

---

# nextnano

Software for semiconductor nanodevices

## **nextnano<sup>3</sup> Documentation**

***Release 2.7.2***

**nextnano GmbH**

**Jan 06, 2026**



## GETTING STARTED

|           |                          |            |
|-----------|--------------------------|------------|
| <b>1</b>  | <b>Overview</b>          | <b>3</b>   |
| <b>2</b>  | <b>Models</b>            | <b>5</b>   |
| <b>3</b>  | <b>Tutorials</b>         | <b>43</b>  |
| <b>4</b>  | <b>Material Database</b> | <b>315</b> |
| <b>5</b>  | <b>Input Syntax</b>      | <b>369</b> |
| <b>6</b>  | <b>Command Line</b>      | <b>487</b> |
| <b>7</b>  | <b>Release Notes</b>     | <b>491</b> |
| <b>8</b>  | <b>FAQs</b>              | <b>495</b> |
| <b>9</b>  | <b>Books</b>             | <b>501</b> |
| <b>10</b> | <b>Journal Papers</b>    | <b>503</b> |
| <b>11</b> | <b>Theses</b>            | <b>509</b> |





The *nextnano*<sup>3</sup> tool is the predecessor of *nextnano++*.



**OVERVIEW***Features*

- includes group IV materials (Si, Ge, SiGe), all III-V materials, II-VI materials and its ternaries as well as lattice-matched quaternaries. The nitride and II-VI materials are available in the zinc blende and wurtzite crystal structure.
- flexible structures and geometries (1D, 2D and 3D)
- quantum mechanics based on single-band and multi-band  $\mathbf{k} \cdot \mathbf{p}$  model within a finite differences grid
- includes strain, piezo and pyroelectric charges
- growth directions along [001], [011], [111], [211], ... in short along any crystallographic direction
- equilibrium and non-equilibrium, calculation of current close to equilibrium (semi-classical), ballistic transport
- magnetic field

Not all keywords have been documented in the new documentation.

The old documentation for *nextnano*<sup>3</sup> is available here: <https://www.nextnano.com/nextnano3/>



Some of models implemented in *nextnano++*.

## 2.1 Crystal coordinate systems

For zinc-blende materials there are three-digit Miller indices. The Miller indices define a **plane**. There exists a vector that is perpendicular to this plane, e.g. in zinc blende materials, the  $[hkl]$  **vector** is always perpendicular to the  $(hkl)$  **plane**. However, for wurtzite, this is not necessarily true. For instance, although the  $[0001]$  vector is perpendicular to the  $(0001)$  plane, in general it does not hold that the vector that is perpendicular to the  $(hkil)$  plane is defined by  $[hkil]$ . Note: For a 1D simulation, the heterostructure is always grown along the x axis. For a 2D simulation, always the (x,y) plane is used.

### Zinc blende

```
crystal_zb{
  x_hkl = [1, 0, 0]    # Specify (hkl) plane perpendicular to x axis
  y_hkl = [0, 1, 0]    # Specify (hkl) plane perpendicular to y axis
```

The x axis of the simulation coordinate system is perpendicular to this  $(hkl)$  plane of the crystal, here:  $(1\ 0\ 0)$ . The y axis of the simulation coordinate system is perpendicular to this  $(hkl)$  plane of the crystal, here:  $(0\ 1\ 0)$ . The Miller indices (here:  $(0\ 0\ 1)$ ) for the z axis are determined automatically. For zinc blende it holds: The vector  $[hkl]$  is perpendicular to the  $(hkl)$  plane.

Another example:

```
crystal_zb{
  x_hkl = [3, 1, 1]    #
  y_hkl = [0, -1, 1]   #
```

x axis of simulation coordinate system is perpendicular to  $(3\ 1\ 1)$  plane of crystal coordinate system, i.e. the x axis is along  $[311]$  direction. y axis of simulation coordinate system is perpendicular to  $(0\ -1\ 1)$  plane of crystal coordinate system, i.e. the y axis is along  $[0-11]$  direction. The Miller indices (here:  $[2, -3, -3]$ ) for the z axis are determined automatically, i.e.  $(2\ -3\ -3)$  plane, i.e. the z axis is along  $[311]$  direction.

### Wurtzite

Usually for wurtzite, the four-digit Miller-Bravais indices  $(h\ k\ i\ l)$  are used. We also use this notation but omit the 'i' because  $i = -h - k$ . The three integer values (Miller indices) that are given for  $x\_hkl$  refer to a plane and not to a direction. The x direction is then the one that is perpendicular to this plane.

This vector along the x axis has indices that are in general not identical to the Miller indices in wurtzite.

```
crystal_wz{
  x_hkl = [ 0, 0, 1]    # e.g. hexagonal [0001] axis along x axis
                    # Specify (hkil) plane perpendicular to x axis: (↪
↪0, 0, 0, 1)
  y_hkl = [ 1, 0, 0]    # Specify (hkil) plane perpendicular to y axis: (↪
↪1, 0, -1, 0)
```

This corresponds to the four-digit Miller-Bravais indices  $hkil = (0, 0, 0, 1)$  that define the  $(hkil)=(0001)$  plane. Coincidentally, the vector  $[0001]$  is perpendicular to it. This corresponds to the four-digit Miller-Bravais indices  $hkil = (1, 0, -1, 0)$  that define the  $(hkil)=(10-10)$  plane. The Miller-Bravais indices for the  $(hkil)$  plane perpendicular to the  $z$  axis are determined automatically inside the code (here:  $(-1\ 2\ -1\ 0)$ ).

