Manual for TOM (Toolbox for OMEN in MATLAB)

Version 0.1

Christian Funck & Matthias Dittberner

July 5th, 2013

Contents

Chapter 1

Toolbox for OMEN in MATLAB (TOM)

The Toolbox for OMEN in MATLAB (TOM) was programmed to simulate spherical nanocrystals and to visualize the results of the simulations. Though we focused on spherical structures, the code is held abstract and slim in order to make extensions to other structures, such as nanowires, fairly easy. The main aim of TOM is to offer OMEN users a toolbox, which includes the following 3 main tasks:

1. Automatization of the OMEN simulation process
   When it comes to simulating Quantum Dots (QDs) with different parameters, users do not want to spend a lot of time on writing command files for OMEN and starting each OMEN task via the shell, but rather enter the main parameters into a Graphical User Interface (GUI) and let TOM do the rest (see section 1.3.2). This makes overnight simulations for large parameter sets possible.
2. Overview of all simulations done in the past
   All information respectively parameters of past simulations can be displayed in a GUI. Exporting selected simulations and visualizing them is possible as well (see section 1.3.3)
3. Visualization of simulation data
   Different kinds of plots are available within the toolbox, such as visualizing band gaps, wave functions or quantum dot structures (see section 1.3.4).

1.1 Installing the Software

You can get a version of TOM at the Laboratory for Nanoelectronics (LNE) at ETH Zurich. Copy the folder `TOM` to any directory on your computer. Now it is necessary to define some paths. Open therefore `TOM/System/initTOM.m` and change all fields, that are listed below. The field *root* contains the path, where the folder `TOM` is saved. Similarly, the paths of the simulations folder and of the OMEN executable are stored in the corresponding fields. Note that these paths do not have to direct to a subfolder or TOM, i.e. simulation data can be stored anywhere on the computer.

```
config.root         = '/usr/home/TOM/';
config.OMEN         = [config.root, 'OMEN_ethz-amd64'];
config.vOMEN        = '04May2013';
config.simulations  = [config.root, 'Simulations/'];
```

1.2 Using a newer version of OMEN

If you would like to use a new version of OMEN , please make the following changes. Open the `TOM/System/initTOM.m` file and change `config.vOMEN` to the date of the new version's release. Afterwards copy the new OMEN executable to the path defined in `config.OMEN`. It might be possible that simulations do not run with the new executable, as the rights are not set to *executable*. In this case, right click on the executable, select properties and afterwards *Run as executable*.

1.3 Using TOM

1.3.1 Basic concept

The **first step** when you want to work with TOM is its initialisation. Therefore add the TOM folder including subfolders to the MATLAB path, type `initTOM.m` in the command window and hit enter.
The command `help TOM` will provide you with all available functionalities including a description how to use them. Another way to get help is by opening the TOM manual, which can also be accessed via the help command or the menu lists in the GUIs.
Table 1.1 gives a quick overview of the most important function parameters, that are used regularly in the TOM code.
Parameters for the simulation are stored and passed as objects to functions of the class *Qdot*, further referred to as Quantum Dot Object (QDO). The class provides properties for all parameters necessary for the simulation, as well as properties for administrative purposes, such as the specific simulation folder. Parameters relating to the geometry are stored in an object of subclass *Geometry*. Since more than one material is possible, the geometry property is often an array of objects of class Geometry. To instantiate a QDO, one can call the constructor with no arguments, which will create an empty QDO. Alternatively, a string with the material name can be passed as an argument, in which case the new QDO will be constructed based on parameters defined in an external file, located in the folder *Classes*. The class *Qdot* provides some methods for basic displaying of some selected parameters, such as *getSelParams*.

Example. Creating a *Qdot* object based on default parameters and set the radius to 4

```
myQdot = Qdot('CdS_CdSe');
myQdot.geometry(2).radius = 4;
```

DIR	Directory path
EDO	Experimental data object
EDOA	Array of experimental data objects
QDO	Quantum dot object
QDOA	Array of quantum dot objects
propertyName	Property name of an EDO or QDO
BGap	Band Gap
NMod	Number of modes
NOrb	Number of orbitals
band	CB: Conduction band or VB: Valence Band
tol	tolerance

