusion constants (Einstein relation). The internal voltage is defined as 0.8V, and corresponds to the difference in LUMO and HOMO levels. Apart from making the remaining command lines easier to understand, the names of these variables serve no special purpose.

```
math/def d 100e-9
math/def T 300
math/def epsrel 3.4
math/def kT_ev kB*T/e
math/def hMob 1e-7
math/def eMob 1e-7
math/def hDiff hMob*kT_ev
math/def eDiff eMob*kT_ev
math/def Vbi 0.8
```

Next, we'll define the names for the charge carrier densities at the contacts. The left contact is the ohmic for holes and will get a hole number density of $p_0 = 10^{19} m^{-3}$. The right contact

is ohmic for electrons and will receive the same density which we'll now call n_d. The values of the densities at the opposite sides is then determined by a Boltzmann factor which reflects the internal voltage.

```
math/def p0 1e19
math/def pd p0*exp(-Vbi/kT_ev)
math/def nd 1e19
math/def n0 nd*exp(-Vbi/kT_ev)
```

To simulate a dift-diffusion system, one first needs to define a grid on which the simulation needs to be carried out. Command names starting with `sim1` are used for a one dimensional simulation, command names starting with `sim2` are used for two dimensional simulations. Not all opties available to a 1D simulation are available for a 2D one, and vice versa. Here, we'll consider a straightforward 1D simulation.

First we define the layout of the simulation grid using the command:

```
sim1/new 200 d
```

This initializes the simulation grid in such a way that it consists of 200 points which span a distance `d` (defined earlier as 100 nanometers). If you'd then run `math/list` again, you'll see that a few constants have been assigned a different value: Dit zorgt ervoor dat een grid gedefinieerd wordt bestaande uit 200 punten die een afstand `d` voorstellen, eerder gedefinieerd als 100 nanometer. Als je daarna opniew `math/list`

```
CONSTANTS:
  NX1  = 200
  W1   = 1e-07
```

The constant `NX1` represents the number of points that's being used in the simulation; the constant `W1` represents the physical width that the simulated device has. When these specific names are used in additional commands, it can make it easier if you'd like to adapt the simulation to one which has a different number of points or a different width, since the same command lines will remain valid.

To run a 2D simulation similar constants will be defined: `NX2` en `NY2` will represent the number of pixels in x- and y-direction. `W2` en `H2` on the other hand will correspond to the physical width and height of a device.

Let's now initialize a number of properties of the simulation. This can be done using (among others) the command `sim1/grid/set`, for example to initialize the relative permittivity throughout the device.

```
sim1/grid/set eps 1 NX1 epsrel
```

This command will set the relative permittivity (property `eps` of the grid) to the value stored in variable `epsrel`, for every grid point from 1 to `NX1` (so for all grid points). Because it's not always practical to specify grid points in this way, it's also possible to define a spefic region and use the command `sim1/reg/set` to assign a value to all points in the region. A pre-defined region has the name `ALL` and represents the entire grid. If you run the command `reg1/list yes`, you'll see a list of all region names which are currently defined, together with a short overview of the grid points they represent. In this case, the output would be:

```
Currently defined regions:
  Name:        ALL
  Description:
        1. Line: 1 -> 200
```

This shows that the region with name `ALL` exists and consists of all points from 1 to 200. The commands which start with `reg1` refer to regions for a 1D simulation, while those that start with `reg2` refer to a 2D simulation. We'll use this approach to define the mobility values and diffusion constants for each grid point.

```
sim1/reg/set dn ALL eDiff
sim1/reg/set dp ALL hDiff
sim1/reg/set nmob ALL eMob
sim1/reg/set pmob ALL hMob
```

To set the values of the electron and hole densities at the contacts, we'll first define regions for this. Let's start with a region named `leftcontact`:

```
reg1/new leftcontact
```

to which we'll add the first pixel:

```
reg1/append/line leftcontact 1 1
```

The two coordinates in this line refer to the first and last pixel of the region, which in this case is the same since we only want one pixel in the region. Similarly, for the right side contact we execute the command:

```
reg1/new rightcontact
reg1/append/line rightcontact NX1 NX1
```

The values for electron and hole densities at the contacts can then be set in the following way:

```
sim1/reg/set n leftcontact n0
sim1/reg/set p leftcontact p0
sim1/reg/set n rightcontact nd
sim1/reg/set p rightcontact pd
```

In principle it's possible to define regions even before a simulation has been created using e.g. `sim1/new`. However, it is important to remember that the contents of e.g. `NX1` is set to a new value when a new simulation is created and when using a name like `NX1` to define a region, it is the *content* of that constant which is used.

For example, if we were to start a new simulation at this point which contained 300 grid points, a list of the existing regions (using `reg1/list yes`) will still show the `rightcontact` name:

```
Name:        rightcontact
  Description:
        1. Line: 200 -> 200
```

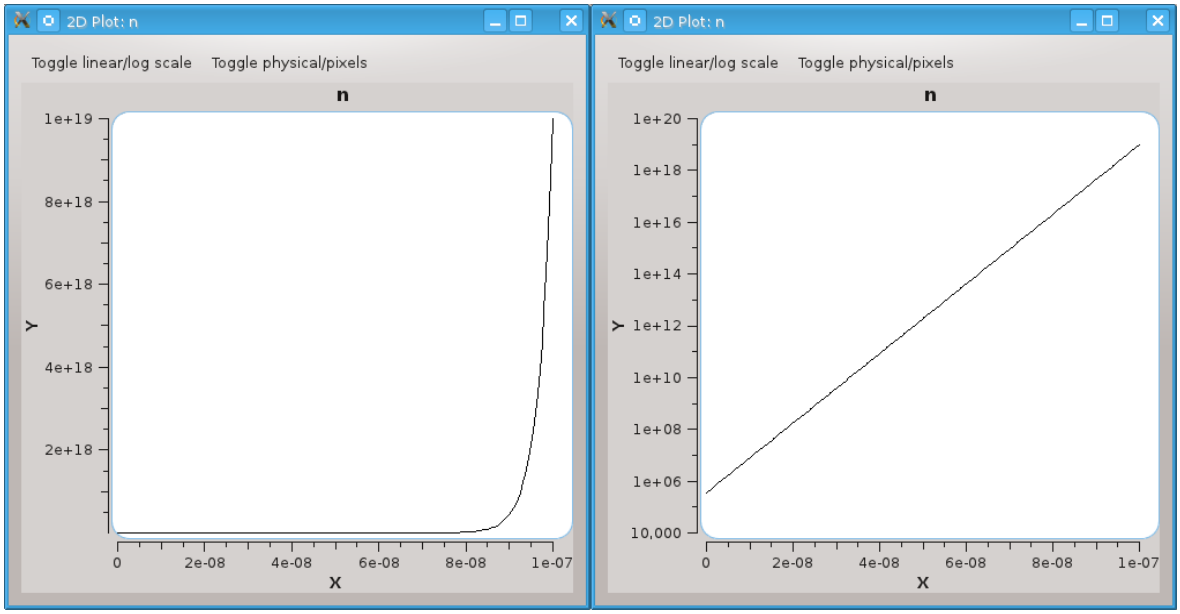
You'll notice that the region is still just point 200, since that was the value assigned to `NX1` at the time the region was defined. It did not automatically change to 300 by just starting the new simulation.

Next we'll initialize a few other properties:

```
sim1/temp/set T
math/def Vapp 1
sim1/phi/set Vbi-Vapp yes
sim1/grid/init n
sim1/grid/init p
```

The first command states that the simulation takes place at temperature `T`. The second command defines a variable that we'll use for the applied voltage and the third line initializes the potential difference between right and left contacts by combining the built-in voltage and the applied voltage `Vapp`. The `yes` in this line tells the program to initialize the electrostatic potential at each grid point by using linear interpolation. The final two lines initialize the electron and hole number densities at each grid point. By default, the values at the contacts are interpolated linearly on a *logarithmic* scale. If you supply `no` or `false` as an additional parameter, no logarithmic scale is used.

In interactive mode, the command `sim1/grid/wplot n` will show the left side figure below. When you click `Toggle linear/log scale`, the plot on the right side is shown instead.



2.2 Solving the equations

At this point we've prepated a straightforward 1D simulation in which there's no recombination or generation of charge carriers. The `simiconductor` program provides a number of ways to look for the equilibrium situation. To make sure that we can easily reuse the initial situation we've prepared, we'll save this situation to a file named `startsituation.dat`.

```
sim1/save startsituatie.dat
```

2.2.1 Time step based method

The drift diffusion equations specify how the charge carrier concentrations change at any particular time. Updating the values of the densities for a small time step over and over according to the equations, the concentrations will automatically evolve towards the equilibrium situation. An outline of the procedure:

1. Start with specific values for n and p at each point of the grid
2. Calculate the corresponding electrostatic potential ϕ at each grid point
3. Calculate the particle currents
4. Calculate $\frac{\partial n}{\partial t}$ and $\frac{\partial p}{\partial t}$
5. Using a small value for the time step Δt, add $\frac{\partial n}{\partial t}\Delta t$ to n and add $\frac{\partial p}{\partial t}\Delta t$ to p.
6. Repeat, starting from step 2

By repeating this procedure a large number of times, you'll notice that at a certain point the currents will have stabilized and won't change by a significant amount anymore. This means that equilibrium has been reached.