Another example:

```
crystal_wz{
    # e.g. (10-10) plane is perpendicular to x axis
    x_hkl = [ 1, 0, 0]    # hkil = ( 1, 0, -1, 0)
    y_hkl = [-1, 2, 0]    # hkil = (-1, 2, -1, 0)
```

This corresponds to the four-digit Miller-Bravais indices  $hkil = (1, 0, -1, 0)$  that define the  $(hkil)=(10-10)$  plane. The  $x$  axis of the simulation coordinate system is perpendicular to this plane. This corresponds to the four-digit Miller-Bravais indices  $hkil = (-1, 2, -1, 0)$  that define the  $(hkil)=(-12-10)$  plane. The  $y$  axis of the simulation coordinate system is perpendicular to this plane. The Miller-Bravais indices of the  $(hkil)$  plane perpendicular to the  $z$  axis are determined automatically inside the code (here:  $(0001)$ ). Coincidentally, the vector  $[0001]$  is perpendicular to it. In this particular case, no rotation has to be applied to the crystal (rotation matrix = identity matrix).

```
crystal_wz{
    ...
    rotation_c_a_ratio_use_substrate = yes # (default: yes)
    rotation_c_a_ratio = 1.63299          # c/a ratio
```

In wurtzite, the  $c/a$  ratio of the two lattice constants ‘ $c$ ’ and ‘ $a$ ’ is important. The ideal one,  $c/a = \sqrt{8/3} = 1.63299\dots$ , is not the one present in GaN, AlN or InN, i.e. in real materials. For the rotation of the crystal coordinate system to the simulation coordinate system, a specific  $c/a$  ratio has to be assumed. By default, we use the one of the substrate material. If you want to use the ideal  $c/a$  ratio, you have to specify `rotation_c_a_ratio_use_substrate = no`.

Additionally, one can specify a custom value for the  $c/a$  ratio. If no customized value is specified, `rotation_c_a_ratio = sqrt(8/3)` (default). The actually used rotation matrix is written to the log file.

```
x_hkl = [ ., ., .]
y_hkl = [ ., ., .]
z_hkl = [ ., ., .]
```

Exactly two of these three axes have to be specified, the third one is calculated internally.

## 2.2 Introduction to strain calculation

Here we introduce the theoretical background of the strain and stress calculation in *nextnano++*. At first we will describe the definition of a strain tensor  $\varepsilon$  and stress tensor  $\sigma$  and then describe the basis of strain tensor calculation in *nextnano++*. A strain tensor is used to calculate the shifts and splittings of band-edge energies and piezoelectric charges.

### Table of contents

- *Strain tensor  $\varepsilon$*
- *Stress tensor  $\sigma$*
- *Strain and stress calculation*
  - *In general*
  - *In nextnano++*

### 2.2.1 Strain tensor $\varepsilon$

The calculation of strain effects in *nextnano++* is based on linear continuum elasticity theory, in which a crystal can be described by a field of material points with coordinates  $\mathbf{x}$ . A distortion of the crystal shifts any point to a new position  $\mathbf{x}' = \mathbf{x}'(\mathbf{x})$ . A field of displacement vectors  $\mathbf{u}$  is defined as the deviation between the new position and the original position:

$$\mathbf{u}(\mathbf{x}) := \mathbf{x}'(\mathbf{x}) - \mathbf{x}$$

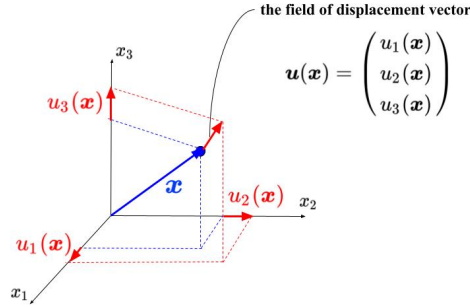


Figure 2.2.1.1: The field of displacement vector  $\mathbf{u}$  at  $\mathbf{x}$ . This is the vector along which the point that was at the position  $\mathbf{x}$  moved through the displacement.

A strain tensor  $\varepsilon$  is defined using this displacement vector:

$$\varepsilon_{ij} := \frac{1}{2} \left[ \frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i} \right]; \quad (i, j = 1, 2, 3)$$

Strain is dimensionless. The diagonal elements of this strain tensor  $\varepsilon_{ii}$  represents the length changes per unit length in  $x_i$ -direction as described in Figure 2.2.1.2.

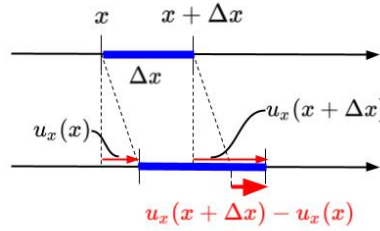


Figure 2.2.1.2: Deformation of a dilatible string in an unstrained (top) and strained state (bottom). We can see the diagonal element  $\varepsilon_{ii} = \frac{\partial u_i}{\partial x_i}$  represents the length changes per unit length in  $x_i$ -direction.

