ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Simulation of Quantum Dots

Group project
Laboratory for Nanoelectronics
Prof. Vanessa Wood
Integrated Systems Laboratory
Department of Information Technology & Electrical Engineering
ETH Zurich

Handed in by
Christian Funck & Matthias Dittberner

Date
June 4, 2013

Supervisor: Prof. Vanessa Wood
Prof. Mathieu Luisier

Christian Funck
Plattenstrasse 71
8032 Zurich
09-920-177
cfunck@student.ethz.ch

Matthias Dittberner
Luegislandstrasse 269
8051 Zurich
09-913-450
mdittber@student.ethz.ch

Contents

Chapter 1

QD Solar Cells

This chapter gives a brief overview over the fundamental principles of photovoltaic cells and QD solar cells in particular, then focusing on the example of PbS cells to give some specific information.

1.1   Solar Cell Principles

The basic idea of a solar cell is to convert light into electrical power. Light is absorbed in a material, thus generating an electron-hole pair. The generated electrons and holes must then be separted and conducted to electrodes attached to the material. The accumulation of carriers at the electrodes generates a potential difference, and a current will flow between the electrodes if a load is connected.

The carrier separation can be achieved by an electirc field inside the material. Different types of solar cells exist based on different approaches. The classical silicon solar cell uses a semiconductor p-n junction: When a photon (with energy greater than the band gap) is absorbed in the semiconductor, an electron is promoted from the valence to the conduction band. Near the junction, these photogenerated electrons and holes are swept away to different sides due to the built-in electric field of the p-n junction (fig. 1.1).

Other approaches are also widely used, for example Schottky contacts (metal-semiconductor interface) or semiconductor-liquid interfaces.

The electrical behaviour of a photovoltaic cell can be well described by its I-V characteristics (fig. 1.2). In the dark, the cell behaves like a diode (in the case of a p-n junction solar cell this seems fairly obvious). Under illumination, the curve is shifted vertically downwards. It crosses now the fourth quadrant, where the electrical power $P = IV$ is negative, which indicates that power is delivered to the load.

To characterise the solar cell, there are some common parameters, which are mentioned in the following paragraph. The open-circuit voltage V_{OC}, which is the maximum voltage provided by the cell, and directly related to the energy band structure and thus built-in potential. The short-circuit current I_{SC}, giving the maximum current if the electrodes are connected. I_{SC} is proportional to the carrier density under illumination and to the carrier mobility, which are therefore important parameters to maximize.

An often used quantity is the *fill factor (FF)*, which gives the ratio between the maximum power $P_M = I_M V_M$ and the product $I_{SC} V_{OC}$. It describes how well behaved the I-V characteristics is (the more the curve approaches a rechtangular shape, the better).
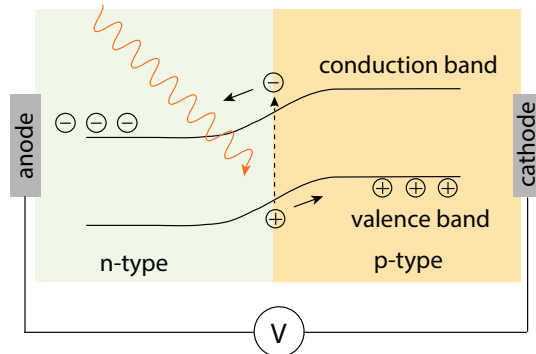

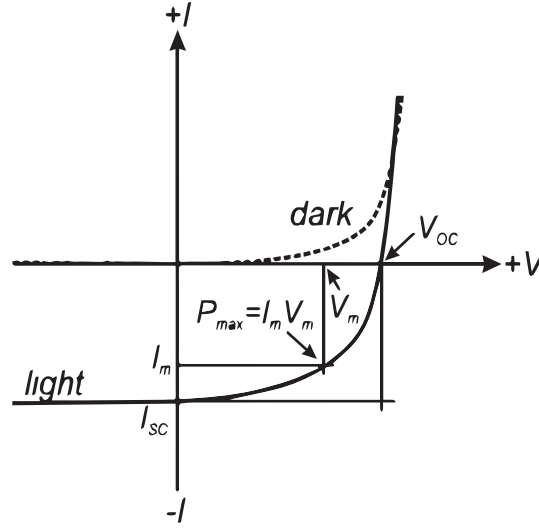Figure 1.1: Principle of a solar cell using a p-n semiconductor junction.