Using the command `sim1/run`, this time step based procedure can be used to evolve the simulation state towards equilibrium. The hard part is choosing an appropriate time step and the number of steps you want to execute. The command

```
sim1/run 250000 1e-11 10000
```

will advance the simulation using 250,000 timesteps of $10^{-11}s$ each. The last parameter instructs the program to display the values of the current (at the contacts, in the center, and averaged) every 10,000 steps. During the last steps, these values will not change anymore; after the command has completed, the values can be shown again using the command `sim1/xcur/show`. In this case, the output will be:

```
Left average:    0.641096 A/m^2
Center average:  0.641096 A/m^2
Right average:   0.641096 A/m^2
Overall average: 0.641096 A/m^2
```

2.2.2 Direct method

In several cases, a different and faster method to look for the equilibrium situation can be used. When $\frac{\partial n}{\partial t}$ and $\frac{\partial p}{\partial t}$ are both equal to zero, there are in principe three equations:

$$F^{\phi} = \vec{\nabla} \cdot \left(\varepsilon_{\text{rel}} \vec{\nabla}\phi\right) + \frac{e}{\varepsilon_0}(p - n + b) = 0 \tag{2.1}$$

$$F^{N} = G - R - \vec{\nabla} \cdot \vec{J}_n = 0 \tag{2.2}$$

$$F^{P} = G - R - \vec{\nabla} \cdot \vec{J}_p = 0 \tag{2.3}$$

The equations for $\vec{J}_n$ and $\vec{J}_p$ are still the same intermediate equations. When these equations are discretized, for each grid point one has three discretized versions of these equations. Since

we'll be looking for three values at each grid point (n, p and ϕ), we actually have an equal amount of equations and variables. Calling M the total number of grid points for which these equilibrium values of n, p and ϕ are sought, we have as many equations as variables which can we written as

$$F_j^{\phi}(\{\phi_i\},\{n_i\},\{p_i\}) = 0 \quad (2.4)$$

$$F_j^{N}(\{\phi_i\},\{n_i\},\{p_i\}) = 0 \quad (2.5)$$

$$F_j^{P}(\{\phi_i\},\{n_i\},\{p_i\}) = 0 \quad (2.6)$$

for each i from 1 to M. Looking for the multi-dimensional zero point of $3M$ equations in $3M$ variables can be done using the so-called Newton-Raphson method. The main disadvantage of this method is that is not guaranteed to evolve towards the correct solution.

Fortunately, if we use the example above everything works well. To load the initial simulation state we saved earlier and then search for the equilibrium using this direct method, the following commands can be used:

```
sim1/load startsituatie.dat yes
sim1/rundirect
```

In the `sim1/rundirect` command all the default parameter values are used since we did not specify any parameters explicitly. The fifth and last parameter that this command accepts is related to the scale which should be used when looking for the solution to the equations: if `no` is specified (the default), a linear scale is used to represent the charge carrier densities. If `yes` is specified, the logarithm of the densities is used (cfr quasi-Fermi level). Sometimes, one version works better than the other, but in this case both are able to find the equilibrium and running

```
sim1/load startsituatie.dat yes
sim1/rundirect * * * * yes
```

leads to a situation for which `sim1/xcur/show` again produces the output:

```
Left average:    0.641096 A/m^2
Center average:  0.641096 A/m^2
Right average:   0.641096 A/m^2
Overall average: 0.641096 A/m^2
```

In some cases, using a multi-resolution method can help to locate a solution. Using this method, first a solution is sought on a much lower scale, i.e. on a grid with fewer points. This solution will then be used as the starting situation for a grid with more pixels, and this process will continue until the desired resolution has been reached.

For the example above, we can execute the following commands to do this:

```
sim1/load startsituatie.dat yes
sim1/rundirectmres
```

2.3 J-V curve

Once an acceptable equilibrium situation has been found, this can be used as a starting point to look for a complete J-V curve. Apart from the built-in voltage, you'll also need to specify

starting and ending applied voltages. To get a good result, it is important that the starting situation is an equilibrium situation and that it corresponds to a voltage that lies somewhere in the range you're specifying this way. For the example we've been using, the following command could be used:

```
sim1/iv Vbi -0.2 1 200 iv.dat
```

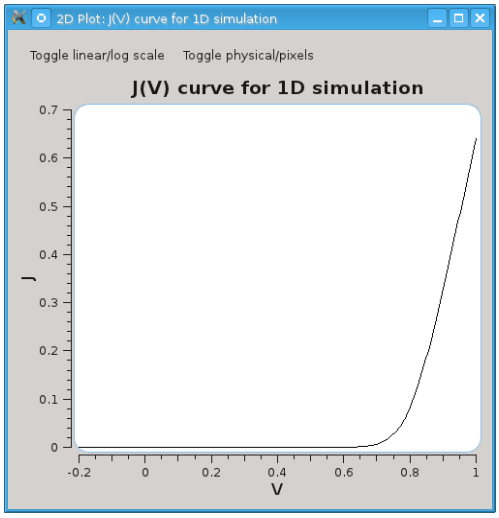
The first parameter here is the built-in voltage; the second and third parameters specify the range of applied voltages that should be considered. The total voltage between the contacts of the device will vary from $V_{bi} - V_{app,start}$ till $V_{bi} - V_{app,stop}$. After the command has been executed, the simulation state will correspond to the equilibrium situation for the this last voltage, in this case for an applied voltage of 1 V.

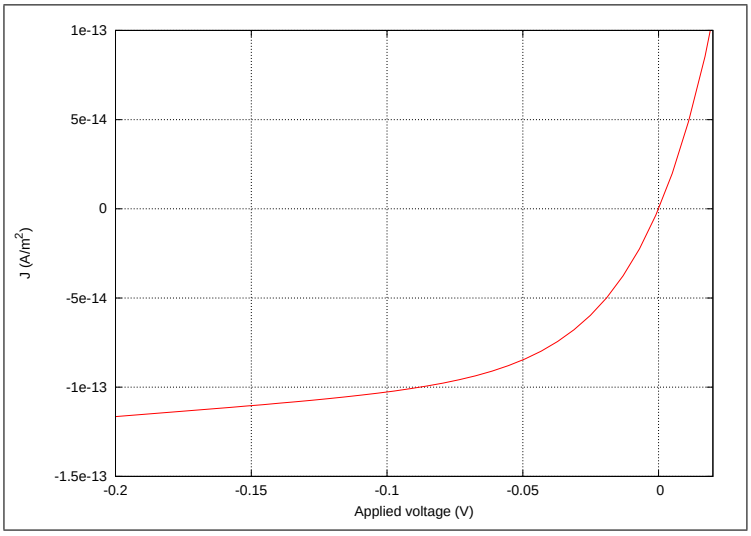
The fourth parameter tells the program that this voltage range should be evaluated at 200 different points. The final parameter specifies that the resulting J-V curve should be written to the file called 'iv.dat'. An important remark at this point is that to look for the J-V curve the program *always uses the Newton-Raphson* method.

In the interactive mode, you'll see the J-V curve as it's calculated, resulting in a final curve which is shown in the left figure below. When the command is finished, it will also show something like:

```
Estimated Voc = -7.74286e-06
Estimated Jsc = -5.53522e-17
```

From the calculated curve, the program will try to determine the points at which it intersects the x-axis (V_{oc}) and the y-axis (J_{sc}). In this case, very small numbers are retrieved, indicating that the curve goes through the origin to a very good approximation. When the 'iv.dat' file is opened and we zoom in on the region from -0.2 V to 0.02 V, the figure on the right will be shown, which visualizes the fact that the curve goes through the origin.





Chapter 3

Simulaties

3.1 Regions

To make it easier to set the values of properties for certain pixels of the grid, one can define regions, as was shown in an example earlier. The main idea is to be able to refer to a specific set of pixels using a name. To use this feature, you must first define the name of the region using the commands `reg1/new` and `reg2/new` for 1D and 2D simulations respectively, for example:

```
reg1/new myregion
```

Using the command `reg1/append/line` one can then associate pixels to this region. Furthermore, this command can be used multiple times allowing you to specify relatively complex regions:

```
reg1/append/line myregion 10 20
reg1/append/line myregion 50 60
```

If you then specify this region in the `sim1/reg/set` command, the value for the specified propertie will be set both for pixels 10 to 20 (included) and for pixels 50 to 60 (also included). For example, the command

```
sim1/reg/set eps myregion 3
```

makes sure that in the similation state that's defined at that time, the relative permittivity ('eps') is set to the value 3 for those pixels.

For 1D regions you can also use the command `reg1/delete` to remove a region and make its name available again, `reg1/clearall` to delete all the currently defined regions and `reg1/list` to obtain a listing of the existing region names. If a simulation has been created, a special region with name 'ALL' will also exist, referring to all pixels in the grid. A `yes` argument to the `reg1/list` command will not only show the names of the region, but also which pixels they contain.

There are similar commands for 2D regions, but there exist two commands to add pixels to a region: `reg2/append/rect` and `reg2/append/img`. The first command is simular to `reg1/append/line` and requires you to specify the coordinates of the region, for example:

```
reg2/new myregion
reg2/append/rect myregion 1 20 5 50
```

This defines a region with name 'myregion' and adds all pixels in the rectangle which has coordinates $(1, 20)$ as its bottom-left corner and $(5, 50)$ as its top-right corner. The command `reg2/append/img` allows you to specify an image to specify which pixels belong to a region. When this command is used, all pixels which are not entirely black are added to the region. Extra parameters to this command can be used to specify a scaling or translation to the coordinates which are obtained this way.

It must be emphasized that such regions can exist without a simulation state. This allows you to define regions before creating a simulation state and defined regions will still exist when a new simulation is started. This also means that care must be taken not to introduce errors when using variables like `NX1`: a region which is defined using such a variable, will be defined according to the contents of the variable *at that moment*. When a new simulation is created with another `NX1`, the region will *not* be adjusted automatically.

3.2 Basic commands

Creating a new simulation grid is done using the commands `sim1/new` and `sim2/new` for a 1D and 2D simulation respectively. As parameters one needs to specify the number of grid points in each direction, as well as the physical dimensions. When the 2D simulation is used, periodic boundary conditions are used for the x-direction. This means that if you want to use for example 64 points along the x-axis spanning $10\mu m$, point 1 will be at $0\mu m$ but point 64 will *not* be at $10\mu m$. It is the next, repeated point (the 65^{th} one, corresponding to the first point again) which lies at $10\mu m$. For 1D simulations, or for the y-direction of 2D simulations, no periodic boundary conditions are used, implying that if a similar example is used, point 64 will indeed lie at $10\mu m$.

Both commands accent a parameter with the name 'force'. In the interactive mode – when the input is not read from a file explicitly – the program tries to prevent that a new grid is created when an existing one contains changes that have not been saved. If this is the case, a message like the following one will be shown:

```
simiconductor> sim2/new 64 64 10e-6 10e-6
Error executing command: sim2/new
Simulation state not saved, specify force flag to override
```

If you're really sure that you don't need the current grid anymore, you can automatically delete this by specifying `yes` as the last parameter:

```
simiconductor> sim2/new 64 64 10e-6 10e-6 yes
```

In non-interactive mode, it is assumed that this 'force' parameter is always equal to `yes`, irrespective of the value you have entered yourself. Explicitly deleting a simulation grid can be done using the commands `sim1/clear` and `sim2/clear`. In interactive mode, a 'force' parameter can be specified here as well.