The off-diagonal elements  $\varepsilon_{ij}(i \neq j)$  arise due to shear deformations of the crystal. Figure 2.2.1.3 shows the deformation of an infinitesimal rectangle in  $x_1x_2$  plane. We can see  $\frac{\partial u_2}{\partial x_1} = \frac{u_2(x_1 + \Delta x_1, x_2) - u_2(x_1, x_2)}{\Delta x_1} = \sin \alpha \simeq \alpha$  and  $\frac{\partial u_1}{\partial x_2} = \frac{u_1(x_1, x_2 + \Delta x_2) - u_1(x_1, x_2)}{\Delta x_2} = \sin \beta \simeq \beta$ . In these angle changes,  $\frac{\alpha + \beta}{2}$  corresponds to a pure solid-body rotation and  $\frac{\alpha - \beta}{2} = \frac{1}{2} \left[ \frac{\partial u_2}{\partial x_1} + \frac{\partial u_1}{\partial x_2} \right] = \varepsilon_{12}$  measures the shear strain.

By definition strain tensor  $\varepsilon$  is symmetric (i.e.  $\varepsilon_{ij} = \varepsilon_{ji}$ ) so the number of components that must be specified is actually 6. Voigt notation is the useful convention in which these 6 independent components are written in form of a  $6 \times 1$  matrix for short. This notation reads:

$$11 \rightarrow 1, 22 \rightarrow 2, 33 \rightarrow 3, 23 \rightarrow 4, 31 \rightarrow 5, 12 \rightarrow 6$$

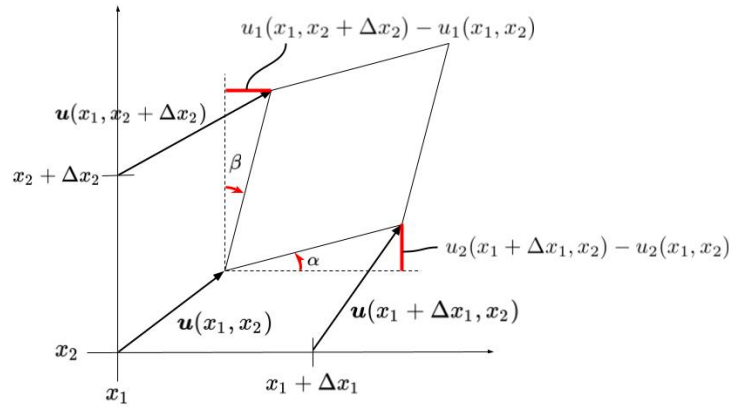


Figure 2.2.1.3: Deformation of an infinitesimal rectangle in a strained state.

and

$$\begin{bmatrix} \varepsilon_1 \\ \varepsilon_2 \\ \varepsilon_3 \\ \varepsilon_4 \\ \varepsilon_5 \\ \varepsilon_6 \end{bmatrix} = \begin{bmatrix} \varepsilon_{11} \\ \varepsilon_{22} \\ \varepsilon_{33} \\ 2\varepsilon_{23} \\ 2\varepsilon_{13} \\ 2\varepsilon_{12} \end{bmatrix}$$

## 2.2.2 Stress tensor $\sigma$

A stress tensor component  $\sigma_{ij}$  represents the force towards  $x_j$ -direction acting on infinitesimal area that is perpendicular to  $x_i$ -direction. Its unit is the same with pressure ( $[\text{Pa}] = [\text{N}/\text{m}^2]$ ).

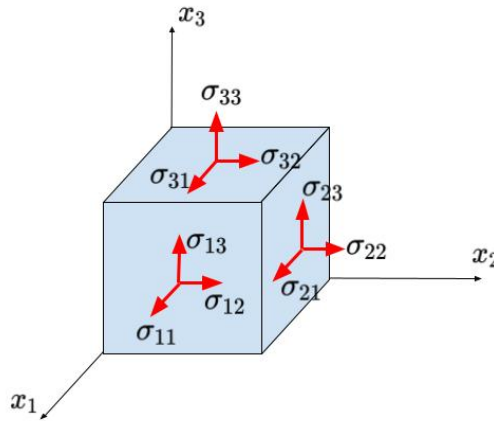


Figure 2.2.2.1: The components of stress tensor  $\sigma$ .

In linear approximation, this stress tensor is related to the strain tensor  $\varepsilon$  by means of Hook's law:

$$\sigma_{ij} = \sum_{kl} C_{ijkl} \varepsilon_{kl}$$

where  $C_{ijkl}$  is the component of elasticity stiffness tensor, which is the forth-order tensor comprising  $3^4 = 81$  components. It's dimension is the same with stress tensor components and defined as  $[\text{GPa}]$  in *nextnano++*. In