Figure 1.2: Typical I-V characteristics of a solar cell. SOURCE: [3, p.427]

Of practical importance for comparing solar cells is the power conversion efficiency η, giving the ratio of optical power which is converted into electrical power.

$$\eta = \frac{P_{max}}{P_{in}} = FF \frac{I_{SC} V_{OC}}{P_{in}}$$

Usually an optical source called AM1.5 is used, whose spectral intensity distribution matches that of sunlight reaching the earth's surface at an angle of 48.2°. [3, pp.426-433]

1.2 Solar Cells using cQDs

Semiconductor nano crystal solids, can be used for solar cells. Thus nano crystals (NC) made of CdSe, CdTe, PbSe, PbS and many more can be used for this purpose. Like with bulk semiconductors, heterojunction solar cells are possible (using materials with different band gaps), such as CdSe-CdTe cells [3, p.430]. Similarly, using Schottky-contacs is also an option.

Using NC solids made of QDs offers several advantages. One big advantage is the possibilty of choosing the size of the band gap by controlling the size of the QDs, which can be done easily during the synthesis of CQDs. Controlling the band gap means essentially choosing the spectrum which can be absorbed, and consequently cells which can make use of a broad spectrum can be engineered.
Furthermore QD solar cells are easy to fabricate and that at low costs. Large-scale production would also be possible [3, p.447].
Using NC solids, there are promising perspectives for more advanced techniques, in order to increase efficiency. These invlove e.g. carrier multiplication (the absorption of one highly energetic photon causes the creation of multiple electron-hole pairs) or hot carrier solar cells (electrons in higher energy states in the conduction band are extracted before they relax to the band edge and lose some energy).

However, there are some problems which have to be overcome. One is the (air-) stability and lifespan of QD based photovoltaic cells. In many cases, the devices lose dramatically in efficiency after some time, which can be as short as hours or even minutes [7, p.26]. Another problem is the currently rather low efficiency of the cells (maxium acchieved efficiency of around 7% for a PbS cell [5, p.1]). One issue leading to reduced efficiency is the presence of undesired states in the band gap (mid-gap or trap states). These arise due to 'dangling' (unbound) bonds of surface atoms, and are significant in a QD, since its surface to volume ratio is high. So-called *passivation* of the surface is thus very important.
Furthermore, for furture large-scale production, the materials used in the production process should be cheap, available in large quantities and preferably non-toxic, which contrasts with some NC materials that contain toxic elements like Cd or Pb.

1.2.1 Example of a PbS QD Solar Cell

Since the structure and fabrication methods of QD solar cell are quite diverse, we will focus here on a PbS QD solar cell, as they were produced at ETH at the Laboratory for Nanoelectronics [6]. Due to the small band
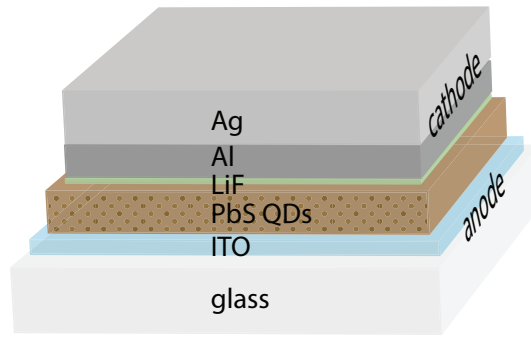
Figure 1.3: Structure of a PbS solar cell.

gap of PbS (0.37eV for bulk PbS), the solar cell is able to convert power from the near infrared spectrum. This is of interest since roughly 50% of the solar energy that reaches the earth is in the infrared spectrum.

Structure

The principle part of the cell is a Schottky junction of PbS and Al, with a thin protection layer (1 nm) of LiF in between. The Schottky barrier is responsible for the electric field which separates the photogenerated carriers. The Al Cathode is covered by a layer of silver, for better air stability. The transparent anode is formed of a layer of indium tin oxide (ITO), on top of the PbS (fig. 1.3).

Fabrication Process

The main steps involved in the fabrication are described below, as they were carried out in reference [6, pp. 13-19].
We start with a glass substrate, which is coated with ITO by lithography. The sample has then to be cleaned (using solvents, and afterwards by O_2 plasma treatment to remove organic residuals).
In the next step, the active layer, i.e. the PbS QDs, is deposited. This can be done by dip coating, where the substrate is immersed in a PbS-hexane solution, and taken out after a few seconds, leaving a thin film of the solution on the sample. Alternatively, spin coating can be used, where several drops of the PbS-Hexane solution are dropped on the substrate, which is then rotated to spread the drops, leading to a film of homogeneous thickness.
Now the long ligands surrounding the QDs have to be exchanged for short ligands, in order to improve inter-particle coupling (and thus e.g. carrier mobility). For this purpose, the substrate is immersed into a suitable compound: Ethandithiol, benzenedithiol (both organic), and ammonium thiocyanate (NH_4SCN, inorganic) can be used.
The sample is then rinsed in acetonitrile and or hexane. In order to obtain a PbS film of desired thickness, the steps of spin (respectively dip) coating, ligand exchange and rinsing have to be repeated several times.
In a last step, the cathode, i.e. the three layers of LiF, Al and Ag are evaporated on the substrate.