The current state of a simulation grid can be stored in a file using the commands `sim1/save` and `sim2/save`. Such a file can be loaded again later using `sim1/load` and `sim2/load` which again take a 'force' parameter.

If you already have created a simulation grid and want to keep the physical dimensions and the number of points, you can still *import* the saved result from another simulation in the current grid. In this operation it is not necessary that the number of grid points of the saved simulation are the same as the current one, the loaded property values will be interpolated if necessary. The relevant commands to import a simulation state are `sim1/import` and `sim2/import`.

For a 1D simulation, only a filename needs to be specified as an argument to this command and all properties of the saved file will be incorporated in the current simulation, interpolating them if necessary. The 2D version is somewhat more advanced[1], allowing you to specify 'varonly' and 'excludeborder' parameters, both defaulting to `no`. If 'varonly' is set to `yes`, only the values of n, p and ϕ will be imported from the saved file, i.e. the grid properties that can change while looking for the equilibrium situation. If 'excludeborder' is set to `yes`, then the borders of the current simulation state will not be adjusted (only in the y-direction, since there aren't really borders in the x-direction due to the periodic boundary conditions).

3.3 Initializing the grid

When a simulation grid has been created, for example using `sim2/new`, one still needs to set the values of various properties before the search for an equilibrium situation can take place. The most straightforward commands to do so, are `sim1/grid/set` and `sim2/grid/set`. Using these commands, you'll need to specify which property you want to adjust, the start and end coordinates, as well as the actual value that should be set. The command line

```
sim2/grid/set eps 2 2 10 15 4
```

will set the value for the relative permittivity ('eps') to 4 in the rectangular area from coordinate $(2, 2)$ to $(10, 15)$ (included). Which properties can be set this way can be seen by using the `help` command, in this case for example `help sim2/grid/set`.

There are a few small differences between the 1D and 2D versions. In the 1D case, it is possible to set a property called 'ni', the so-called intrinsic electron number density n_{int}. This value is actually only used for the recombination rule, in which the formula

$$R = r_f(n\,p - n_{\text{int}}^2)$$

is used (cf (1.4) and (1.5)). The prefactor r_f can be set on each individual grid point as well. In the 2D version, no n_{int} can be defined since a simpler recombination rule

$$R = r_f\,n\,p$$

is used. However, since in (1.4) and (1.5) the combination $G-R$ is always used, it is perfectly feasible to obtain the same effect. Support you'd like to use a specific value of n_{int}, then

$$G - R = G - r_f(n\,p - n_{\text{int}}^2) = G - r_f\,n\,p + r_f\,n_{\text{int}}^2 = G' - r_f\,n\,p$$

in which $G' = G + r_f\,n_{\text{int}}^2$. This means that by setting a different value of the generation rate, the same effect as in the 1D version can be obtained.

It is possible to set two additional parameters V_n and V_p. These are extra potential values which only have effect on the mentioned charge carrier, i.e. V_n will only influence the electrons,

[1] The only reason that this is not the case for the 1D simulation, is that it hasn't been necessary yet.

which V_p only influences the holes. To be specific, this means that the equations for the current densities become:

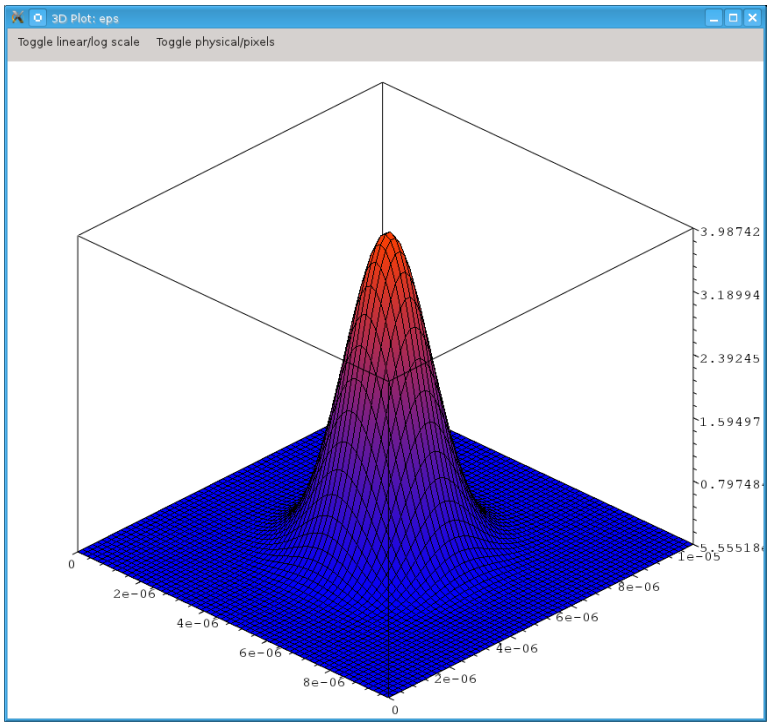
$$\begin{aligned} \vec{J}_n &= -D_n \vec{\nabla} n + \mu_n n \left(\vec{\nabla} \phi + \vec{\nabla} V_n \right) \\ \vec{J}_p &= -D_p \vec{\nabla} p - \mu_p p \left(\vec{\nabla} \phi + \vec{\nabla} V_p \right) \end{aligned}$$

These values, which won't change during the search for equilibrium, can be used to have more control over the number densities of electrons and holes in certain areas, as described in the *Supporting Information* of the article *Description of the morphology dependent charge transport and performance of polymer:fullerene bulk heterojunction solar cells.*

There also exist commands that can use region names. Apart from adding flexibility with respect to which pixels need to be initialized, this usually increases the readability of the command lines. The commands `sim1/reg/set` and `sim2/reg/set` are similar to the commands above, but instead of specifying coordinates you need to specify a region name. Additional flexibility can be obtained using the commands `sim1/reg/formula` and `sim2/reg/formula`: instead of specifying a single value for all pixels in the area, this allows you to set values according to a formula. For the 2D grid for example, the following command lines make sure the relative permittivity is assigned a value according to a gaussian pattern:

```
simiconductor> math/def CX 5e-6
simiconductor> math/def CY 5e-6
simiconductor> math/def WIDTH 1e-6
simiconductor> sim2/reg/formula eps ALL 4*exp(-((X-CX)^2+(Y-CY)^2)/(2*WIDTH^2))
```

A plot of the relative permittivity then looks like this:



In such a formula, there a number of special symbols you can use. The variables `PX` and `PY` refer to the grid coordinates of a certain grid point, for example from 1 to 64. The variables

`X` and `Y`, shown in the example above, refer to the physical coordinates, e.g. between 0 and $10\mu m$.

When a 2D simulation is used, the extra command `sim2/reg/bin` can be used as well. This allows you to set the values of a certain properties based on a BIN file or on an image file (PNG, JPG, ...). In case an image file is used, the color values stored in said image will always be converted to gray scale values, from 0 to 255. If a BIN file is used and it does not have the same number of points as the simulation grid, a warning will be shown and a rescaling version will be used. Finally, in this command you can also specify a formula, in which case the variables `PX`, `PY`, `X` and `Y` have the same meaning as before. Variables that you can use as well are `V` and `NV`: the first variable specifies the value as read from the BIN file or image file, while the second represents a normalized version. In this last case, the values of the file are rescaled in such a way that the smallest one has value 0, and the largest one has value 1.

Obtaining the value stored in a pixel can be done using the commands `sim1/grid/get` and `sim2/grid/get`. If desired, this value can also be stored in a variable. For example, the command

```
simiconductor> sim1/grid/get eps 1 EPS_LEFT
```

will retrieve the value of the relative permittivity for the leftmost pixel in the 1D simulation grid, and will store this value in the variable `EPS_LEFT`.

Using the commands `sim1/temp/set` and `sim2/temp/set` you can set the temperature for the simulation. Although the temperature is not present in the equations (1.1)-(1.5), it *is* used in certain methods to evolve the equations towards their equilibrium situation. Using `sim1/temp/show` and `sim2/temp/show` you can show the temperature which is currently set in a simulation.

With the commands `sim1/phi/set` and `sim2/phi/set`, the potential difference between the contacts can be specified. In the 1D case, this is the potential difference between right and left contact; in the 2D case this is the difference between top and bottom contacts. By default, this command will only fix the values of the electrostatic potential at the contacts, everywhere else in the grid the values will not be changed. This might be helpful if you already have the equilibrium situation for a certain potential difference, and want to start from a similar situation to look for the equilibrium for a slightly modified potential difference. In case the 'initgrid' parameter is set to `yes`, the values for the electrostatic potential at other grid points will indeed be modified, using a linear interpolation based on the values at the contacts. This can be useful when you don't have a good approximation of the potential yet, for example when starting a new simulation. The commands `sim1/phi/show` and `sim2/phi/show` will show the potential difference that has been set between the contacts.

Finally, other commands which are mainly useful when specifying the starting situation are `sim1/grid/init` and `sim2/grid/init`. Often you'll only know the values of n and p at the contacts, and using these commands you can interpolate these values to the rest of the grid. Using the 'logarithm' parameter (defaults to `yes`), you can specify if this interpolation should be done on a regular or logarithmic scale. In 2.1 an example of this was already shown.

3.4 Searching for the equilibrium situation

As explained in the introductory chapter 2, the methods to look for the equilibrium situation of the drift-diffusion equations can be subdivided in two categories: time step based methods and Newton-Raphson based methods. The time step based methods are executed using the commands `sim1/run` and `sim2/run`. Using those, you'll need to specify at least the number of steps to execute as well as the size of the time step. The procedure which is followed to evolve the equations, is described in 2.2.1. The advantage of this method is that, in principle, it will evolve towards the correct equilibrium situation, on the condition that the time step is not too large (otherwise the method becomes unstable). The main disadvantage is that this method can be very slow, especially when the number of grid points becomes larger (which is easily the case for 2D simulations). A number of parameters allow you to inspect intermediate results, or to write them to a file. This way plots can be updated while the simulation is running, the currents can be shown, plot data can be added to a file, etc.