Voigt notation,  $C$  is the form of a  $6 \times 6$  matrix by putting  $C_{ijkl} = C_{mn}$  ( $i, j, k, l = 1, 2, 3, m, n = 1, \dots, 6$ ). Then the Hook's law reads

$$\begin{bmatrix} \sigma_1 \\ \sigma_2 \\ \sigma_3 \\ \sigma_4 \\ \sigma_5 \\ \sigma_6 \end{bmatrix} = \begin{bmatrix} C_{11} & C_{12} & C_{13} & C_{14} & C_{15} & C_{16} \\ C_{21} & C_{22} & C_{23} & C_{24} & C_{25} & C_{26} \\ C_{31} & C_{32} & C_{33} & C_{34} & C_{35} & C_{36} \\ C_{41} & C_{42} & C_{43} & C_{44} & C_{45} & C_{46} \\ C_{51} & C_{52} & C_{53} & C_{54} & C_{55} & C_{56} \\ C_{61} & C_{62} & C_{63} & C_{64} & C_{65} & C_{66} \end{bmatrix} \begin{bmatrix} \varepsilon_1 \\ \varepsilon_2 \\ \varepsilon_3 \\ \varepsilon_4 \\ \varepsilon_5 \\ \varepsilon_6 \end{bmatrix}$$

For many crystal structures with high symmetry, many of these coefficients are 0 and some are related to others. The elasticity tensor of zincblende and wurtzite crystals are given by

$$C_{zb} = \begin{bmatrix} C_{11} & C_{12} & C_{12} & & & \\ C_{12} & C_{11} & C_{12} & & & \\ C_{12} & C_{12} & C_{11} & & & \\ & & & C_{44} & & \\ & & & & C_{44} & \\ & & & & & C_{44} \end{bmatrix}$$

$$C_{wz} = \begin{bmatrix} C_{11} & C_{12} & C_{13} & & & \\ C_{12} & C_{11} & C_{13} & & & \\ C_{13} & C_{13} & C_{33} & & & \\ & & & C_{44} & & \\ & & & & C_{44} & \\ & & & & & C_{66} \end{bmatrix}$$

with  $C_{66} = \frac{1}{2}[C_{11} - C_{22}]$  in wurtzite.

These constants are defined in *database\_nnp.in*. You can also overwrite these values in your input file.

- For zinc-blend materials, for example:

```
database{
  binary_zb{
    name = GaAs
    valence = III_V

    elastic_consts{
      c11 = 122.1                # [GPa] elastic constants
      c12 = 56.6                # 1 * 1011 dyn/cm2 = 10 GPa ->
      ↪ 12.21 * 1011 dyn/cm2 = 122.1 GPa
      c44 = 60.0                # The elastic constants are
      ↪ needed for the calculation of the strain in heterostructures.
    }
  }
}
```

- For wurtzite materials, for example:

```
database{
  binary_zb{
    name = GaN
    valence = III_V

    elastic_consts{
      c11 = 390                # [GPa] elastic constants
      c12 = 145                # 1 * 1011 dyn/cm2 = 10 GPa ->
      ↪ 39.0 * 1011 dyn/cm2 = 390 GPa
      c13 = 106                #
    }
  }
}
```

(continues on next page)

(continued from previous page)

```

        c33 = 398                                #
        c44 = 105                                # The elastic constants are
↪needed for the calculation of the strain in heterostructures.
    }
}

```

### 2.2.3 Strain and stress calculation

Next we will describe how the strain tensor  $\varepsilon$  and stress tensor  $\sigma$  are determined in general. Then the two types of calculation implemented in *nextnano++* are introduced briefly.

#### In general

The principle of conservation of linear momentum results in the following equations of stress tensor components for  $i = 1, 2, 3$ :

$$\sum_{j=1}^3 \frac{\partial \sigma_{ji}}{\partial x_j} + f_i = 0$$

where  $\mathbf{f}$  is the body force such as gravity. When the boundary conditions are specified, the field of displacement vector  $\mathbf{u}$ , by which the stress tensor components  $\sigma_{ij}$  are eventually written, is determined according to these simultaneous differential equations. Then the strain tensor  $\varepsilon$  and stress tensor  $\sigma$  are also determined from  $\mathbf{u}$ .

#### **i** Note

The principle of conservation of angular momentum, on the other hand, results in the symmetricity of stress tensor:  $\sigma_{ij} = \sigma_{ji}$

The field of displacement vector which satisfies the above balance equations and boundary conditions also minimizes the total potential energy  $U + V_E$  where  $U$  is the elastic strain energy and  $V_E$  is the potential energy associated with the body force  $\mathbf{f}$ . This is so called minimum total potential energy principle.

In the linear approximation regime, the elastic energy stored in the whole body is:

$$U = \frac{1}{2} \int_V C_{ijkl} \varepsilon_{ij} \varepsilon_{kl} dV$$

When the body force  $\mathbf{f}$  is assumed to be zero throughout the system, solving the above differential equations is equivalent to find the strain tensor that minimizes this elastic energy  $U$ .

#### In nextnano++

There are two kinds of calculation of strain, `pseudomorphic_strain{ }` and `minimized_strain{ }`, in *nextnano++*. In both of implementations pseudomorphic layer is assumed as the boundary condition between the substrate and the layer grown on this substrate. The substrate is assumed to be so thick that the in-plane lattice constants of the layer is matched to that of substrate. Also, the body force  $\mathbf{f}$  is assumed to be 0 throughout the structure.

In this assumption, the analytic expressions for strain tensor that satisfies the aforementioned stress balance equations (i.e. that minimizes the elastic energy) can be found for 1D structures. This analytic solution is implemented on `pseudomorphic_strain{ }`. This feature also works in 2D or 3D but the user must be sure that the model makes sense from a physical point of view (i.e. the 2D/3D structure should consist of different layers along the growth direction whereas the layers must be homogenous along the two perpendicular directions).