Chapter 2

Analysis of PbS QD simulation data

In this chapter we will take a look at the simulation data. The parameters used for the simulation can be found in appendix **??**. We will make a brief analysis of the energy levels and wave function of PbS QDs and draw some conclusions.

Remark. There are a few practical aspects to note concerning the simulation of PbS QDs with OMEN : Since OMEN will consider in its calculation 18 orbitals for the atoms, one must keep in mind that the simulation uses significantly more computing resources than a simulation of, for instance, a CdSe-CdS QD. Especially the memory usage during the calculation can easily exceed 10GB for larger QDs. The size of the generated simulation data for one QD can also amount to more than a few 100MB. It has to be kept in mind that number of atoms scales with the third potence of the radius. Another problem is the high degeneracy of energy levels. This makes it necessarry to simulate a higher number of modes, thus increasing the usage of resources further.

2.1 Energy leves

Remark. In the following discussion, the terms conduction band, band edge and so on are used. Although there are, no energy bands present in a QD, but rather discrete energy levels, these terms nevertheless make sense, since the states above the band gap are similar to conduction band states, and those below to valence band states, respectively. Indeed, if the QD gets bigger, we finally reach the limit where we can treat it as a bulk material. The term *conduction band* thus refers in this context to the discrete energy states above the band gap.

Taking a look at the eigenenergies close to the bandedge, one can see that there are 8 closely spaced modes, for conduction and valence band respectively. Generally, closest to the band edge there is a twofold degenerate energy level, followed by a fourfold degenerate level and then again a twofold degenerate level. (For a few simulations the order is different, for example the fourfold degenerate energy level is closest to the band
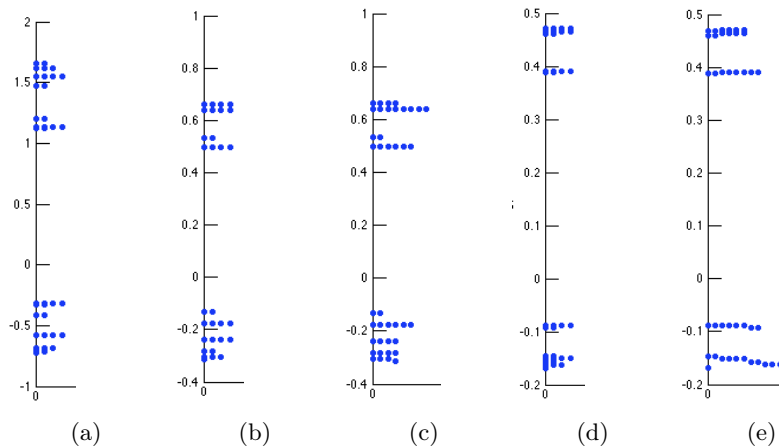

Figure 2.1: Energy levels for different sized QDs: 2nm (a), 6nm (b) and (c), 10nm (d) and (e). Vertical axis in eV. Multiple dots on the horizontal axis signify degeneracy. In plots (c) and (e) energies within 0.03eV are plottet as degenerate, for better visibility.
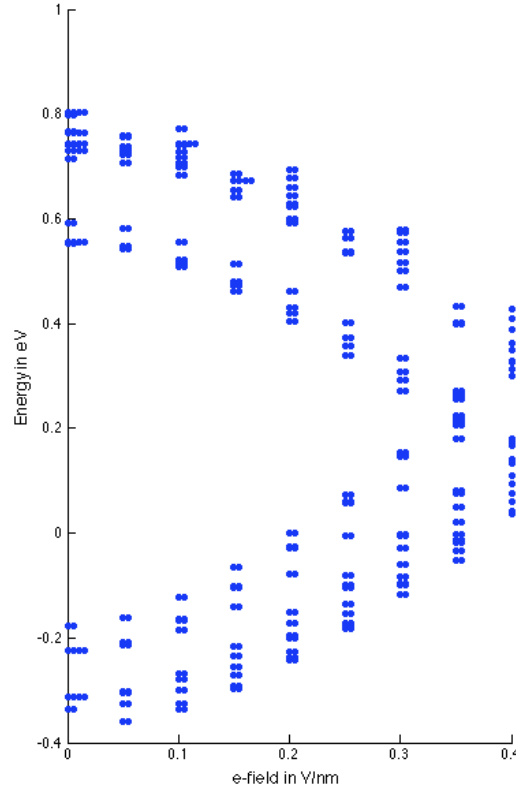
Figure 2.2: Energy states (including degeneracy) for different electric fields.

edge.)