Table 1.1: Important function parameters

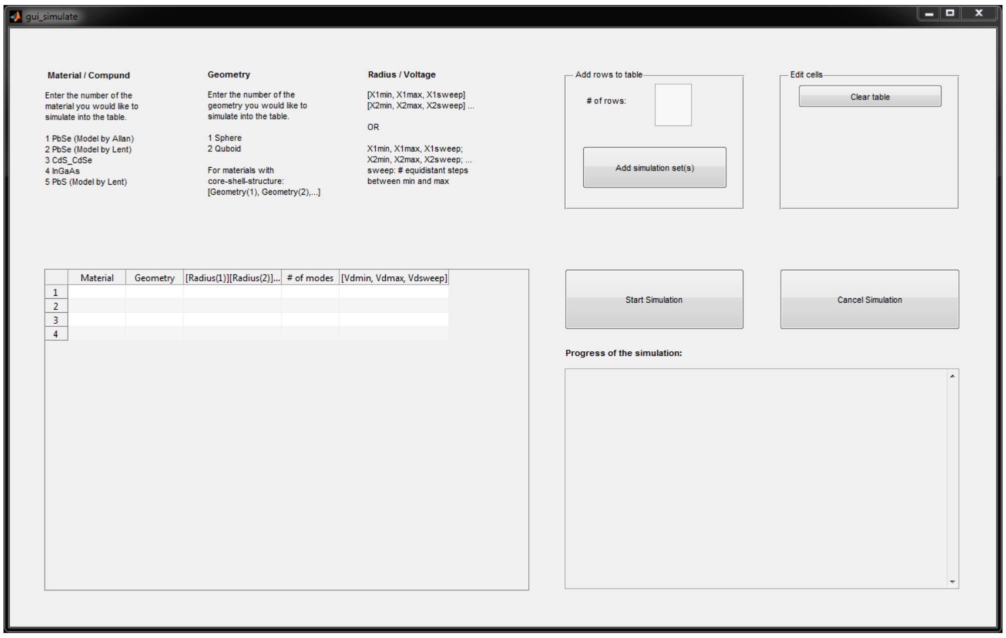
1.3.2 Running a simulation

Type `gui_simulate` in the MATLAB command window and hit enter. The window as in figure 1.1 will open. A simulation set is defined as one row of the table, i.e one material with all kinds of sweeps. You can add more simulation sets by using the *Add rows to the table* panel. It is possible to copy and paste single cells of the table using the appropriate short cuts of your MATLAB default keyboard setup (`CTRL+C & CTRL+V` Windows setting, `ALT+W & CTRL+Y` Emacs setting). The columns are filled as follows:

1. Material
   Enter the number for the material you would like to simulate, according to the *Material / Compound* list.

*Material Name*_lent	Simulations done with tight binding parameters by Lent
*Material Name*_allan	Simulations done with tight binding parameters by Allan

2. Geometry
   Enter the number of the geometry given in the *Geometry* list. Very important in the case of materials with shells is, that you have to enter the geometry type of the core and the shell. The geometry types are separated with a comma.



Material / Compund

Enter the number of the material you would like to simulate into the table.

1 PbSe (Model by Allan)
2 PbSe (Model by Lent)
3 CdS_CdSe
4 InGaAs
5 PbS (Model by Lent)

Geometry

Enter the number of the geometry you would like to simulate into the table.

1 Sphere
2 Quboid

For materials with core-shell-structure:
[Geometry(1), Geometry(2),...]

Radius / Voltage

[X1min, X1max, X1sweep]
[X2min, X2max, X2sweep] ...

OR

X1min, X1max, X1sweep;
X2min, X2max, X2sweep; ...
sweep: # equidistant steps between min and max

	Material	Geometry	[Radius(1)][Radius(2)]...	# of modes	[Vdmin, Vdmax, Vdsweep]
1					
2					
3					
4					

Figure 1.1: The gui_simualte window

Example. For a spherical CdS-CdSe quantum dot the cell would look like this: 1, 1

3. Radius [nm]
The radius has to be entered in a specific way. The syntax is:
`[Rmin(1), Rmax(1), Rsweep(1)][Rmin(2), Rmax(2), Rsweep(2)]...`

Rmin(i)	smallest radius of the ith material to be simulated
Rmax(i)	largest radius of the ith material to be simulated
Rsweep(i)	number of equidistant points between Rmin(i) & Rmax(i)

Example. For the simulation of a single spherical PbS quantum dot with radius 3.5 nm you would enter: [3.5,3.5,1]

4. # of modes
Enter a number of modes you would like to calculate.

5. Electric field in units [V/nm]
The electric field parameter is entered in the same way as the radius. You find more information under the remark below.

6. update_bs_target
For higher electric fields OMEN cannot differentiate conduction and valence band, therefore the user has to make a guess where the band gap might be. OMEN will then simulate the # of modes around this given energy (bs_target). If you would like to make use of this option enter 1 in the *update_bs_target* field otherwise 0. Please note, that if you choose 0, you will have to enter any numerical value (e.g. 0) in the *bs_target* field as well, otherwise the GUI cannot process the data.

7. bs_target [eV]
The energy value around which the energy levels are calculated.