One parameter that deserves a bit more explanation, is 'inversematrixsolver'. The default value is `yes`, indicating that when the electrostatic potential corresponding to certain boundary conditions and charge carrier distributions needs to be found, a matrix-based method will be used. When discretizing equation (1.1) and temporarily keeping the electron and hole densities fixed, the potential values can be found using a matrix inversion. In principle, this is the most accurate method, which helps to guide the equations towards equilibrium faster. However, it is possible that this method fails, or is simply too slow. In that case, an alternative method, a so-called *relaxation method*, can be used instead. This method calculates the error for the current potential values, and updates them based upon this error value.[2]. By repeating this a number of times the potential itself will converge towards the correct values.

The basic commands to search for equilibrium using the Newton-Raphson method are `sim1/rundirect` and `sim2/rundirect`. The last parameter for these commands is called 'uselog', and when this is set to `yes`, the electron and hole densities will be represented on a logarithmic scale. For some cases the regular representation works best, for others the logarithmic; there is unfortunately no way to predict which to use. The standard parameters will cause estimates to be made for the scale on which the densities and diffusion values should be represented. You can however also set relevant scale factors yourself, which can be helpful since these scale values certainly do have an effect on the algorithm.

In general, the Newton-Raphson based methods are more accurate than the time step base methods. Furthermore, if they work, they are usually also very fast. Unfortunately, the starting situation can have a large effect on the ability to find the equilibrium. In general, when the starting situation is close to the equilibrium situation, the method most of the times is able to find the correct final situation. It is for this reason that Newton-Raphson is always used for J-V curves. Also, even if you get a good estimate of the equilibrium situation using the time step based method, it can certainly be worthwhile to run a Newton-Raphson based search, starting from the situation you already found. Because the situation is a very good approximation to the equilibrium one, the Newton-Raphson method has a very good chance of working, and furthermore it will find the equilibrium situation with much more precision.

Finally, there are also the commands `sim1/rundirectmres` and `sim2/rundirectmres`. In case the previous Newton-Raphson commands were not able to find a good result, for example because the starting situation still differed much from the equilibrium one, it is certainly worth a try to use these methods. First, they will transform the specified starting situation to one

[2]Updating the values is typically done using a checkerboard pattern (in the 2D version), updating the black and red tiles in turn.

on a much smaller grid (e.g. a 5x5 one), and using this grid the Newton-Raphson method then searches for a low-resolution version of the equilibrium situation. The result of this is then interpolated onto a slightly larger grid and used as a starting situation for another Newton-Raphson procedure. This is then repeated until the initial grid resolution has been reached. By increasing the number of unknowns gradually, this procedure may find a stable equilibrium situation relatively fast. In these commands, you can specify the minimal grid resolution and the number of steps used to go from this low-resolution version to the desired resolution. Apart from these parameters, you can also specify a scale for the densities or for the diffusion values, as well as whether or not a logarithmic scale should be used for the charge densities.

3.5 Inspecting results

The values of the currents in different directions can be retrieved using the commands `sim1/xcur/show`, `sim2/xcur/show` and `sim2/ycur/show`. In case of a 2D simulation mostly the current in the y-direction will be interesting as this is the current between the contacts.

In interactive mode, the commands `sim1/grid/wplot` and `sim2/grid/wplot` can be used to show the values of a certain grid property on the screen. To get an overview of the properties which can be visualized, it's best to use the `help` command as described earlier. It's also possible to write the values of all properties (electron number density, hole number density, electrostatic potential, currents, ...) to a file. This is done using the commands `sim1/grid/fplot` and `sim2/grid/fplot`. Using those, in addition to a file name you'll also need to specify if the current values should be appended to a file (if it already exists), or overwritten. The data are written in text format, and can be visualized using some other program, e.g. using `gnuplot`. Which properties reside in which column can be found using e.g. `help sim1/grid/fplot`.

When using a 2D simulation, there are additional plot commands `sim2/line/fplot` and `sim2/avg/fplot`. The first command allows you to write the value of a property along a specified line, either in the x- or y-direction, to a file. For example, the command

```
sim2/line/fplot eps line.txt 10 x
```

will write the value of the relative permittivity along the line parallel to the x-axis and with coordinate $y = 10$ to the file called 'line.txt'. **Be careful:** if the file already exists, it will be overwritten! The command `sim2/avg/fplot` does something similar, but averaging the values perpendicular to the specified direction. The command

```
sim2/avg/fplot eps avg.txt x
```

will average the 'eps' values along the y-axis and write the resulting values, which still may vary along the x-axis, to the file 'avg.txt'. In this case the file will also be overwritten in case it already exists!

3.6 Calculating J-V curves

To calculate a J-V curve, you must always start from a situation which is already in equilibrium. Furthermore, this situation must correspond to a potential difference that corresponds

to a potential difference on the curve. The commands used to find a J-V curve are called `sim1/iv` and `sim2/iv`. When using these commands, you must supply values for V_{int}, V_{start} and V_{stop} as parameters. The externally applied voltage, V_{app}, will vary from V_{start} to V_{stop}; the total potential difference between the contacts for a particular value of V_{app} is $V_{\text{int}} - V_{\text{app}}$. It is also possible to specify in how many steps the interval between V_{start} and V_{stop} should be explored.

In the interactive mode, the J-V curve will be shown on the screen, but it is also possible to write the curve to a file. In that case, you'll also need to specify if the curve should be appended to the file, or if the file should be overwritten. Such a file is a simple text file with two columns, one for V_{app} and the other for the corresponding current density.

To obtain an equilibrium situation when going from one V_{app} to the next, the program always uses the Newton-Raphson method. For this reason, these commands also allow you to specify if you want to use a logarithmic scale to represent the densities, as well as scale values for the densities and diffusion values involved.

The `sim2/iv` command also has a parameter called 'doublesearch'. When this is set to `yes`, a certain procedure will internally be run twice. This can be helpful if you notice that the calculations for a certain J-V curve become unstable: perhaps this option can help to let everything converge.

3.7 Recombination models

If a 2D simulation is used, only a very simple recombination model is currently supported: $R = r_f\, n\, p$, where r_f is a parameter that can be defined at each point on the grid. When using a 1D simulation, two recombination models are possible: one is nearly the same as the 2D case, the other is the model used by Koster et al. in the article *Device model for the operation of polymer/fullerene bulk heterojunction solar cells*.

This alternative recombination model can be selected using the command `sim1/rec/ext`, where some parameters of the model will need to be specified. The command `sim1/rec/basic` allows you to switch back to the simpler recombination model. If you want to use the extened recombination model, it's best to use the Newton-Raphson method to obtain detailed results. The time step based method uses some simplifications at some points to prevent the simulation from becoming very slow.

If the Koster model is used, the command `sim1/P/show` is available as well. In this model, instead of

$$G - R$$

for the net generation rate, one uses

$$PG - (1 - P)R$$

where R equals

$$R = r_f(n\, p - n_{\text{int}}^2)$$

The factor P represents the average dissociation rate for electron-hole pairs. The command mentioned above then calculates the *average* dissociation rate for the current simulation (assuming it's in equilibrium).

Appendix A

Overview of the available commands

A.1 Index

A.2 Commands

A.2.1 version

```
USAGE:
version

DESCRIPTION:
```

A.2.2 restart

```
USAGE:
restart force

DESCRIPTION:
Clears simulation, regions and all newly defined symbols.

ARGUMENTS:
(1) force (Bool) (Default: 'no')
In interactive mode, enabling this flag forces a new grid even if the previous
one hasn't been saved. Ignored in non-interactive mode.
```

A.2.3 exit

```
USAGE:
exit force

DESCRIPTION:
Leaves the application

ARGUMENTS:
(1) force (Bool) (Default: 'no')
In interactive mode, force exit even if it hasn't been saved. In
non-interactive mode this flag is ignored.
```

A.2.4 quit

```
USAGE:
quit force

DESCRIPTION:
Leaves the application

ARGUMENTS:
(1) force (Bool) (Default: 'no')
In interactive mode, force exit even if it hasn't been saved. In
non-interactive mode this flag is ignored.
```

A.2.5 help

```
USAGE:
help commandname

DESCRIPTION:
Request help about a specific command or request a list of commands to be
displayed.

ARGUMENTS:
(1) commandname (String) (Default: '*')
Command name of which help should be retrieved. Use '*' for a list of commands.
```

A.2.6 exec

```
USAGE:
exec command

DESCRIPTION:
Executes a command.

ARGUMENTS:
(1) command (String)
Command string to be executed.
```

A.2.7 chdir

```
USAGE:
chdir directory

DESCRIPTION:
Change the current working directory.

ARGUMENTS:
(1) directory (String)
Directory which should become the current working directory.
```

A.2.8 getdir

```
USAGE:
getdir varname

DESCRIPTION:
Display the current working directory, optionally storing it in a preprocessor
variable.

ARGUMENTS:
(1) varname (String) (Default: '*')
If specified, the current directory will be stored in this preprocessor
```

variable.