As the size of the QDs increases, the energy levels get closer to each other, resulting in higher degeneracy. This is shown in figure 2.1 (a), (b), (d), where the energy levels (including degeneracy) are shown for 2, 6 and 10nm QDs. Since some levels are very closely spaced, they are not clearly distinguishable form each other. For this reason, the tolerance, within wich an energy level is shown as degenerate, was increased to 0.03 in figure 2.1 (c), (e). An effective eightfold degeneracy for the 10nm QD is now clearly visible.

In the presence of an electric field, this degeneracy is broken, leading to more energy levels, which are all, interestingly, twofold degenerate. This is shown in figure 2.2 for a 5nm QD, where the energy levels are plotted against applied electric field. One can also clearly see how the bandgap gets smaller, as is discussed later on.

2.1.1   Bandgap

2.1.2   Bandgap of QDs in presence of an electric field

2.2   Wave functions

2.2.1   Modes and shape of the wave functions

QDs bigger than 3nm

Remark. Although the term *wave function* appears a lot in this section, the term is rather meant in the sense of *probability density*, since the plots, on which the analysis is based, are more easily interpreted if they show probabilities instead of complex numbers.

Sometimes, the wave functions of QDs are described in analogy to atomic orbitals, labeling them S, P, D and so on, with subscript e or h, denoting electron-like (conduction band) states or hole-like (valence band) states respectively. We will see how well this picture corresponds to the simulation.

The 8 energy states ('modes') closest to the band edge (for conduction and valence band respectively) have wave functions with spherical symmetry, where the highest probability density in the center (fig. 2.3). This is in agreement with the 8 predicted $1S_{e,h}$ orbitals for PbS [3].

The higher energy states show wave functions with more complex shapes. For example, a 8nm QD shows the following wave function shapes: After the 8 spherical conduction band states follow 4 (degenerate) states with barbell shape, in different orientations. Then 2 states with spherical shell shape. After that, 2 states
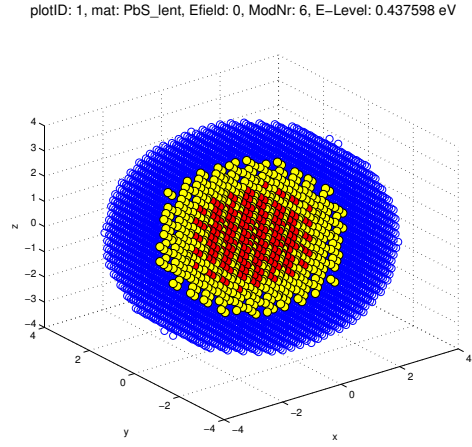
Figure 2.3: Probability density for an energy state in the lower conduction band of an 8nm QD. Locations of very high probability are shown in red, those of medium to high probability in yellow, such that the sum of the probabilities of all yellow and red locations is 50%.
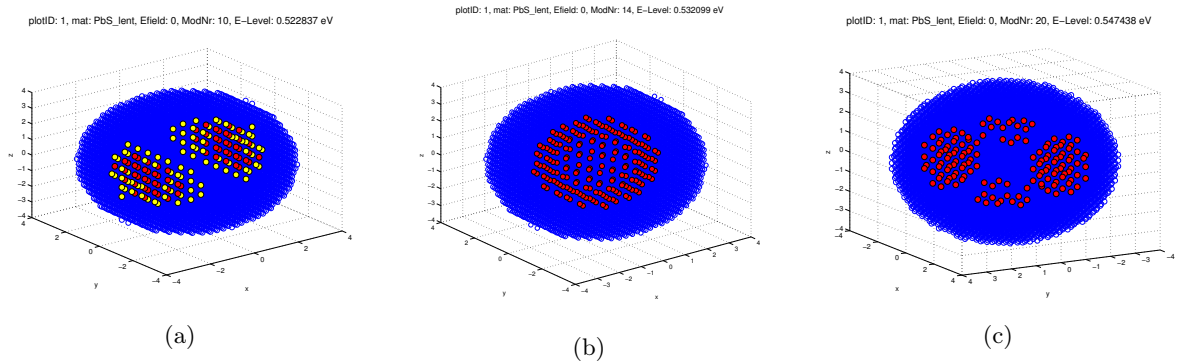

Figure 2.4: Probability density for higher energy states in an 8nm QD: barbell-shape (a), spherical shell-shape (b), crossed barbell-shape (c). In plots (b) and (c) the locations of medium high probability (yellow points) are not shown for clarity.