8. Permute
Enter 0, if the radii vectors should be combined element wise, i.e. first element of R(1) and R(2) give a simulation, second element of R(1) and R(2) give a simulation and so on. Each radius pair will be simulated with all possible electric fields specified. Selecting 1 will permute all possible radii with each other and all electric fields.

Example. For a spherical CdS-CdSe QD the input cell would look like this: [1,4,4][5,6,2]
If you select Permute = 1, TOM will calculate all possible permutations and generate therefore 8 quantum dots:

Quantum dot	1	2	3	4	5	6	7	8
Core radius in nm	1	1	2	2	3	3	4	4
Shell radius in nm	5	6	5	6	5	6	5	6

Example. For a spherical CdS-CdSe quantum dot the input cell would look like this: [1,2,2][3,4,2] If you select Permute = 0, TOM will combine elementwise the entries of the two vectors:

Quantum dot	1	2
Core radius in nm	1	2
Shell radius in nm	3	4

Remark (Vectors & Matrices). As the MATLAB GUIs do not directly accept vectors or matrices in standard MATLAB syntax, the parameters have to be entered as a string and are converted to matrices later on.
There are different ways to enter the parameters. Use the one that is the most clear for you.

```
 a,b,c,...,d                  or
[a,b,c,...,d]                 becomes a double vector

V = [ a b c ... d]


[a,b,c][d,e,f]...[g,h,j]      or
 a,b,c; d,e,f;...;g,h,j       or
[a,b,c; d,e,f;...;g,h,j]      becomes a double matrix

M = [ a b c
      d e f
      . . .
      g h j ]

where a,b,...,j are doubles as strings.
```

These input styles can be applied to the columns *Geometry*, *Radius* and *Voltage*.
You can use as many spaces as you want in between. The first and the last brace are not necessary if only a vector is entered or if rows are separated by semicolons.

After entering all the necessary parameters proceed by clicking `Start Simulation`. The *Progress of the simulation* panel will keep you informed about the warnings, wrong entered parameters and the current status of the simulation.

1.3.3 Display simulation information

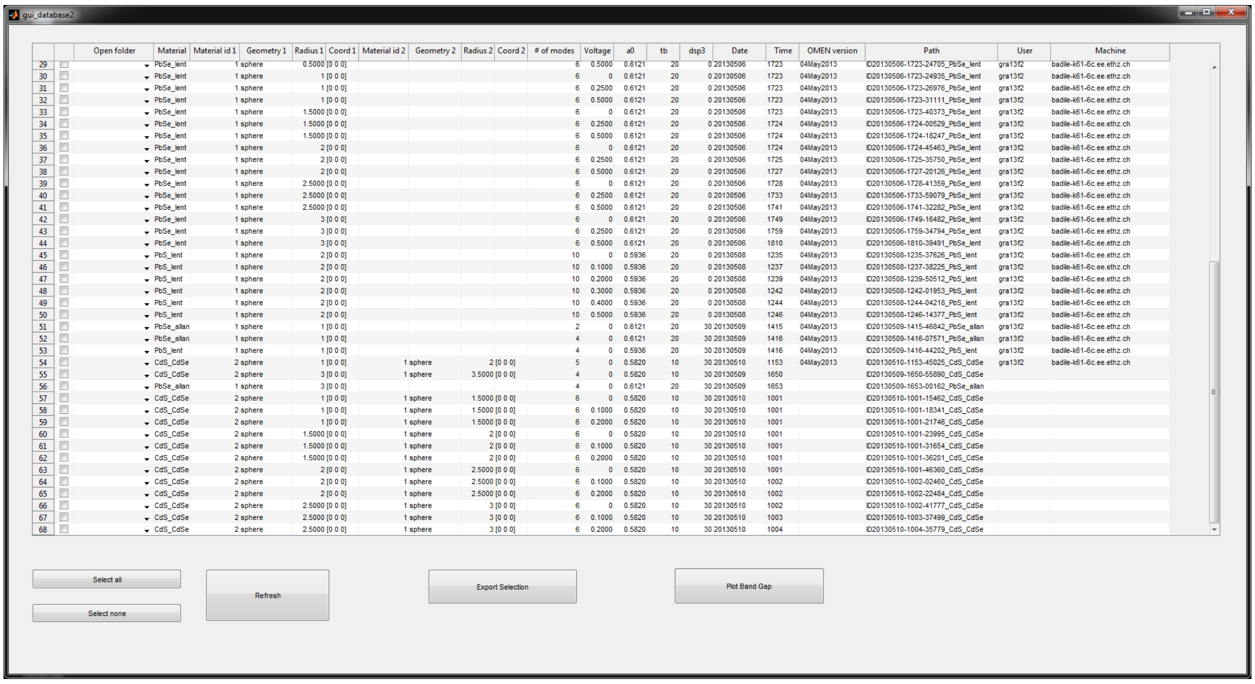
There are two ways of displaying simulation information. Either you can see the whole database (all simulations that are stored in the `Simulations` folder) or only specific data, for example only PbS simulation data.