A.2.9 echo

```
USAGE:
echo string

DESCRIPTION:
Displays a string.

ARGUMENTS:
(1) string (String) (Default: '')
String to be displayed.
```

A.2.10 import

```
USAGE:
import filename

DESCRIPTION:
Import command lines from a file.

ARGUMENTS:
(1) filename (String)
Name of the file from which commands should be imported.
```

A.2.11 export

```
USAGE:
export filename append

DESCRIPTION:
Export the commands which have been executed so far to a text file.

ARGUMENTS:
(1) filename (String)
Name of the file to which the commands should be exported.

(2) append (Bool)
If enabled and an existing file is specified, the data will be appended.
Otherwise, the file is overwritten.
```

A.2.12 pre/def

```
USAGE:
pre/def name value

DESCRIPTION:
```

```
Assigns a specific content to a preprocessor variable name. If the name is A,
$A will be replaced by the content of A.

ARGUMENTS:
(1) name (String) (not expanded)
Name of the preprocessor variable.

(2) value (String) (not expanded)
Content of the preprocessor variable.
```

A.2.13 pre/list

```
USAGE:
pre/list

DESCRIPTION:
List the names and content of the preprocessor variables currently defined.
```

A.2.14 sim1/new

```
USAGE:
sim1/new numx width force

DESCRIPTION:
Create a new 1D simulation state.

ARGUMENTS:
(1) numx (Integer) (MIN: 3) (Default: '128')
Number of pixels in X direction.

(2) width (Real) (MIN: 0)
Actual physical size in X direction.

(3) force (Bool) (Default: 'no')
In interactive mode, enabling this flag forces a new grid even if the previous
one hasn't been saved. Ignored in non-interactive mode.
```

A.2.15 sim1/clear

```
USAGE:
sim1/clear force

DESCRIPTION:
Clear the current simulation entirely.

ARGUMENTS:
(1) force (Bool) (Default: 'no')
In interactive mode, enabling this flag force clearing the simulation even if
the previous one hasn't been saved. Ignored in non-interactive mode.
```

A.2.16 sim1/save

```
USAGE:
sim1/save filename

DESCRIPTION:
Save current simulation settings.

ARGUMENTS:
(1) filename (String)
Name of the file to write the simulation to.
```

A.2.17 sim1/load

```
USAGE:
sim1/load filename force

DESCRIPTION:
Load simulation state.

ARGUMENTS:
(1) filename (String)
Name of the file to read the simulation state from.

(2) force (Bool) (Default: 'no')
In interactive mode, enabling this flag forces loading a grid even if the
previous one hasn't been saved. Ignored in non-interactive mode.
```

A.2.18 sim1/import

```
USAGE:
sim1/import filename

DESCRIPTION:
Import data from a specified simulation state save file into the current
simulation state. Grid sizes do not need to match, values will be interpolated
if needed.

ARGUMENTS:
(1) filename (String)
Name of the file from which the simulation state should be imported into the
current state.
```

A.2.19 sim1/reg/set

```
USAGE:
sim1/reg/set property regionname value

DESCRIPTION:
```

```
Set the value of a property of the simulation in a specific named region.

ARGUMENTS:
(1) property (Choice)
Specifies the property to set.
Valid choices:
 - n      : electron number density
 - p      : hole number density
 - bg     : fixed background number density
 - g      : generation rate for electron-hole pairs
 - rf     : recombination rate will be rf*(n*p-ni*ni)
 - dn     : diffusion constant for electrons
 - dp     : diffusion constant for holes
 - nmob   : base electron mobility
 - pmob   : base hole mobility
 - eps    : relative permittivity
 - ni     : intrinsic electron number density
 - gamman : sensitivity of electron mobility to electric field (Poole-Frenkel)
 - gammap : sensitivity of hole mobility to electric field (Poole-Frenkel)
 - Vn     : extra electron potential
 - Vp     : extra hole potential

(2) regionname (String)
Name of the region in which this value should be set.

(3) value (Real)
Value to be set in the specified region.
```

A.2.20 sim1/reg/formula

```
USAGE:
sim1/reg/formula property regionname formula

DESCRIPTION:
Set the value of a property of the simulation in a specific named region
according to a specific formula.

ARGUMENTS:
(1) property (Choice)
Specifies the property to set.
Valid choices:
 - n      : electron number density
 - p      : hole number density
 - bg     : fixed background number density
 - g      : generation rate for electron-hole pairs
 - rf     : recombination rate will be rf*(n*p-ni*ni)
 - dn     : diffusion constant for electrons
 - dp     : diffusion constant for holes
 - nmob   : base electron mobility
```

```
 - pmob   : base hole mobility
 - eps    : relative permittivity
 - ni     : intrinsic electron number density
 - gamman : sensitivity of electron mobility to electric field (Poole-Frenkel)
 - gammap : sensitivity of hole mobility to electric field (Poole-Frenkel)
 - Vn     : extra electron potential
 - Vp     : extra hole potential

(2) regionname (String)
Name of the region in which this value should be set.

(3) formula (String)
The formula to apply to the pixels in the region. You can use 'X' to specify
the physical coordinate of a pixel (which begins at 0), and 'PX' to specify a
pixel value (which begins at 1)
```

A.2.21 sim1/grid/set

```
USAGE:
sim1/grid/set property x1 x2 value

DESCRIPTION:
Set the value of a property of the simulation in a specific region.

ARGUMENTS:
(1) property (Choice)
Specifies the property to set.
Valid choices:
 - n      : electron number density
 - p      : hole number density
 - bg     : fixed background number density
 - g      : generation rate for electron-hole pairs
 - rf     : recombination rate will be rf*(n*p-ni*ni)
 - dn     : diffusion constant for electrons
 - dp     : diffusion constant for holes
 - nmob   : base electron mobility
 - pmob   : base hole mobility
 - eps    : relative permittivity
 - ni     : intrinsic electron number density
 - gamman : sensitivity of electron mobility to electric field (Poole-Frenkel)
 - gammap : sensitivity of hole mobility to electric field (Poole-Frenkel)
 - Vn     : extra electron potential
 - Vp     : extra hole potential

(2) x1 (Integer) (MIN: 1)
Left pixel coordinate of the region.

(3) x2 (Integer) (MIN: 1)
Right pixel coordinate of the region.
```

```
(4) value (Real)
Value to be set in the specified region.
```

A.2.22 sim1/grid/get

```
USAGE:
sim1/grid/get property x variable

DESCRIPTION:
Retrieves the value of a simulation property at a specific pixel coordinate,
and optionally stores the result in a variable.

ARGUMENTS:
(1) property (Choice)
Specifies the property to retrieve.
Valid choices:
 - n      : electron number density
 - p      : hole number density
 - bg     : fixed background number density
 - g      : generation rate for electron-hole pairs
 - rf     : recombination rate will be rf*(n*p-ni*ni)
 - dn     : diffusion constant for electrons
 - dp     : diffusion constant for holes
 - nmob   : base electron mobility
 - pmob   : base hole mobility
 - eps    : relative permittivity
 - ni     : intrinsic electron number density
 - gamman : sensitivity of electron mobility to electric field (Poole-Frenkel)
 - gammap : sensitivity of hole mobility to electric field (Poole-Frenkel)
 - Vn     : extra electron potential
 - Vp     : extra hole potential

(2) x (Integer) (MIN: 1)
X coordinate of the simulation pixel.

(3) variable (String) (Default: '*')
If specified, the resulting pixel value will be stored in this variable.
```

A.2.23 sim1/grid/init

```
USAGE:
sim1/grid/init property logarithm

DESCRIPTION:
Initialize one of the densities based on the values set at the boundaries. A
linear interpolation of either the densities themselves or their logarithms is
used.
```

```
ARGUMENTS:
(1) property (Choice)
Specifies the property to initialize based on the boundary conditions.
Valid choices:
 - n : electron number density
 - p : hole number density

(2) logarithm (Bool) (Default: 'yes')
If not set, the number densities themselves will be used, otherwise the
logarithm of the number densities will be used in the linear interpolation.
```

A.2.24 sim1/grid/wplot

```
USAGE:
sim1/grid/wplot property

DESCRIPTION:
Plots a property of the simulation.

ARGUMENTS:
(1) property (Choice)
Specifies the property to plot.
Valid choices:
 - n    : electron number density
 - p    : hole number density
 - bg   : fixed background number density
 - g    : generation rate for electron-hole pairs
 - rf   : recombination rate will be rf*(n*p-ni*ni)
 - dn   : diffusion constant for electrons
 - dp   : diffusion constant for holes
 - nmob : total electron mobility
 - pmob : total hole mobility
 - eps  : relative permittivity
 - r    : recombination rate
 - V    : potential
 - jx   : current in x-direction
 - jnx  : electron current in x-direction
 - jpx  : hole current in x-direction
 - ni   : intrinsic electron number density
 - Vn   : extra electron potential
 - Vp   : extra hole potential
```

A.2.25 sim1/grid/fplot

```
USAGE:
sim1/grid/fplot filename append

DESCRIPTION:
Write all data of the current simulation to a file which can be plotted by
```

```
gnuplot. The columns are: xpixel, n, p, V, jn, jp, j, R, dissprob, ni, bg, G,
rf, dn, dp, nmob, pmob, epsrel, Vn, Vp

ARGUMENTS:
(1) filename (String)
Name of the file to which the plot data should be written.

(2) append (Bool)
Flag indicating if the plot data should be appended to an existing file.
```

A.2.26 sim1/phi/set

```
USAGE:
sim1/phi/set potentialdifference initgrid

DESCRIPTION:
Sets the potential difference to be used in the simulation. Note that this
includes the possible internal potential difference of the simulated device.

ARGUMENTS:
(1) potentialdifference (Real)
Difference in potential between right and left pixel in the simulation.

(2) initgrid (Bool) (Default: 'no')
If this flag is set, not only the potential at the edges will be set, but also
the potential on other grid points, using a linear interpolation.
```

A.2.27 sim1/phi/show

```
USAGE:
sim1/phi/show

DESCRIPTION:
Shows the potential difference set in the current simulation (Vright-Vleft).
```

A.2.28 sim1/temp/set

```
USAGE:
sim1/temp/set T

DESCRIPTION:
Sets the temperature parameter of the simulation.

ARGUMENTS:
(1) T (Real) (MIN: 0, MAX: 10000)
Temperature for the simulation.
```

A.2.29 sim1/temp/show

```
USAGE:
sim1/temp/show

DESCRIPTION:
Shows the potential difference set in the current simulation (Vright-Vleft).
```

A.2.30 sim1/run

```
USAGE:
sim1/run steps dt xcursteps tmpsavefile tmpsavesteps plotfile plotfilesteps
plotwinsteps inversematrixsolver

DESCRIPTION:
Advance the simulation for a number of time steps, starting from the current
state.

ARGUMENTS:
(1) steps (Integer) (MIN: 1)
Number of time steps to advance the simulation with.

(2) dt (Real)
Size of the time step in numerrically solving the differential equation.

(3) xcursteps (Integer) (Default: '0')
The number of steps after which each time the x-current will be shown. A value
smaller than 1 disables the feature.

(4) tmpsavefile (String) (Default: '*')
Name of the file to which temporary results should be saved. The character '%'
will be expanded to the current simulation step number.

(5) tmpsavesteps (Integer) (Default: '0')
Each multiple of this number of steps, the current simulation state will be
saved to a file. A value smaller than 1 disables this feature.

(6) plotfile (String) (Default: '*')
Name of the file to which the simulation plot data should be written each time
a number of steps has passed. If the file exists, data will be appended. The
character '%' will be expanded to the current simulation step number.

(7) plotfilesteps (Integer) (Default: '0')
Each time this number of steps has passed, the current simulation plot data
will be written to the specified file. A value smaller than 1 disables the
feature.

(8) plotwinsteps (Integer) (Default: '0')
Each time this number of steps has passed, the current simulation data will be
plotted on screen. A value smaller than 1 disables the feature.
```

(9) inversematrixsolver (Bool) (Default: 'yes')
Use the inverse matrix based potential solver (should be somewhat more accurate which can lead to faster convergence to steady state).