(a) 3nm QD. 7th valence band state.
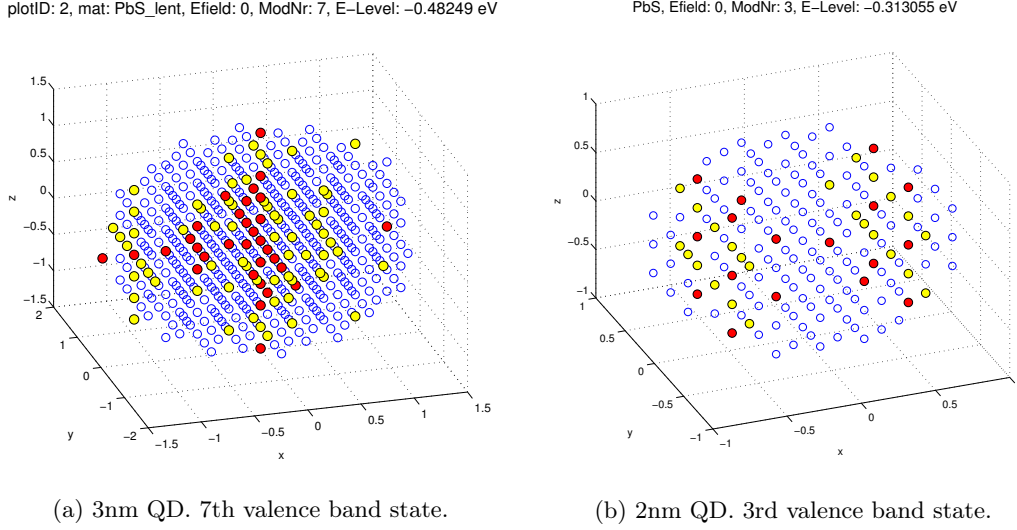

(b) 2nm QD. 3rd valence band state.

Figure 2.5: Probability density for valence band states. The shape starts to deviate from the previously spherical shapes.

with a shape similar to two crossed barbells. Then again 2 spherical shell-like states, followed by 2 ring-shaped wave functions, and so on (fig. 2.4). For the valence band states, the shapes are similar, although they do not occur in the same order (the same is true for QDs of different sizes). This is, however, not consistent with the expected 24 1P orbitals, TODO

QDs smaller than 3nm

For QDs smaller than 3nm, the problem is that it is more difficult to see the shape of the wavefunction (too few atoms). Furthermore it is not clear how well this models a real QD, since the surface (and thus passivation of the surface) beginns to have an even bigger influence.

For a 2 and 3nm QDs, the 8 lowest conduction band states still remain more or less spherical. Whereas the valence band states start loose spherical symetry. For a 3nm QD, mode 7 and 8 are already slightly asymetric (fig. 2.5 (a)), and for the 2nm QD even modes 1 to 4 are clearly not spherical, but the wave function is rather localised at two sides (fig. 2.5 (b)).

2.2.2   Influence of an electirc field

The presence of an electric field results in a shift of the wave function, i.e. the maximum of the probability density is not in the center anymore and the spherical symetry of the 8 first states is broken (for conduction and valence band respectively). Furthermore, one can see that the valence band states really behave differently than conduction band states: The wave function is shifted in the opposite direction (fig. 2.6). This confirms that the valence band states are hole-like, similar to semiconductor band theory.
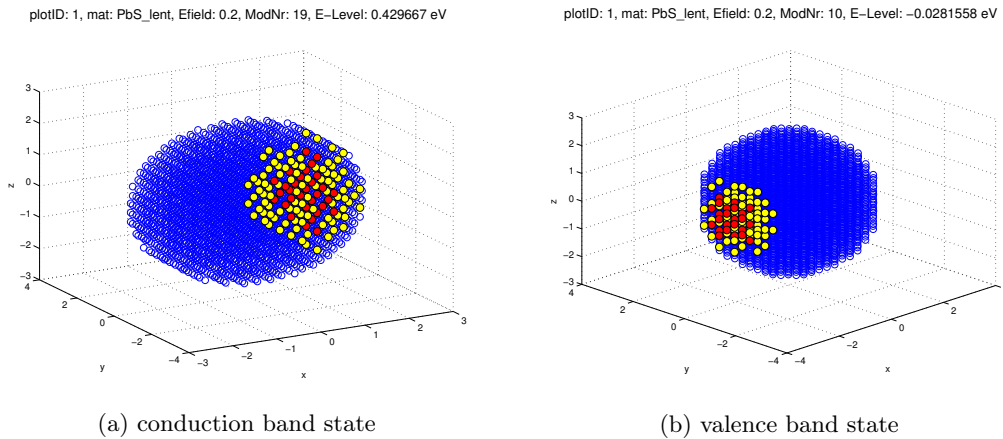