Remark. TOM uses objects (called *qdotObj*) to store all important data of simulations such as the simulation parameters, date of simulation etc. All operations are done using these objects or arrays thereof (QDOA). How to create such an array is described in section 1.3.5.

Displaying the whole database can be done by typing `gui_db` into the MATLAB command window. A window as in figure 1.2 will appear showing all parameters and technical information of each simulation. Please note, that according to the size of the database, it might take some seconds to load all data. If you only want to display simulation data for QDOs in a QDOA, you can also call `gui_db(QDOA)`.
Within the GUI you can sort the simulations with the column header, select simulations and plot them, open the directory of a simulation or even export a selection, which is then available as an Quantum Dot Object Array (QDOA) in the main workspace entitled *ExportedDB*.

1.3.4 Plotting

Here follows a short description of functions, which can be used for basic visualization of the data obtained by the OMEN simulation. For a more detailed description, please refer to the code. Note that these functions are based on the directory structure created by the simulation function `simAll.m`, i.e. to work properly, the simulation data, as well as the corresponding QDO must be located in their own folder, with the folder name specified in `QDO.path` property.



		Open folder	Material	Material id 1	Geometry 1	Radius 1	Coord 1	Material id 2	Geometry 2	Radius 2	Coord 2	# of modes	Voltage	a0	tb	dsp3	Date	Time	OMEN version	Path	User	Machine
29	☐	▾	PbSe_lent	1	sphere	0.5000	[0 0 0]					6	0.5000	0.6121	20	0	20130506	1723	04May2013	ID20130506-1723-24705_PbSe_lent	gra13f2	badile-k61-6c.ee.ethz.ch
30	☐	▾	PbSe_lent	1	sphere	1	[0 0 0]					6	0	0.6121	20	0	20130506	1723	04May2013	ID20130506-1723-24935_PbSe_lent	gra13f2	badile-k61-6c.ee.ethz.ch
31	☐	▾	PbSe_lent	1	sphere	1	[0 0 0]					6	0.2500	0.6121	20	0	20130506	1723	04May2013	ID20130506-1723-26976_PbSe_lent	gra13f2	badile-k61-6c.ee.ethz.ch
32	☐	▾	PbSe_lent	1	sphere	1	[0 0 0]					6	0.5000	0.6121	20	0	20130506	1723	04May2013	ID20130506-1723-31111_PbSe_lent	gra13f2	badile-k61-6c.ee.ethz.ch
33	☐	▾	PbSe_lent	1	sphere	1.5000	[0 0 0]					6	0	0.6121	20	0	20130506	1723	04May2013	ID20130506-1723-40373_PbSe_lent	gra13f2	badile-k61-6c.ee.ethz.ch
34	☐	▾	PbSe_lent	1	sphere	1.5000	[0 0 0]					6	0.2500	0.6121	20	0	20130506	1724	04May2013	ID20130506-1724-00529_PbSe_lent	gra13f2	badile-k61-6c.ee.ethz.ch
35	☐	▾	PbSe_lent	1	sphere	1.5000	[0 0 0]					6	0.5000	0.6121	20	0	20130506	1724	04May2013	ID20130506-1724-18247_PbSe_lent	gra13f2	badile-k61-6c.ee.ethz.ch
36	☐	▾	PbSe_lent	1	sphere	2	[0 0 0]					6	0	0.6121	20	0	20130506	1724	04May2013	ID20130506-1724-45463_PbSe_lent	gra13f2	badile-k61-6c.ee.ethz.ch
37	☐	▾	PbSe_lent	1	sphere	2	[0 0 0]					6	0.2500	0.6121	20	0	20130506	1725	04May2013	ID20130506-1725-35750_PbSe_lent	gra13f2	badile-k61-6c.ee.ethz.ch
38	☐	▾	PbSe_lent	1	sphere	2	[0 0 0]					6	0.5000	0.6121	20	0	20130506	1727	04May2013	ID20130506-1727-20126_PbSe_lent	gra13f2	badile-k61-6c.ee.ethz.ch
39	☐	▾	PbSe_lent	1	sphere	2.5000	[0 0 0]					6	0	0.6121	20	0	20130506	1728	04May2013	ID20130506-1728-41359_PbSe_lent	gra13f2	badile-k61-6c.ee.ethz.ch
40	☐	▾	PbSe_lent	1	sphere	2.5000	[0 0 0]					6	0.2500	0.6121	20	0	20130506	1733	04May2013	ID20130506-1733-59079_PbSe_lent	gra13f2	badile-k61-6c.ee.ethz.ch
41	☐	▾	PbSe_lent	1	sphere	2.5000	[0 0 0]					6	0.5000	0.6121	20	0	20130506	1741	04May2013	ID20130506-1741-32282_PbSe_lent	gra13f2	badile-k61-6c.ee.ethz.ch