A.2.31 sim1/rundirect

USAGE:
sim1/rundirect maxit showcurrents densscale diffscale uselog

DESCRIPTION:
Use a direct search for the steady state solution (uses a Newton-Raphson search).

ARGUMENTS:
(1) maxit (Integer) (MIN: 1) (Default: '1000')
Maximum number of iterations to perform.

(2) showcurrents (Bool) (Default: 'yes')
If enabled, after each iteration the current in the x-direction will be shown.

(3) densscale (Real) (Default: '-1')
Density scale to be used in the NR based simulation. A negative value causes the scale to be estimated automatically.

(4) diffscale (Real) (Default: '-1')
Diffusion scale to be used in the NR based simulation. A negative value causes the scale to be estimated automatically.

(5) uselog (Bool) (Default: 'no')
Use a logarithmic scale for the electron and hole densities (works better in some cases).

A.2.32 sim1/rundirectmres

USAGE:
sim1/rundirectmres maxit showcurrents minx steps errlimit densscale diffscale uselog

DESCRIPTION:
Instead of using a time step based simulation, use a direct search for the steady state solution. This version uses a multi-resolution approach: first a solution is sought to a low-resolution approximation, which is used as starting situation in a higher resolution version. This process is continued until the true resolution is reached.

ARGUMENTS:
(1) maxit (Integer) (MIN: 1) (Default: '100000')
Maximum number of iterations to perform.

(2) showcurrents (Bool) (Default: 'yes')
If enabled, after each iteration the current in the x-direction will be shown.

(3) minx (Integer) (MIN: 3) (Default: '8')
Minimal grid size in the x-direction

(4) steps (Integer) (MIN: 2) (Default: '5')
Number of multiresolution steps.

(5) errlimit (Real) (Default: '-1')
If the error after convergence exceeds this value, the multiresolution search will stop. A negative value disables this feature.

(6) densscale (Real) (Default: '-1')
Density scale to be used in the NR based simulation. A negative value causes the scale to be estimated automatically.

(7) diffscale (Real) (Default: '-1')
Diffusion scale to be used in the NR based simulation. A negative value causes the scale to be estimated automatically.

(8) uselog (Bool) (Default: 'no')
Use a logarithmic scale for the electron and hole densities (works better in some cases).

A.2.33 sim1/xcur/show

USAGE:
sim1/xcur/show

DESCRIPTION:
Calculate and display the net current in the simulation at certain positions.

A.2.34 sim1/iv

USAGE:
sim1/iv vint vstart vstop steps filename append maxit errlimit densscale diffscale uselog warningsaserrors

DESCRIPTION:
Calculate an IV curve using the direct solution method. The voltage between right and left contacts is set to 'vint-vapplied'. The Newton-Raphson based solver is used for this.

ARGUMENTS:
(1) vint (Real)
The built-in potential.

(2) vstart (Real)
Start value of the applied potential.

(3) vstop (Real)
End value of the applied potential.

(4) steps (Integer) (MIN: 2) (Default: '100')
The number of points in the IV curve.

(5) filename (String) (Default: '*')
Name of the file to which the plot data should be written.

(6) append (Bool) (Default: 'yes')
Flag indicating if the plot data should be appended to an existing file.

(7) maxit (Integer) (MIN: 1) (Default: '1000')
Maximum number of solution iterations to perform, for each potential difference.

(8) errlimit (Real) (Default: '-1')
If the error after convergence exceeds this value, the multiresolution search will stop. A negative value disables this feature.

(9) densscale (Real) (Default: '-1')
Density scale to be used in the NR based simulation. A negative value causes the scale to be estimated automatically.

(10) diffscale (Real) (Default: '-1')
Diffusion scale to be used in the NR based simulation. A negative value causes the scale to be estimated automatically.

(11) uselog (Bool) (Default: 'no')
Use a logarithmic scale for the electron and hole densities (works better in some cases).

(12) warningsaserrors (Bool) (Default: 'yes')
Treat warnings as errors

A.2.35 sim1/rec/ext

USAGE:
sim1/rec/ext pairdistance kf

DESCRIPTION:
Enable the extended recombination model and set its parameters.

ARGUMENTS:
(1) pairdistance (Real) (MIN: 0)
Electron-hole pair distance parameter.

```
(2) kf (Real) (MIN: 0)
Rate for a bound electron-hole pair to decay to the ground state.
```

A.2.36 sim1/rec/basic

```
USAGE:
sim1/rec/basic

DESCRIPTION:
Enable the default, basic recombination model.
```

A.2.37 sim1/rec/show

```
USAGE:
sim1/rec/show

DESCRIPTION:
Show the current recombination model and its parameters.
```

A.2.38 sim1/P/show

```
USAGE:
sim1/P/show

DESCRIPTION:
If the extended recombination model is used, display the current average
electron-hole pair dissociation probability.
```

A.2.39 sim2/new

```
USAGE:
sim2/new numx numy width height force

DESCRIPTION:
Create a new 2D simulation state.

ARGUMENTS:
(1) numx (Integer) (MIN: 3) (Default: '32')
Number of pixels in X direction.

(2) numy (Integer) (MIN: 3) (Default: '32')
Number of pixels in Y direction.

(3) width (Real) (MIN: 0)
Actual physical size in X direction.

(4) height (Real) (MIN: 0)
Actual physical size in Y direction.
```

(5) force (Bool) (Default: 'no')
In interactive mode, enabling this flag forces a new grid even if the previous one hasn't been saved. Ignored in non-interactive mode.

A.2.40 sim2/clear

USAGE:
sim2/clear force

DESCRIPTION:
Clear the current simulation entirely.

ARGUMENTS:
(1) force (Bool) (Default: 'no')
In interactive mode, enabling this flag force clearing the simulation even if the previous one hasn't been saved. Ignored in non-interactive mode.

A.2.41 sim2/save

USAGE:
sim2/save filename

DESCRIPTION:
Save current simulation settings.

ARGUMENTS:
(1) filename (String)
Name of the file to write the simulation to.

A.2.42 sim2/load

USAGE:
sim2/load filename force

DESCRIPTION:
Load 2D simulation state.

ARGUMENTS:
(1) filename (String)
Name of the file to read the simulation from.

(2) force (Bool) (Default: 'no')
In interactive mode, enabling this flag forces loading a grid even if the previous one hasn't been saved. Ignored in non-interactive mode.

A.2.43 sim2/import

```
USAGE:
sim2/import filename varonly excludeborder

DESCRIPTION:
Import data from a specified simulation save file into the current simulation.
Grid sizes do not need to match, values will be interpolated if needed.

ARGUMENTS:
(1) filename (String)
Name of the file from which simulation data should be imported into the current
simulation.

(2) varonly (Bool) (Default: 'false')
If set, only the properties that can change during the simulation will be
imported: electron density, hole density and potential.

(3) excludeborder (Bool) (Default: 'false')
If set, the top and bottom pixels in the simulation will not be replaced.
```

A.2.44 sim2/reg/set

```
USAGE:
sim2/reg/set property regionname value

DESCRIPTION:
Set the value of a property of the simulation in a specific named region.

ARGUMENTS:
(1) property (Choice)
Specifies the property to set.
Valid choices:
 - n    : electron number density
 - p    : hole number density
 - bg   : fixed background number density
 - g    : generation rate for electron-hole pairs
 - rf   : recombination rate will be rf*n*p
 - dn   : diffusion constant for electrons
 - dp   : diffusion constant for holes
 - nmob : electron mobility
 - pmob : hole mobility
 - eps  : relative permittivity
 - Vn   : extra electron potential
 - Vp   : extra hole potential

(2) regionname (String)
Name of the region in which this value should be set.

(3) value (Real)
```

```
Value to be set in the specified region.
```

A.2.45 sim2/reg/formula

```
USAGE:
sim2/reg/formula property regionname formula

DESCRIPTION:
Set the value of a property of the simulation in a specific named region
according to a specific formula.

ARGUMENTS:
(1) property (Choice)
Specifies the property to set.
Valid choices:
 - n    : electron number density
 - p    : hole number density
 - bg   : fixed background number density
 - g    : generation rate for electron-hole pairs
 - rf   : recombination rate will be rf*n*p
 - dn   : diffusion constant for electrons
 - dp   : diffusion constant for holes
 - nmob : electron mobility
 - pmob : hole mobility
 - eps  : relative permittivity
 - Vn   : extra electron potential
 - Vp   : extra hole potential

(2) regionname (String)
Name of the region in which this value should be set.

(3) formula (String)
The formula to apply to the pixels in the region. You can use 'X' to specify
the physical coordinate of a pixel (which begins at 0), and 'PX' to specify a
pixel value (which begins at 1)
```

A.2.46 sim2/reg/bin

```
USAGE:
sim2/reg/bin property regionname binfile formula

DESCRIPTION:
Set the value of a property of the simulation in a specific named region
according to a specific formula, using the values from a BIN file.

ARGUMENTS:
(1) property (Choice)
Specifies the property to set.
Valid choices:
```

```
 - n    : electron number density
 - p    : hole number density
 - bg   : fixed background number density
 - g    : generation rate for electron-hole pairs
 - rf   : recombination rate will be rf*n*p
 - dn   : diffusion constant for electrons
 - dp   : diffusion constant for holes
 - nmob : electron mobility
 - pmob : hole mobility
 - eps  : relative permittivity
 - Vn   : extra electron potential
 - Vp   : extra hole potential

(2) regionname (String)
Name of the region in which this value should be set.

(3) binfile (String)
File name of the BIN file (or image file to import from) containing the values
to be used.

(4) formula (String) (Default: 'V')
The formula to apply to the pixels in the region. You can use 'X' to specify
the physical coordinate of a pixel (which begins at 0), and 'PX' to specify a
pixel value (which begins at 1). The raw values in the BIN file can be referred
to by 'V', the values rescaled to an interval [0,1] can be specified using 'NV'.
```