(a) conduction band state

(b) valence band state

Figure 2.6: Probability densities for a 5nm QD in an electic field of 0.2V/nm.

plotID: 1, mat: PbS_lent, Efield: 0.4, ModNr: 24, E–Level: 0.313057 eV

plotID: 1, mat: PbS_lent, Efield: 0.4, ModNr: 26, E–Level: 0.324968 eV

plotID: 1, mat: PbS_lent, Efield: 0.4, ModNr: 28, E–Level: 0.329427 eV

(a)　　　　　　　　　　(b)　　　　　　　　　　(c)

plotID: 1, mat: PbS_lent, Efield: 0.4, ModNr: 30, E–Level: 0.34852 eV

plotID: 1, mat: PbS_lent, Efield: 0.4, ModNr: 32, E–Level: 0.348806 eV

plotID: 1, mat: PbS_lent, Efield: 0.4, ModNr: 34, E–Level: 0.362633 eV

(d)　　　　　　　　　　(e)　　　　　　　　　　(f)
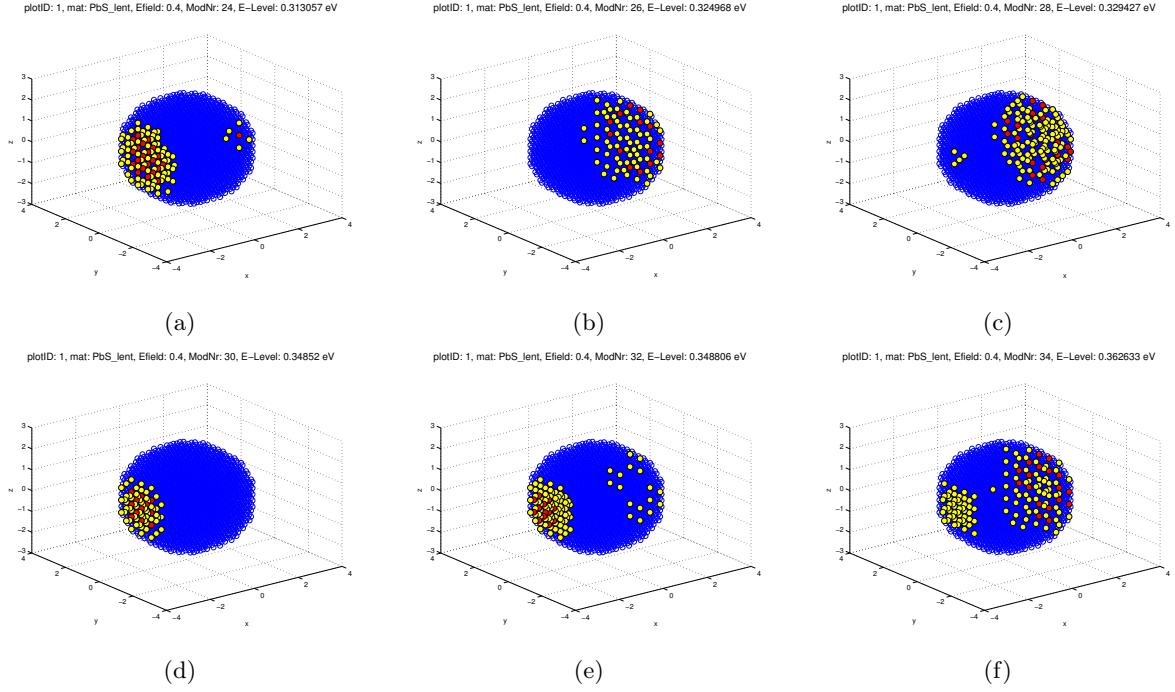
Figure 2.7: Probability density in the presence of a high electric field (0.4V/nm). Increasing energy from (a) to (f). Hole-like states (a),(d),(e) become mixed with electron-like states (b),(c),(f).

For larger electric fields the bandgap gets smaller until it finally disappears, i.e. conduction and valence band states are not separated anymore. For example, for a 5nm QD, the band gap disappers for electirc fields larger than 0.35 V/nm (fig. 2.2). The states which are close to the (former) band gap cannot be sepearted in conduction and valence band: as energy increases, some hole-like states (below 0.313 eV) are followed by electron-like (i.e. conduction band-like) states (0.325...0.329 eV) which then again are followed by hole-like states (around 0.348 eV) and later again electron-like states (fig. 2.7).