On the other hand, `minimized_strain{ }` calculates the strain tensor by minimizing the elastic energy mentioned before. This can also be used for 1D simulations. In this case, the results will be equivalent to the analytical model `pseudomorphic_strain{ }`.

Please refer to [\[AndlauerPhD2009\]](#) for more details about these topics.

*Last update: nnnn/nn/nn*

## 2.3 Piezoelectricity in wurtzite

The *nextnano++* and *nextnano<sup>3</sup>* tools can simulate growth orientation dependence of the piezoelectric effect in heterostructures. Following A.E. Romanov et al., Journal of Applied Physics 100, 023522 (2006), we consider  $\text{In}_x\text{Ga}_{1-x}\text{N}$  and  $\text{Al}_x\text{Ga}_{1-x}\text{N}$  thin layers pseudomorphically grown on GaN substrates. The c-axis of the substrate GaN is inclined by an angle  $\theta$  with respect to the interface of the heterostructure.

The layer is assumed to be very thin compared to substrate so that the strain is approximately homogeneous in all direction (pseudomorphic), and the ternary alloys mimic the orientation of crystallography direction. The layer material deforms such that the lattice translation vector of each layer has a common projection onto the interface.

The strain in a crystal induces piezoelectric polarization, which contributes as an additional component to the total charge density profile. The important consequence of their analysis is that the piezoelectric polarization normal to the interface becomes zero at a nontrivial angle. The piezoelectric charge in a heterostructure in general results in an additional offset between electron and hole spatial probability distribution, thereby reducing the overlap of their wave functions in real space. The small overlap of electron and hole leads to an inefficient radiative recombination, i.e. lower efficiency of optoelectronic devices. The work by Romanov et al. paved the way to device optimization by the growth direction of the crystal.

An introduction for the strain calculation is described here: [Introduction to strain calculation](#)

Table of contents

- [Specify crystal orientation](#)
- [Parameter sweep of the angle using Template: Sweep over the variable theta](#)
- [Strain](#)
- [Piezoelectric effect \(first-order\)](#)
- [Post-Processing for polarization](#)
- [Alloy content dependence](#)
- [AlGaIn](#)
- [Piezoelectric effect \(second-order\)](#)

### References

- A.E. Romanov, T.J. Baker, S. Nakamura, and J.S. Speck, Journal of Applied Physics **100**, 023522 (2006)
- S. Schulz and O. Marquardt, Phys. Rev. Appl. **3**, 064020 (2015)
- S.K. Patra and S. Schulz, Phys. Rev. B **96**, 155307 (2017)

The corresponding input files are located in the *nextnano++* sample files folder:

- [Romanov\\_InGaIn\\_theta\\_nnp.in](#)
- [Romanov\\_AlGaIn\\_theta\\_nnp.in](#)
- [Romanov\\_InGaIn\\_theta\\_nnp\\_2nd.in](#)
- [Romanov\\_InGaIn\\_theta\\_nn3.in](#)
- [Romanov\\_InGaIn\\_theta\\_nn3\\_2nd.in](#)

### 2.3.1 Specify crystal orientation

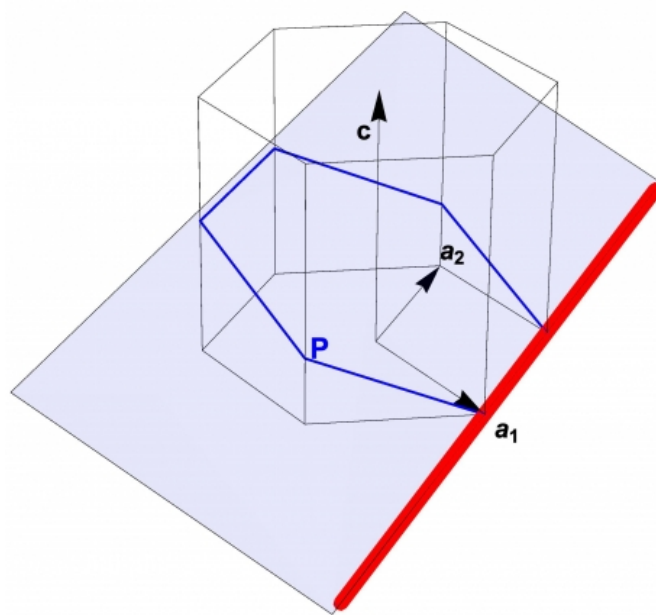


Figure 2.3.1.1: Rotation of a wurtzite structure. The blue plane is parallel to the interface.

The `nextnano` software treats the rotation of crystal orientation by the **Miller-Bravais indices** in the input file. The setup of our system is as follows: the x-axis of the simulation coordinate system (hereafter  $\mathbf{x}'$ -axis) is taken to the normal vector of the interface. The z-axis of the simulation system ( $\mathbf{z}'$ ) is normal to the  $(-1\ 2\ -1\ 0)$  plane of the crystal, i.e. it is along  $\mathbf{a}_2$  direction in Figure 2.3.1.1. The rotation axis indicated with red line is along  $\mathbf{z}'$ -axis, and the interface is shown as the blue plane. The inclination angle  $\theta$  is defined as the angle between the c-axis  $[0001]$  and the normal vector of the blue plane, which is  $\mathbf{x}'$ -axis.