42	☐	▾	PbSe_lent	1	sphere	3	[0 0 0]					6	0	0.6121	20	0	20130506	1749	04May2013	ID20130506-1749-16482_PbSe_lent	gra13f2	badile-k61-6c.ee.ethz.ch
43	☐	▾	PbSe_lent	1	sphere	3	[0 0 0]					6	0.2500	0.6121	20	0	20130506	1759	04May2013	ID20130506-1759-34794_PbSe_lent	gra13f2	badile-k61-6c.ee.ethz.ch
44	☐	▾	PbSe_lent	1	sphere	3	[0 0 0]					6	0.5000	0.6121	20	0	20130506	1810	04May2013	ID20130506-1810-39491_PbSe_lent	gra13f2	badile-k61-6c.ee.ethz.ch
45	☐	▾	PbS_lent	1	sphere	2	[0 0 0]					10	0	0.5936	20	0	20130508	1235	04May2013	ID20130508-1235-37626_PbS_lent	gra13f2	badile-k61-6c.ee.ethz.ch
46	☐	▾	PbS_lent	1	sphere	2	[0 0 0]					10	0.1000	0.5936	20	0	20130508	1237	04May2013	ID20130508-1237-38225_PbS_lent	gra13f2	badile-k61-6c.ee.ethz.ch
47	☐	▾	PbS_lent	1	sphere	2	[0 0 0]					10	0.2000	0.5936	20	0	20130508	1239	04May2013	ID20130508-1239-50512_PbS_lent	gra13f2	badile-k61-6c.ee.ethz.ch
48	☐	▾	PbS_lent	1	sphere	2	[0 0 0]					10	0.3000	0.5936	20	0	20130508	1242	04May2013	ID20130508-1242-01953_PbS_lent	gra13f2	badile-k61-6c.ee.ethz.ch
49	☐	▾	PbS_lent	1	sphere	2	[0 0 0]					10	0.4000	0.5936	20	0	20130508	1244	04May2013	ID20130508-1244-04218_PbS_lent	gra13f2	badile-k61-6c.ee.ethz.ch
50	☐	▾	PbS_lent	1	sphere	2	[0 0 0]					10	0.5000	0.5936	20	0	20130508	1246	04May2013	ID20130508-1246-14377_PbS_lent	gra13f2	badile-k61-6c.ee.ethz.ch
51	☐	▾	PbSe_allan	1	sphere	1	[0 0 0]					2	0	0.6121	20	30	20130509	1415	04May2013	ID20130509-1415-46842_PbSe_allan	gra13f2	badile-k61-6c.ee.ethz.ch
52	☐	▾	PbSe_allan	1	sphere	1	[0 0 0]					4	0	0.6121	20	30	20130509	1416	04May2013	ID20130509-1416-07571_PbSe_allan	gra13f2	badile-k61-6c.ee.ethz.ch
53	☐	▾	PbS_lent	1	sphere	1	[0 0 0]					4	0	0.5936	20	30	20130509	1416	04May2013	ID20130509-1416-44202_PbS_lent	gra13f2	badile-k61-6c.ee.ethz.ch
54	☐	▾	CdS_CdSe	2	sphere	1	[0 0 0]	1	sphere	2	[0 0 0]	5	0	0.5820	10	30	20130510	1153	04May2013	ID20130510-1153-45025_CdS_CdSe	gra13f2	badile-k61-6c.ee.ethz.ch
55	☐	▾	CdS_CdSe	2	sphere	3	[0 0 0]	1	sphere	3.5000	[0 0 0]	4	0	0.5820	10	30	20130509	1650		ID20130509-1650-55890_CdS_CdSe		
56	☐	▾	PbSe_allan	1	sphere	3	[0 0 0]					4	0	0.6121	20	30	20130509	1653		ID20130509-1653-00162_PbSe_allan		
57	☐	▾	CdS_CdSe	2	sphere	1	[0 0 0]	1	sphere	1.5000	[0 0 0]	6	0	0.5820	10	30	20130510	1001		ID20130510-1001-15462_CdS_CdSe		
58	☐	▾	CdS_CdSe	2	sphere	1	[0 0 0]	1	sphere	1.5000	[0 0 0]	6	0.1000	0.5820	10	30	20130510	1001		ID20130510-1001-18341_CdS_CdSe		
59	☐	▾	CdS_CdSe	2	sphere	1	[0 0 0]	1	sphere	1.5000	[0 0 0]	6	0.2000	0.5820	10	30	20130510	1001		ID20130510-1001-21746_CdS_CdSe		
60	☐	▾	CdS_CdSe	2	sphere	1.5000	[0 0 0]	1	sphere	2	[0 0 0]	6	0	0.5820	10	30	20130510	1001		ID20130510-1001-23995_CdS_CdSe		
61	☐	▾	CdS_CdSe	2	sphere	1.5000	[0 0 0]	1	sphere	2	[0 0 0]	6	0.1000	0.5820	10	30	20130510	1001		ID20130510-1001-31654_CdS_CdSe		
62	☐	▾	CdS_CdSe	2	sphere	1.5000	[0 0 0]	1	sphere	2	[0 0 0]	6	0.2000	0.5820	10	30	20130510	1001		ID20130510-1001-36201_CdS_CdSe		
63	☐	▾	CdS_CdSe	2	sphere	2	[0 0 0]	1	sphere	2.5000	[0 0 0]	6	0	0.5820	10	30	20130510	1001		ID20130510-1001-46360_CdS_CdSe		
64	☐	▾	CdS_CdSe	2	sphere	2	[0 0 0]	1	sphere	2.5000	[0 0 0]	6	0.1000	0.5820	10	30	20130510	1002		ID20130510-1002-02460_CdS_CdSe		
65	☐	▾	CdS_CdSe	2	sphere	2	[0 0 0]	1	sphere	2.5000	[0 0 0]	6	0.2000	0.5820	10	30	20130510	1002		ID20130510-1002-22484_CdS_CdSe		
66	☐	▾	CdS_CdSe	2	sphere	2.5000	[0 0 0]	1	sphere	3	[0 0 0]	6	0	0.5820	10	30	20130510	1002		ID20130510-1002-41777_CdS_CdSe		
67	☐	▾	CdS_CdSe	2	sphere	2.5000	[0 0 0]	1	sphere	3	[0 0 0]	6	0.1000	0.5820	10	30	20130510	1003		ID20130510-1003-37499_CdS_CdSe		
68	☐	▾	CdS_CdSe	2	sphere	2.5000	[0 0 0]	1	sphere	3	[0 0 0]	6	0.2000	0.5820	10	30	20130510	1004		ID20130510-1004-35779_CdS_CdSe		