Additionally, with higher electric fields, the wave functions seem not only to be shifted, but change shape: Some now look more like asymetric barbells or even stranger, ring-like shapes (fig. 2.7 (a) and (f)).

2.3 Conclusions

Chapter 3

Simulations with TOM

TOM was programmed to simulate spherical nanocrystals and to visualize the results of the simulations. Though we focused on spherical structures, the code is hold abstract and slim to make extending 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 with different parameters users do not want to spend a lot of time on writing command files for OMEN by themselves and start 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 3.2.1). 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 3.2.2)

3. Visualization of simulation data
   All different kinds of plots are available within the toolbox, such as visualizing band gaps, wave functions or quantum dot structures (see section 3.2.3).

3.1   Installing the Software

```
**********************
****** TO DO !!! ******
**********************
```

3.2   Using TOM

3.2.1   Running a simulation

Type gui_simulate in the MATLAB command window and hit enter. The window as in figure 3.1 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.

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.

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

3. Radius
   The radius has to be entered in a specific way. The syntax is:

Figure 3.1: The gui_simualte window

```
[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. Simulation a single spherical PbS quantum dot with radius 3.5 nm you would enter: [3.5,3.5,1]

Example. For a spherical CdS-CdSe quantum dot the cell would look like this: [1,4,4][5,6,2] Looking at the radius parameters, TOM will do all the 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

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

5. Voltage
   The Voltage sweep is entered in the same way as the radius. You find more information under the following remark.

Remark (Vectors & Matrices). As the MATLAB GUIs do not accept vectors or matrices as it is known from the command window, the parameters have to be entered as a string and are converted to matrices later on.

There are different ways how 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 column *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.

3.2.2 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. (futher information section 3.3.1). All operations are done using these objects. If you would like to display only certain simulation data, you would need an *qdotObj* array. How to create such an array please read section 3.2.4.

Displaying the whole database can be done by typing `gui_db` into the MATLAB command window. A window as in figure 3.2 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 a set of simulation data, you can also call `gui_db(qdotObj_ARRAY)`, where `qdotObj_ARRAY` is an array of qdotQbj objects.

Figure 3.2: The gui_database window

Within the GUI you can sort the simulations with the column header, select simulations and let plot them, open the directory of a simulation or even export a selection, which is available as an qdotObj array in the main workspace entitled *ExportedDB*.

3.2.3 Visualization

As explained in the previous section, simulation data can be visualized through the *gui_ database* window, but also manually. In this section the plotting functions are explained.

```
plotBandGap(DB)
```

3.2.4 Additional tools

3.3 Maintaining TOM

3.3.1 General Structure

```
/root
 ├─ MATLAB
 │   ├─ Classes
 │   ├─ Functions
 │   ├─ GUI
 │   └─ System
 ├─ OMEN executable
 └─ Simulations
     ├─ ID*
     ├─ :
     └─ log
```

```
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_E_0_0.dat
```

Figure 3.3: The TOM structure by default   Figure 3.4: Structure of a simulation set

```
1          IDyyyymmdd–hhmm–ssfff_MATNAME
```

```
1        struct global config {
           config.root         = '';
3          config.system       = '';
           config.log          = '';
5          config.user         = '';
           config.machine      = '';
7          config.OMEN         = '';
           config.simulations  = '';
9          config.vOMEN        = '';
           config.cancelSim    = '';
11       }
```

Chapter 4

MATLAB

4.1 Install

4.2 User

4.3 Maintainer

4.3.1 Basic concepts

Parameters for the simulation are stored and passed as arguments to functions as objects of the class *Qdot*, further refered to as QDO. The class provides properties for all parameters neccessarry 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;
```

4.3.2 GUI

4.3.3 Simulation

The simulation parameters from the GUI are passed to the functions 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 e-field) are at this stage not scalars, but a 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 QDOG. Supported sweep parameters are 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 data saved in seperate folders, called ID*timestamp_ material*:

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

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

5. 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.

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.

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.

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

4.3.4   Database

In order to know which parameter sets already have been simulated, there are some useful tools, which can be found in the folder *Functions/DataUtils*. The basic principle is to get all 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 QDOA = getQDOA()

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

Deleting simulation data

function deleteSimData(QDOA)

Besides the possibility of deleting simulation data manually, an easy and fast way is using the function *deleteSimData.m*. The simulation folders for all elements in QDOA will be deleted permanently. Combined with a filter function, it thus becomes very easy to delete the desired simulation data.