Then the crystal orientation is specified in `nextnano++` input file as

```
crystal_wz{
  x_hkl = [ 1, 0, l(theta)] # x axis perpendicular to (hkl) plane = (hkil) plane
  z_hkl = [-1, 2, 0]        # z axis perpendicular to (hkl) plane = (hkil) plane
}
```

where  $l(\theta)$  is an integer determined by the inclination angle. This statement means *the  $\mathbf{x}'$ -axis is normal to the  $(1\ 0\ -1\ l(\theta))$  plane of the crystal, whereas  $\mathbf{z}'$ -axis is normal to the  $(-1\ 2\ -1\ 0)$  plane.* (Note that `nextnano++` does not require the third entry, i.e. the letter i, in Miller-Bravais notation  $(hkil)$  because  $i = -(h+k)$ .)

The index  $l(\theta)$  is deduced from a simple geometry consideration. Figure 2.3.1.2 shows the cross-section of a wurtzite lattice that is perpendicular to the rotation axis in Figure 2.3.1.1.

- When  $\theta = 0$ , the interface is normal to the  $(0001)$  plane, i.e.  $\mathbf{x}'$ -axis is normal to the  $(0001)$  plane.
- When  $\theta = 90$  degree, the  $\mathbf{x}'$ -axis should be normal to the  $(1\ 0\ -1\ 0)$  plane of the crystal.
- When  $0 < \theta < 90$  degree, definition of the index is  $l(\theta) := \frac{c}{d}$  and the following relation holds

$$d = \frac{\sqrt{3}}{2} a \tan \theta.$$

From these equations we find

$$l(\theta) = \frac{2c}{\sqrt{3}a \tan \theta}.$$

The plane to be determined can be then taken as

$$(hkil) = (\sin \theta\ 0\ -\sin \theta\ \frac{2c}{\sqrt{3}a} \cos \theta)$$

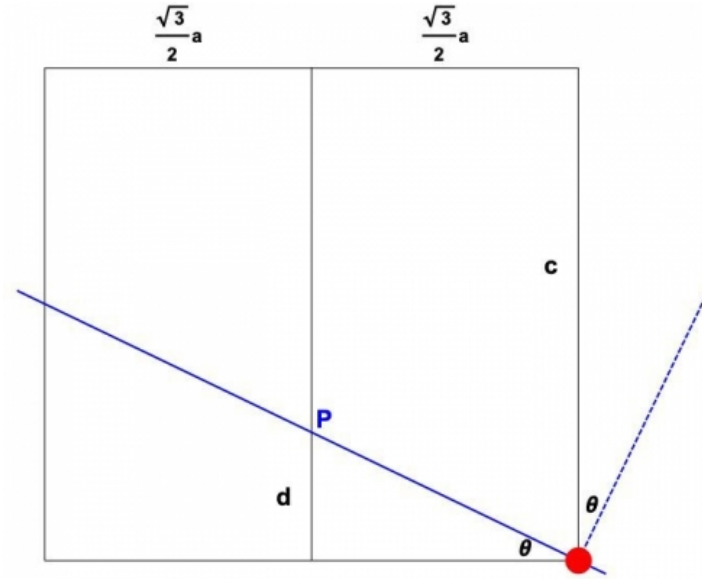


Figure 2.3.1.2: Cross-section of the wurtzite lattice. The dashed blue line indicates the  $\mathbf{x}'$ -direction, which is normal to the interface (solid blue line).

We note that the expression in the third case includes the other two special cases. To approximate the direction with integer entries, we multiply 100 and take the floor function:

```
$gamma = $c_InGaN / $a_InGaN # c/a ratio
# ideal c/a ratio in wurtzite is SQRT(8/3)=1.63299
$h = floor(100*sin(theta))
$l = floor(100*2*gamma*cos(theta)/sqrt(3))
x_hkl = [$h, 0, $l] # x axis perpendicular to (hkl) plane = (hkil) plane
```

Since nextnano<sup>3</sup> does not support variables in the Miller-Bravais indices, we explicitly give the indices in the input file using statements like !IF %ORIENTATION40 hkil-x-direction = 64 0 -64 143.

## 2.3.2 Parameter sweep of the angle using Template: Sweep over the variable theta

- Input file: *Romanov\_InGaN\_theta\_nnp.in*

One can make use of ‘**Template**’ feature of *nextnanomat* to sweep the angle  $\theta$  and obtain crystal orientation dependence of several physical quantities. Here, calculation is performed for every 5 degrees.

We obtain the angle dependence using ‘**post-processing**’ feature. Here, we collect the strain tensor components  $\epsilon_{xx}$ ,  $\epsilon_{yy}$ ,  $\epsilon_{zz}$ ,  $\epsilon_{xy}$ ,  $\epsilon_{xz}$  and  $\epsilon_{yz}$  that are in columns 2, 3, 4, 5, 6, 7 of the file *strain\_simulation.dat*.

- Select file containing values for the strain tensor components *strain\_simulation.dat* by clicking on the folder icon below *post-processing*.
- Select 1 for the *Maximum number of values lines*.