Figure 1.2: The gui_database window

Plotting the probability density

These functions all take an array of *Qdot* objects (QDOA) as an input. Additionally, it has to be specified which modes should be displayed, as well as the band (conduction or valence band). The visualisation will then be done for every one these objects, and for all specified eigenmodes. The most useful function is probably *plotEV3Dmax.m*

```
function plotEV3D(QDOA, band, Mod)
```

Plot the atoms of the quantum dot, their color indicating the probability density of an electron or hole. Red corresponds to high, blue to low probability.

```
function plotEV3DcrossSection(QDOA, NMod)
```

This function produces similar plots to the above, but it plots two cross sections of the quantum dot, for valence and conduction band respectively, in one window.

```
function plotEV3Dmax(QDOA, band, probLim, Mod)
```

Again very similar to the *plotEV3D*, but the color code is simplified. The atoms with very high probability densities are red, the ones with high probability yellow, the rest transparent. The color is determined in the following way: The sum of the probabilities of all red locations is smaller than a probability value specified in *probLim(1)*. An analogous argument is applied for the yellow marked atoms.
This function makes it a lot easier to see how the wave function roughly looks like and how it changes from one mode to the next.

```
function plotEVAlongAxis(QDOA, propertyName, startPoint, direction, plotGrid,
tolerance, NMod, band)
```

Plot the probability density along an arbitrary axis through the crystal. The data for all elements of QDOA is plotted in the same plot, thus making it easier to compare quantum dots with different parameters. The axis is specified through *startPoint* and *direction*, including a tolerance, which is the maximum distance which an atom can deviate from the specified line. Depending on the direction, the tolerance has to be adjusted to include a sufficient number of atoms. To check this, it is useful to specify the input argument *gridPlot*, which will plot the atoms, the chosen axis, and highlight the atoms on the line in red. However, this function is probably only suitable for large QDs. Furthermore an averaging over neighbouring atoms would be recommendable.

```
function compareEV(QDOA, band, NMod, tol, propertyName, showGrid)
```

Plots the same as *plotEVAlongAxis*, but for three different directions (x, y, z-axis), and arranges them in subplots in one figure.

Plotting energies

```
function plotBandGap(QDOA)
```

Plots the band gaps of a QDOA in dependence of radius and applied voltage. Calling the function without an input argument, will create plots for the whole database.

```
function plotEnergyLevels(QDOA)
```

Plots all simulated energy levels of an array of QDOs and highlights the band gap.

```
function plotVoltBandGap(QDOA)
```

Plots the band gap against voltage for a constant radius. The QDOs must have the same radius/radii.