Remove duplicates

function [cleaned, duplicates] = removeDuplicates(QDOA)

It is well possible that, after performing many simulations, one has multiple simulations of exactly the same parameters. In this case one may want to delete the duplicates, which can be done by the above function. The function returns a QDOA with the duplicates removed, as well as a QDOA with only the duplicates. Note that this function does not delete the simulation data of the duplicates from the disk. For this purpose one can use the function *deleteSimData.m* subsequently.

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

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

4.3.5   Plotting

Here follows a short description of functions which can be used for basic visualisation of the data obtained by the OMEN simulation. For a more detailed description, please refer to the code. *Note:* 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 Qdot.path property.

Plotting the eigenenergies

function plotEvsField(QDOA)

Plot the energy levels for every QDO in QDOA against the applied electric field. Degeneracy of energy levels is also shown.

Plotting the probability density

These functions all take an array of *Qdot* objects (QDOA) as an input. Additionally, the number of eigenmodes to be displayed has to be specified, 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, NMod)

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, NMod)

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 atoms is smaller than a probability value specified in *probLim*. 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 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 quantum dots. 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.

Appendix A

Q & A

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

Why does the simulation not stop, when I hit *Cancel Simulation*?   MATLAB is blocked during the time it has send a command to the shell. You might have to abort the process by pushing `CTRL+C` a couple of times.

Bibliography

[1] John D. Dow Robert S. Allgaier Otto F. Sankey Craig S. Lent, Marshall A. Bowen and Eliza S. Ho. Relativistic empirical tight-binding theory of the energy bands of gete, snte, pbte, pbse, pbs, and their alloys. *Superlattices and Microstructures*, 2(5):491–499, 1986.

[2] Olesya Yarema Deniz Bozyigit, Michael Jakob and Vanessa Wood. Deep level transient spectroscopy (dlts) on colloidal-synthesized nanocrystal solids. November 2012.

[3] Maksym V. Kovalenko Dmitri V. Talapin, Jong-Soo Lee and Elena V. Shevchenko. Prospects of colloidal nanocrystals for electronic and optoelectronic applications. *Chemical Reviews*, 110(1):389–485, January 2010.

[4] Al. L. Efros and M. Rosen. The electronic structure of semiconductor nanocrystals. *Annual Review of Materials Science*, 30:475–521, 2000.

[5] Alexander H. Ip et. al. Hybrid passivated colloidal quantum dot solids. *Nature nanotechnology*, 7(9):577–582, September 2012.

[6] Micheal Jakob. Deep level transient spectroscopy on solar cells based on colloidal nanocrystals. Ms thesis, ETH Zurich.

[7] Edward H. Sargent Jiang Tang. Infrared colloidal quantum dots for photovoltaics: Fundamental and recent progress. *Advanced Materials*, 23(1):12–29, January 2011.

[8] Victor I. Klimov. *Nanocrystal Quantum Dots*. CRC Press, Second Edition edition, 2010.

[9] Gianluca Calestani Andrea Migliori Antonietta Guagliardi Ludovico Cademartiri, Erica Montanari and Geoffrey A. Ozin. Size-dependent extinction coefficients of pbs quantum dots. *Journal of the American Chemical Society*, 128(31):10337–10346, August 2006.

[10] Mathieu Luisier. *OMEN Manual*. ETH Zürich, Integrated Systems Laboratory, October 2012.

[11] Gregory D. Scholes Margaret A. Hines. Colloidal pbs nanocrystals with size-turnable near-infrared emission: Observation of post-synthesis self-narrowing of the particle size distribution. *Advanced Materials*, 15(21):1844–1849, November 2003.

List of Tables

Declaration of Originality

We hereby declare that the written work we have submitted entitled

Simulation of Quantum Dots

is original work which we alone have authored and which is written in our own words.

Authors

LAST NAME	FIRST NAME
Dittberner	Matthias
Funck	Christian

Supervisors

LAST NAME	FIRST NAME	DEGREE
Luisier	Mathieu	Professor
Wood	Vanessa	Professor

With the signature we declare that we have been informed regarding normal academic citation rules and that we have read and understood the information on *Citation etiquette* (http://www.ethz.ch/students/exams/plagiarism_s_en.pdf). The citation conventions usual to the discipline in question here have been respected. The above written work may be tested electronically for plagiarism. [1]

Place and date	Christian Funck		Place and date	Matthias Dittberner

---

[1] Based on the official Declaration of ETH Zurich: http://www.ethz.ch/faculty/exams/plagiarism/confirmation_en.pdf