A.2.47 sim2/grid/set

```
USAGE:
sim2/grid/set property x1 y1 x2 y2 value

DESCRIPTION:
Set the value of a property of the simulation state in a specific region.

ARGUMENTS:
(1) property (Choice)
Specifies the property to set.
Valid choices:
 - n    : electron number density
 - p    : hole number density
 - bg   : fixed background number density
 - g    : generation rate for electron-hole pairs
 - rf   : recombination rate will be rf*n*p
 - dn   : diffusion constant for electrons
 - dp   : diffusion constant for holes
 - nmob : electron mobility
 - pmob : hole mobility
 - eps  : relative permittivity
 - Vn   : extra electron potential
```

```
 - Vp   : extra hole potential

(2) x1 (Integer) (MIN: 1)
Left pixel coordinate of the region.

(3) y1 (Integer) (MIN: 1)
Bottom pixel coordinate of the region.

(4) x2 (Integer) (MIN: 1)
Right pixel coordinate of the region.

(5) y2 (Integer) (MIN: 1)
Top pixel coordinate of the region.

(6) value (Real)
Value to be set in the specified region.
```

A.2.48 sim2/grid/wplot

```
USAGE:
sim2/grid/wplot property

DESCRIPTION:
Plots a property of the simulation.

ARGUMENTS:
(1) property (Choice)
Specifies the property to plot.
Valid choices:
 - n    : electron number density
 - p    : hole number density
 - bg   : fixed background number density
 - g    : generation rate for electron-hole pairs
 - rf   : recombination rate will be rf*n*p
 - dn   : diffusion constant for electrons
 - dp   : diffusion constant for holes
 - nmob : electron mobility
 - pmob : hole mobility
 - eps  : relative permittivity
 - r    : recombination rate
 - V    : potential
 - jx   : current in x-direction
 - jy   : current in y-direction
 - jnx  : electron current in x-direction
 - jny  : electron current in y-direction
 - jpx  : hole current in x-direction
 - jpy  : hole current in y-direction
 - Vn   : extra electron potential
 - Vp   : extra hole potential
```

A.2.49 sim2/grid/fplot

```
USAGE:
sim2/grid/fplot filename append

DESCRIPTION:
Write all data of the current simulation to a file which can be plotted by
gnuplot. The columns are: xpixel, n, p, V, jnx, jpx, jx, jny, jpy, jy, R,
dissprob, ni, bg, G, rf, dn, dp, nmob, pmob, epsrel, Vn, Vp

ARGUMENTS:
(1) filename (String)
Name of the file to which the plot data should be written.

(2) append (Bool)
Flag indicating if the plot data should be appended to an existing file.
```

A.2.50 sim2/grid/get

```
USAGE:
sim2/grid/get property x y variable

DESCRIPTION:
Retrieves the value of a simulation property at a specific pixel coordinate,
and optionally stores the result in a variable.

ARGUMENTS:
(1) property (Choice)
Specifies the property to retrieve.
Valid choices:
 - n    : electron number density
 - p    : hole number density
 - bg   : fixed background number density
 - g    : generation rate for electron-hole pairs
 - rf   : recombination rate will be rf*n*p
 - dn   : diffusion constant for electrons
 - dp   : diffusion constant for holes
 - nmob : electron mobility
 - pmob : hole mobility
 - eps  : relative permittivity
 - Vn   : extra electron potential
 - Vp   : extra hole potential

(2) x (Integer) (MIN: 1)
X coordinate of the simulation pixel.

(3) y (Integer) (MIN: 1)
Y coordinate of the simulation pixel.

(4) variable (String) (Default: '*')
```

```
If specified, the resulting pixel value will be stored in this variable.
```

A.2.51 sim2/grid/init

```
USAGE:
sim2/grid/init property logarithm useVextra

DESCRIPTION:
Initialize one of the densities based on the values set at the boundaries. A
linear interpolation of either the densities themselves or their logarithms is
used.

ARGUMENTS:
(1) property (Choice)
Specifies the property to initialize based on the boundary conditions.
Valid choices:
 - n : electron number density
 - p : hole number density

(2) logarithm (Bool) (Default: 'yes')
If not set, the number densities themselves will be used, otherwise the
logarithm of the number densities will be used in the linear interpolation.

(3) useVextra (Bool) (Default: 'yes')
Use the Vextra potentials to account for concentration differences in N and P.
Only works if the logarithm is being used.
```

A.2.52 sim2/phi/set

```
USAGE:
sim2/phi/set potentialdifference initgrid

DESCRIPTION:
Sets the potential difference to be used in the simulation. Note that this
includes the possible internal potential difference of the simulated device.

ARGUMENTS:
(1) potentialdifference (Real)
Difference in potential between the top row and bottom row in the simulation.

(2) initgrid (Bool) (Default: 'no')
If this flag is set, not only the potential at the edges will be set, but also
the potential on other grid points, using a linear interpolation.
```

A.2.53 sim2/phi/show

```
USAGE:
sim2/phi/show
```

```
DESCRIPTION:
Shows the potential difference set in the current simulation (Vtop-Vbottom).
```

A.2.54 sim2/temp/set

```
USAGE:
sim2/temp/set T

DESCRIPTION:
Sets the temperature parameter of the simulation.

ARGUMENTS:
(1) T (Real) (MIN: 0, MAX: 10000)
Temperature for the simulation.
```

A.2.55 sim2/temp/show

```
USAGE:
sim2/temp/show

DESCRIPTION:
Shows the potential difference set in the current simulation (Vright-Vleft).
```

A.2.56 sim2/run

```
USAGE:
sim2/run steps dt ycursteps tmpsavefile tmpsavesteps plotfile plotfilesteps
plotwinsteps simtype inversematrixsolver

DESCRIPTION:
Advance the simulation for a number of time steps, starting from the current
state.

ARGUMENTS:
(1) steps (Integer) (MIN: 1)
Number of time steps to advance the simulation with.

(2) dt (Real)
Size of the time step in numerrically solving the differential equation.

(3) ycursteps (Integer) (Default: '0')
The number of steps after which each time the y-current will be shown. A value
smaller than 1 disables the feature.

(4) tmpsavefile (String) (Default: '*')
Name of the file to which temporary results should be saved. The character '%'
will be expanded to the current simulation step number.

(5) tmpsavesteps (Integer) (Default: '0')
```

```
Each multiple of this number of steps, the current simulation state will be
saved to a file. A value smaller than 1 disables this feature.

(6) plotfile (String) (Default: '*')
Name of the file to which the simulation plot data should be written each time
a number of steps has passed. If the file exists, data will be appended. The
character '%' will be expanded to the current simulation step number.

(7) plotfilesteps (Integer) (Default: '0')
Each time this number of steps has passed, the current simulation plot data
will be written to the specified file. A value smaller than 1 disables the
feature.

(8) plotwinsteps (Integer) (Default: '0')
Each time this number of steps has passed, the current simulation data will be
plotted on screen. A value smaller than 1 disables the feature.

(9) simtype (Choice) (Default: 'double')

Valid choices:
 - double    : Straightforward double based simulation.
 - doublerel : Used current state as base and calculates relative changes each
time.
 - gpu       : GPU based implementation. Should be faster for larger grids, but
since on the GPU a float representation is used, it may be less precise.

(10) inversematrixsolver (Bool) (Default: 'yes')
Use the inverse matrix solver to solver the Poisson equation (should be
slightly more accurate, causing a faster convergence to steady state)
```

A.2.57 sim2/rundirect

```
USAGE:
sim2/rundirect maxit showcurrents errlimit timeout densscale diffscale uselog
nponly

DESCRIPTION:
Instead of using a time step based simulation, use a direct search for the
steady state solution.

ARGUMENTS:
(1) maxit (Integer) (MIN: 1) (Default: '1000')
Maximum number of iterations to perform.

(2) showcurrents (Bool) (Default: 'yes')
If enabled, after each iteration the current in the y-direction will be shown.

(3) errlimit (Real) (Default: '-1')
If the error after convergence exceeds this value, an error is generated. A
```

negative value disables this feature.

(4) timeout (Real) (Default: '-1')
The maximum amount of time the search for the equilibrium situation can take. Set to a negative value to disable.

(5) densscale (Real) (Default: '-1')
Density scale, is only used in NR based simulation and potential based simulation. A negative value causes the scale to be estimated automatically.

(6) diffscale (Real) (Default: '-1')
Diffusion scale, is only used in NR based simulation. A negative value causes the scale to be estimated automatically.

(7) uselog (Bool) (Default: 'no')
Use a logarithmic scale for the electron and hole densities (works better in some cases).

(8) nponly (Bool) (Default: 'no')
Do not change the electrostatic potential V, only look for N and P solutions.

A.2.58 sim2/rundirectmres

USAGE:
sim2/rundirectmres maxit showcurrents minx miny steps errlimit timeout densscale diffscale uselog

DESCRIPTION:
Instead of using a time step based simulation, use a direct search for the steady state solution. This version uses a multi-resolution approach: first a solution is sought to a low-resolution approximation, which is used as starting situation in a higher resolution version. This process is continued until the true resolution is reached.

ARGUMENTS:
(1) maxit (Integer) (MIN: 1) (Default: '100000')
Maximum number of iterations to perform.

(2) showcurrents (Bool) (Default: 'yes')
If enabled, after each iteration the current in the y-direction will be shown.

(3) minx (Integer) (MIN: 2) (Default: '8')
Minimal grid size in the x-direction

(4) miny (Integer) (MIN: 3) (Default: '8')
Minimal grid size in the y-direction

(5) steps (Integer) (MIN: 2) (Default: '5')
Number of multiresolution steps.

(6) errlimit (Real) (Default: '-1')
If the error after convergence exceeds this value, the multiresolution search will stop. A negative value disables this feature.

(7) timeout (Real) (Default: '-1')
The maximum amount of time the search for the equilibrium situation can take. Set to a negative value to disable.

(8) densscale (Real) (Default: '-1')
Density scale to be used in the NR based simulation. A negative value causes the scale to be estimated automatically.

(9) diffscale (Real) (Default: '-1')
Diffusion scale to be used in the NR based simulation. A negative value causes the scale to be estimated automatically.