1.3.5 Additional tools

In order to know which parameter sets have already been simulated, there are some useful tools, which can be found in the folder *Functions/QdotUtils*. The basic principle is to get all parameters which were simulated, which can then be displayed in the GUI, filtered, deleted and so on.

Getting the parameters from all simulations

```
function getQDOA()
```

This is done by loading the QDOs of all performed simulations from their folders, and storing them in a QDOA.

Filtering

```
function filtered = filterQDOA(QDOA, propertyName, value, mode, tol)
```

The filtering can be applied to any QDOA, and returns a subset of this array, matching specified criteria. The argument *propertyName* specifies the *Qdot* property which is compared to *value*. The filter criteria are specified by selecting a mode of filtering. The following filtering modes are available:

1. The property exactly matches *value*.
2. The property lies within a range of values, specified by a vector: *value* = [min max]
3. The property approximately matches *value*. For numeric properties this is specified using a tolerance. For string properties the *value* should be a substring of the property.
4. Filter for a constant difference between two properties. The property names are specified in a cell array: *propertyName* = {*propertyName1, propertyName2*}
5. Filter for a constant ratio between two properties.

The last two modes are especially interesting for selecting QDOs with two or more materials, to find the objects with a specified shell-thickness.

Example. Filter for QDOs with different properties:

- selection 1: only QDOs with radius = 3.5nm

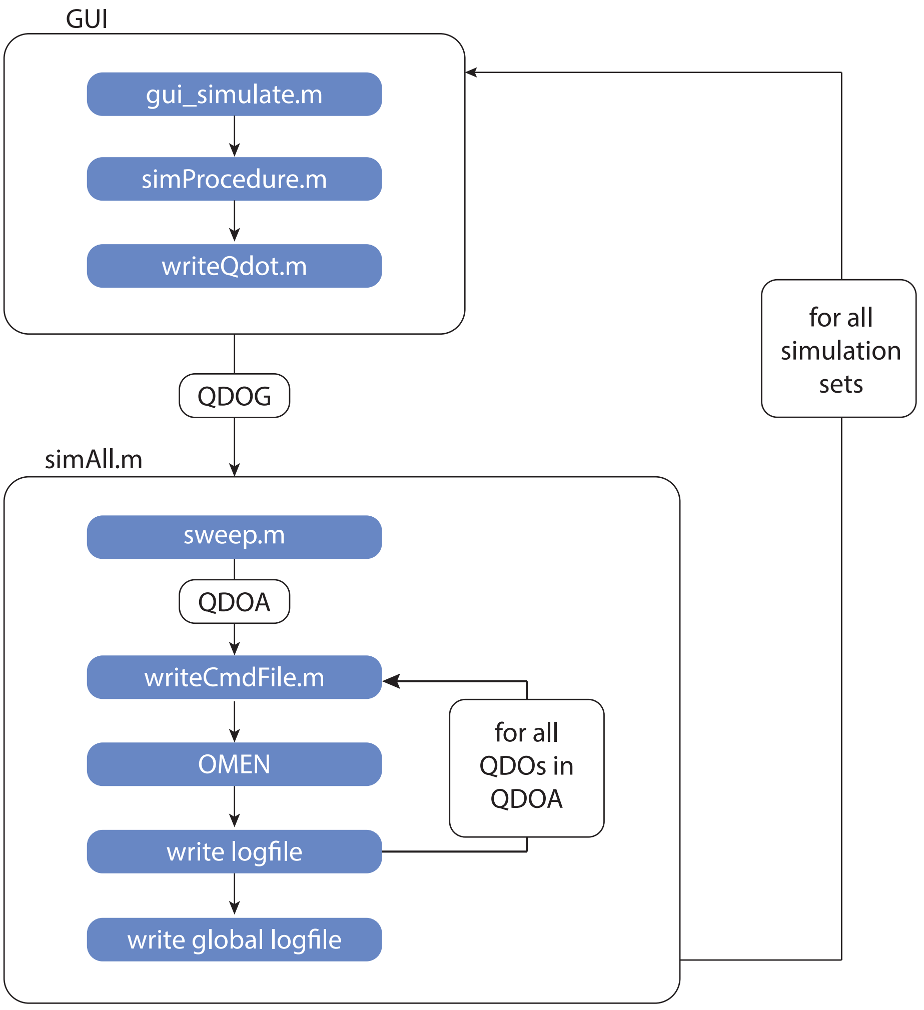


Figure 1.3: Workflow of the simulation procedure.

- selection 2: only QDOs with a constant difference of 0.7nm (with a tolerance of +/- 10%) between radius of material 1 and radius of material 2.