- Select 2 for the *Number of relevant column*. (to do: Improve nextnanomat to include all columns.)
- Click on *Create file with combined data* to generate file `theta_strain_simulation_Column2.dat`.
- Select 3 for the *Number of relevant column*.
- Click on *Create file with combined data* to generate file `theta_strain_simulation_Column3.dat`.
- Select 4 for the *Number of relevant column*.
- Click on *Create file with combined data* to generate file `theta_strain_simulation_Column4.dat`.
- Select 5 for the *Number of relevant column*.
- Click on *Create file with combined data* to generate file `theta_strain_simulation_Column5.dat`.
- Select 6 for the *Number of relevant column*.
- Click on *Create file with combined data* to generate file `theta_strain_simulation_Column6.dat`.
- The post-processing results are contained in the folder `<name_of_input_file>_postprocessing`.
- Finally, the plotted results of the post-processing file can be exported to gnuplot. Add all columns to the Overlay, and then click on: *Create and Open Gnuplot (\*.plt) from Items of Overlay*

### 2.3.3 Strain

Figure 2.3.3.1 and Figure 2.3.3.2 are the strain tensor elements as a function of inclination angle  $\theta$ , with respect to **simulation** and **crystal** coordinate systems, respectively. One can confirm that they reproduce correctly Figure 5 and 6 in [Romanov2006]. Please note that Romanov takes  $z'$ -axis as growth direction, while we take  $x'$ -axis. Therefore  $x'$ - and  $z'$ -axes are interchanged from [Romanov2006].

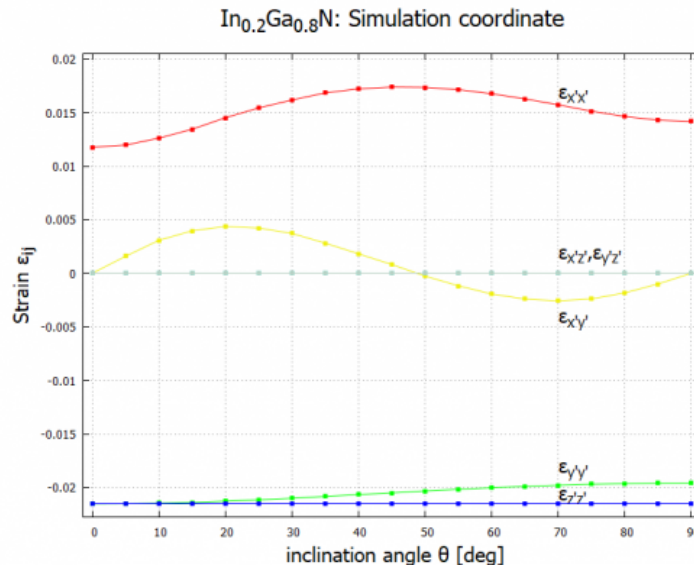


Figure 2.3.3.1: Elastic strain tensor components as a function of c-axis inclination angle  $\theta$  in **simulation** coordinate system.

### 2.3.4 Piezoelectric effect (first-order)

The piezoelectric effect is at first instance described by a linear response against strain. In crystal coordinate system,

$$P_{\mu}^{(1)} = \sum_{j=1}^6 e_{\mu j} \epsilon_j,$$

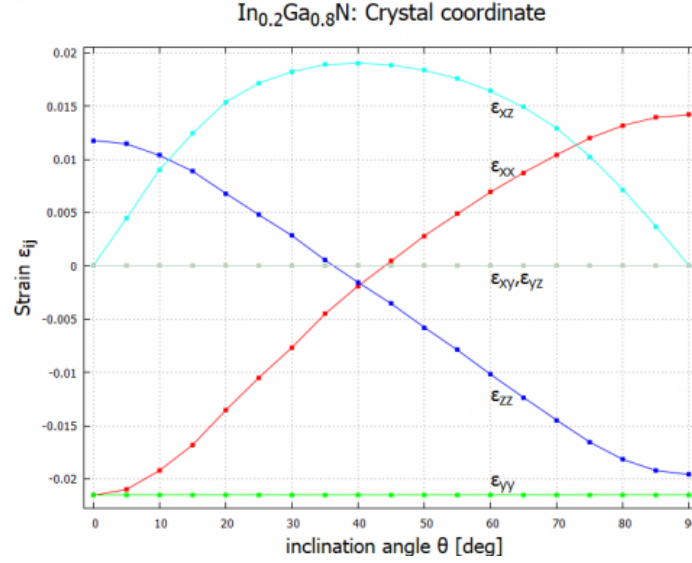


Figure 2.3.3.2: Elastic strain tensor components as a function of c-axis inclination angle  $\theta$  in **crystal** coordinate system.

where  $\mu = 1, 2, 3$  and the strain tensor is expressed in six-dimensional Voigt notation

$$\begin{pmatrix} \epsilon_1 \\ \epsilon_2 \\ \epsilon_3 \\ \epsilon_4 \\ \epsilon_5 \\ \epsilon_6 \end{pmatrix} = \begin{pmatrix} \epsilon_{xx} \\ \epsilon_{yy} \\ \epsilon_{zz} \\ 2\epsilon_{yz} \\ 2\epsilon_{xz} \\ 2\epsilon_{xy} \end{pmatrix}.$$