(10) uselog (Bool) (Default: 'no')
Use a logarithmic scale for the electron and hole densities (works better in some cases).

A.2.59 sim2/xcur/show

USAGE:
sim2/xcur/show

DESCRIPTION:
Calculate and display the net current in the simulation at certain positions.

A.2.60 sim2/ycur/show

USAGE:
sim2/ycur/show

DESCRIPTION:
Calculate and display the net current in the simulation at certain positions.

A.2.61 sim2/avg/fplot

USAGE:
sim2/avg/fplot property filename xy

DESCRIPTION:
Write 1D plot data of a quantity to a file. The values along the other axis are averaged out.

ARGUMENTS:
(1) property (Choice)
Specifies the property to plot.

```
Valid choices:
 - n    : electron number density
 - p    : hole number density
 - bg   : fixed background number density
 - g    : generation rate for electron-hole pairs
 - rf   : recombination rate will be rf*n*p
 - dn   : diffusion constant for electrons
 - dp   : diffusion constant for holes
 - nmob : electron mobility
 - pmob : hole mobility
 - eps  : relative permittivity
 - r    : recombination rate
 - V    : potential
 - jx   : current in x-direction
 - jy   : current in y-direction
 - jnx  : electron current in x-direction
 - jny  : electron current in y-direction
 - jpx  : hole current in x-direction
 - jpy  : hole current in y-direction
 - Vn   : extra electron potential
 - Vp   : extra hole potential

(2) filename (String)
Name of the file to which the plot data should be written.

(3) xy (Choice)
Specifies the axis which should remain.
Valid choices:
 - x : X axis
 - y : Y axis
```

A.2.62 sim2/line/fplot

```
USAGE:
sim2/line/fplot property filename coordinate xy

DESCRIPTION:
Write 1D plot data of a quantity to a file. The values along a specific line
are used.

ARGUMENTS:
(1) property (Choice)
Specifies the property to plot.
Valid choices:
 - n    : electron number density
 - p    : hole number density
 - bg   : fixed background number density
 - g    : generation rate for electron-hole pairs
 - rf   : recombination rate will be rf*n*p
```

```
 - dn   : diffusion constant for electrons
 - dp   : diffusion constant for holes
 - nmob : electron mobility
 - pmob : hole mobility
 - eps  : relative permittivity
 - r    : recombination rate
 - V    : potential
 - jx   : current in x-direction
 - jy   : current in y-direction
 - jnx  : electron current in x-direction
 - jny  : electron current in y-direction
 - jpx  : hole current in x-direction
 - jpy  : hole current in y-direction
 - Vn   : extra electron potential
 - Vp   : extra hole potential

(2) filename (String)
Name of the file to which the plot data should be written.

(3) coordinate (Integer)
The X or Y coordinate of the line that should be plotted.

(4) xy (Choice)
Specifies the axis which should remain.
Valid choices:
 - x : X axis
 - y : Y axis
```

A.2.63 sim2/iv

```
USAGE:
sim2/iv vint vstart vstop steps filename append maxit errlimit timeout
doublesearch densscale diffscale uselog warningsaserrors

DESCRIPTION:
Calculate an IV curve using the direct solution method. The voltage between
right and left contacts is set to 'vint-vapplied'.

ARGUMENTS:
(1) vint (Real)
The built-in potential.

(2) vstart (Real)
Start value of the applied potential.

(3) vstop (Real)
End value of the applied potential.

(4) steps (Integer) (MIN: 2) (Default: '100')
```

The number of points in the IV curve.

(5) filename (String) (Default: '*')
Name of the file to which the plot data should be written.

(6) append (Bool) (Default: 'yes')
Flag indicating if the plot data should be appended to an existing file.

(7) maxit (Integer) (MIN: 1) (Default: '1000')
Maximum number of solution iterations to perform, for each potential difference.

(8) errlimit (Real) (Default: '-1')
If the error after convergence exceeds this value, the multiresolution search will stop. A negative value disables this feature.

(9) timeout (Real) (Default: '-1')
The maximum amount of time the search for the IV curve can take. Set to a negative value to disable.

(10) doublesearch (Bool) (Default: 'false')
If set, a routine to search for the equilibrium at each voltage will be run twice. This takes a bit longer but gives better results for more difficult parameter combinations.

(11) densscale (Real) (Default: '-1')
Density scale to be used in the NR based simulation. A negative value causes the scale to be estimated automatically.

(12) diffscale (Real) (Default: '-1')
Diffusion scale to be used in the NR based simulation. A negative value causes the scale to be estimated automatically.

(13) uselog (Bool) (Default: 'no')
Use a logarithmic scale for the electron and hole densities (works better in some cases).

(14) warningsaserrors (Bool) (Default: 'yes')
Treat warnings as errors

A.2.64 math/def

USAGE:
math/def variable expression

DESCRIPTION:
Calculate a mathematical expression or define a variable and shows the result.

ARGUMENTS:
(1) variable (String)

```
Variable to store the result of the calculation in.

(2) expression (String)
Expression to evaluate
```

A.2.65 math/calc

```
USAGE:
math/calc expression

DESCRIPTION:
Calculate a mathematical expression or define a variable and shows the result.

ARGUMENTS:
(1) expression (String)
Expression to evaluate
```

A.2.66 math/calc/mu

```
USAGE:
math/calc/mu E_min E_max E_A E_D E_gap kT N_C N_V N_A N_D variable

DESCRIPTION:
Finds the chemical potential for a semiconductor. The precise units used for
energy or densities don't matter as long as the same unit is used for all
energy values (including k*T) and the same unit is used for all density values.
The chemical potential value that is calculated will be in the same units as
the energies specified.

ARGUMENTS:
(1) E_min (Real)
The chemical potential is assumed to lie in the inverval [Emin, Emax]

(2) E_max (Real)
The chemical potential is assumed to lie in the inverval [Emin, Emax]

(3) E_A (Real)
The difference between the acceptor energy level and the valence band top
energy level.

(4) E_D (Real)
The difference between the conduction band bottom energy level and the donor
energy level.

(5) E_gap (Real)
The size of the energy band gap.

(6) kT (Real)
The value of k*T, in the same units as the energies specified in the other
```

```
parameters.

(7) N_C (Real) (MIN: 0)
Effective density of states of the conduction band.

(8) N_V (Real) (MIN: 0)
Effective density of states of the valence band.

(9) N_A (Real) (MIN: 0)
Acceptor density.

(10) N_D (Real) (MIN: 0)
Donor density.

(11) variable (String)
Variable to store the result of the calculation in.
```

A.2.67 math/list

```
USAGE:
math/list

DESCRIPTION:
List the names of constants and variables currently in use.
```

A.2.68 math/clearvar

```
USAGE:
math/clearvar

DESCRIPTION:
Clear all currently defined variables.
```

A.2.69 reg1/new

```
USAGE:
reg1/new name

DESCRIPTION:
Create a new (but still empty) region with a specified name.

ARGUMENTS:
(1) name (String)
Name of the new region.
```

A.2.70 reg1/append/line

```
USAGE:
```

```
reg1/append/line name x1 x2

DESCRIPTION:
Add an area to the region with the specified name.

ARGUMENTS:
(1) name (String)
Name of the region to which the specified area should be added.

(2) x1 (Integer) (MIN: 1)
Left pixel coordinate of the area.

(3) x2 (Integer) (MIN: 1)
Right pixel coordinate of the area.
```

A.2.71 reg1/delete

```
USAGE:
reg1/delete name

DESCRIPTION:
Delete a specific region name.

ARGUMENTS:
(1) name (String)
Name of the region to delete.
```

A.2.72 reg1/clearall

```
USAGE:
reg1/clearall

DESCRIPTION:
Clear all currently defined region names.
```

A.2.73 reg1/list

```
USAGE:
reg1/list printdescription

DESCRIPTION:
List currently defined region names and their descriptions.

ARGUMENTS:
(1) printdescription (Bool) (Default: 'no')
If set to true, not only the region names but also a description (if available)
will be shown.
```

A.2.74 reg2/new

```
USAGE:
reg2/new name

DESCRIPTION:
Create a new (but still empty) region with a specified name.

ARGUMENTS:
(1) name (String)
Name of the new region.
```

A.2.75 reg2/append/rect

```
USAGE:
reg2/append/rect name x1 y1 x2 y2

DESCRIPTION:
Add an area to the region with the specified name.

ARGUMENTS:
(1) name (String)
Name of the region to which the specified area should be added.

(2) x1 (Integer) (MIN: 1)
Left pixel coordinate of the area.

(3) y1 (Integer) (MIN: 1)
Bottom pixel coordinate of the area.

(4) x2 (Integer) (MIN: 1)
Right pixel coordinate of the area.

(5) y2 (Integer) (MIN: 1)
Top pixel coordinate of the area.
```

A.2.76 reg2/append/img

```
USAGE:
reg2/append/img regionname filename xscale yscale xoffset yoffset

DESCRIPTION:


ARGUMENTS:
(1) regionname (String)
Name of the region to which the specified area should be added.

(2) filename (String)
```

Name of the image file (e.g. a PNG file) used to define this sub-region. All non-black pixels will be considered a part of this sub-region.

(3) xscale (Real) (MIN: 0) (Default: '1')
Factor by which the sub-area corresponding to the image file should be scaled in the X-direction.

(4) yscale (Real) (MIN: 0) (Default: '1')
Factor by which the sub-area corresponding to the image file should be scaled in the Y-direction.

(5) xoffset (Integer) (Default: '0')
The sub-area loaded from the image and scaled by the previous factors, will be offset by this amount in the X-direction.

(6) yoffset (Integer) (Default: '0')
The sub-area loaded from the image and scaled by the previous factors, will be offset by this amount in the Y-direction.

A.2.77 reg2/delete

USAGE:
reg2/delete name

DESCRIPTION:
Delete a specific region name.

ARGUMENTS:
(1) name (String)
Name of the region to delete.

A.2.78 reg2/clearall

USAGE:
reg2/clearall

DESCRIPTION:
Clear all currently defined region names.

A.2.79 reg2/list

USAGE:
reg2/list printdescription

DESCRIPTION:
List currently defined region names and their descriptions.

ARGUMENTS:
(1) printdescription (Bool) (Default: 'no')

If set to true, not only the region names but also a description (if available) will be shown.