- selection 3: only QDOs with a material containing Pb.

```
            myQDOA = getQDOA;
            sel1=filterQDOA(myQDOA,'geometry(1).radius',3.5,1,0);
            sel2=filterQDOA(myQDOA,{'geometry(1).radius','geometry(2).radius
'}, 0.7, 4, 0.1);
            sel3=filterQDOA(myQDOA,'mat_name','Pb',2,0);
```

1.4 Maintaining TOM

1.4.1 General Structure

TOM stores all files of one simulation in a folder. To avoid conflicts in the data storage, the folder names are bijective. They have the following form: `IDyyyymmdd-hhmm-ssfff_MATNAME`. The name contains the date and time (including milliseconds) when the simulation was started as well as the name of the material, that is simulated. Figure 1.5 illustrates the structure of a simulation folder. Note, that if you simulate with parameter `update_bs_target = 1`, OMEN will not create files for valence and conduction band, but rather write all eigenenergies into `VB_E_0_0.dat` or `CB_E_0_0.dat`. If you would like to make changes in TOM or add new structures, such as nanowires, it is useful to get familiar with the workflow of the the software. Figure 1.3 illustrates the process.

```
/root
├── TOM
│   ├── Classes
│   ├── DataUtils
│   ├── GUI
│   ├── Plotting
│   ├── SimProcedure
│   └── System
├── OMEN
│   └── OMEN executables
└── Simulations
    ├── ID*
    ├── ⋮
    └── log
```

Figure 1.4: The TOM structure by default

```
ID*
├── CB_E_0_0.dat
├── CB_V_0_0.dat
├── H_0.dat
├── Layer_Matrix.dat
├── qdot_cmd
├── qdotObj.mat
├── simlog_yyyymmdd_hhmm_ssfff
├── VB_E_0_0.dat
└── VB_V_0_0.dat
```

Figure 1.5: Structure of a simulation set

1.4.2 Simulation

The simulation parameters entered in the GUI are passed to a function, which will then start the simulation with OMEN. The parameters are stored in QDOs. Since it is often desirable to simulate over a range of parameters, some parameters (i.e. the radii and the electric field) are at this stage not scalars, but vectors, containing starting value, end value and number of steps in between. Such a generic QDO is further refered to as QDOG.

The QDOG is passed to the function *simAll.m*, which will simulate all desired parameter combinations, by performing the following steps:

1. An array of QDOs (QDOA) with all combinations of the parameter is created in the function *sweep.m* from the QDOG. Supported sweep parameters are currently radius and electric field. This could easily be extended to other parameters by modifying the function *sweep.m.*

2. The elements in the QDOA are then simulated one after the other, and all generated data is saved in seperate folders, named ID*timestamp_ material*:

   (a) The OMEN command-file *qdot_ cmd* is written, using the function *writeCmdFile.m.*

   (b) The OMEN simulation will be started using the MATLAB function *unix*, which calls the operating system, to execute the specified command.

   (c) A logfile is written, recording the duration and the success of the simulation, as well as the console output of OMEN. The success of the simulation is checked by inspecting whether the desired files were created.

3. After the simulation of all elements in the QDOA, an additional logfile is written, and saved in the folder *log*. It gives information about the success of all simulations, and thus provides an easy way to check if (and which) simulations failed.

4. Returned to the calling function is the QDOA, as well as a vector indicating the success of every simulation.

Note that it is not strictly necessary to start the simulations using the GUI. The QDOG containing the desired parameters can also be created with standard MATLAB syntax, and passed subsequently to the function *simAll.m*, as is shown in the following example.

Example. Simulate PbS quantum dots with radii 1.5, 2, 2.5, 3nm.

```
myQdot = Qdot('PbS_lent');
myQdot.geometry.radius = [1.5 3 4];
simAll(myQdot);
```

1.5 Q & A

Why is opening gui_db is not possible? It might be that there are failed simulations in the database. Please delete the corresponding folders and try again.

How can I stop the simulation? MATLAB is blocked during the simulation process. You might have to abort the process by pushing `CTRL+C` a couple of times.

Why does the simulation not start? You might have to make to set execution rights for the OMEN executable: Tick 'Run as ' after right clicking the OMEN executable and selecting properties.

Acronyms

CQD	Colloidal Quantum Dot. *Glossary:* Colloidal Quantum Dot
GUI	Graphical User Interface. 5–8
LNE	Laboratory for Nanoelectronics. 5
QD	Quantum Dot. 5
QDO	Quantum Dot Object. 6, 8, 10
QDOA	Quantum Dot Object Array. 8, 10
TEM	Transmission electron microscopy. *Glossary:* Transmission electron microscopy
TOM	Toolbox for OMEN in MATLAB. 5, 6, 11