Please note that the indices  $x, y, z$  without prime refer to the axes of the crystal coordinate system. The superscript <sup>(1)</sup> indicates first-order piezoeffect. For the symmetry of the wurtzite structure, only three parameters remain in the piezoelectric coefficient tensor  $e_{ij}$

$$\begin{pmatrix} P_x^{(1)} \\ P_y^{(1)} \\ P_z^{(1)} \end{pmatrix} = \begin{pmatrix} 0 & 0 & 0 & 0 & e_{15} & 0 \\ 0 & 0 & 0 & e_{15} & 0 & 0 \\ e_{31} & e_{31} & e_{33} & 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} \epsilon_{xx} \\ \epsilon_{yy} \\ \epsilon_{zz} \\ 2\epsilon_{yz} \\ 2\epsilon_{xz} \\ 2\epsilon_{xy} \end{pmatrix} = \begin{pmatrix} 2e_{15}\epsilon_{xz} \\ 2e_{15}\epsilon_{yz} \\ e_{31}(\epsilon_{xx} + \epsilon_{yy}) + e_{33}\epsilon_{zz} \end{pmatrix},$$

cf. Eq. (4) in [Schulz2015]. **Note that Eq. (14) in [Romanov2006] misses the factor 2 for off-diagonal elements of the strain tensor.** These equations are implemented in both the *nextnano++* and *nextnano<sup>3</sup>* tools with corresponding material parameters in the respective database. The following flags export the strain tensor components and piezoelectric polarization vector in **crystal** and **simulation** coordinate systems (cf. [nextnano3](#) documentation). The piezoelectric polarization vector with respect to the simulation coordinate system can be found in the file `Strain\piezoelectric_polarization_vector_simulation.dat`.

```
strain{
  output_strain_tensor{
    crystal_system = yes
    simulation_system = yes
  }

  output_polarization_vector{
    crystal_system = yes
    simulation_system = yes
  }
}
```

(continues on next page)

(continued from previous page)

```

output_polarization_vector_components{
    crystal_system    = yes
    simulation_system = yes
}
}

```

```

$output-strain
strain-simulation-system = yes
strain-crystal-system   = yes
polarization-vector     = yes
$end_output-strain

```

For consistency, we have used the same material parameters as [Romanov2006], i.e. we have overwritten our default material parameters of the database with the values specified in the input file.

Analytical expression is derived as follows [Schulz2015]. Since we are interested in the polarization normal to the interface, it is useful to switch to the simulation coordinate system  $(x', y', z')$ . This can be done by transforming the polarization vector and the strain tensor to the simulation system,

$$P_{\mu'}^{(1)} = \left( R P^{(1)} \right)_{\mu'} = \sum_{\mu=1}^3 R_{\mu'\mu} P_{\mu}^{(1)}, \quad \epsilon_{\mu'\nu'} = (R \epsilon R^{-1})_{\mu'\nu'} = \sum_{\mu,\nu=1}^3 R_{\mu'\mu} R_{\nu'\nu} \epsilon_{\mu\nu},$$

where the  $3 \times 3$  rotation matrix  $R$  accounts for a rotation of angle  $\theta$

and we have used the fact that the rotation matrix is orthogonal:  $(R^{-1})_{\mu\nu} = R_{\nu\mu}$ . Prime denotes the axes in simulation coordinate system. These equations can be expressed in vector form as

$$\begin{pmatrix} P_{x'}^{(1)} \\ P_{y'}^{(1)} \\ P_{z'}^{(1)} \end{pmatrix} = R^{-1}(\theta) \begin{pmatrix} P_{x'}^{(1)} \\ P_{y'}^{(1)} \\ P_{z'}^{(1)} \end{pmatrix}, \quad \begin{pmatrix} \epsilon_{xx} \\ \epsilon_{yy} \\ \epsilon_{zz} \\ 2\epsilon_{yz} \\ 2\epsilon_{xz} \\ 2\epsilon_{xy} \end{pmatrix} = S^{-1}(\theta) \begin{pmatrix} \epsilon_{x'x'} \\ \epsilon_{y'y'} \\ \epsilon_{z'z'} \\ 2\epsilon_{y'z'} \\ 2\epsilon_{x'z'} \\ 2\epsilon_{x'y'} \end{pmatrix}$$

where  $S(\theta)$  is a  $6 \times 6$  matrix. The second transformation is given in Eq. (13) in [Romanov2006]. From equations above, we obtain the first-order piezoelectric effect in the simulation coordinate system

$$\begin{pmatrix} P_{x'}^{(1)} \\ P_{y'}^{(1)} \\ P_{z'}^{(1)} \end{pmatrix} = R(\theta) \begin{pmatrix} 0 & 0 & 0 & 0 & e_{15} & 0 \\ 0 & 0 & 0 & e_{15} & 0 & 0 \\ e_{31} & e_{31} & e_{33} & 0 & 0 & 0 \end{pmatrix} S^{-1}(\theta) \begin{pmatrix} \epsilon_{x'x'} \\ \epsilon_{y'y'} \\ \epsilon_{z'z'} \\ 2\epsilon_{y'z'} \\ 2\epsilon_{x'z'} \\ 2\epsilon_{x'y'} \end{pmatrix}.$$

The  $z'$ -component is explicitly

$$\begin{aligned} P_{z'}^{(1)} = & e_{31} \cos \theta \epsilon_{x'x'} \\ & + \left( e_{31} \cos^3 \theta + \frac{e_{33} - 2e_{15}}{2} \sin \theta \sin 2\theta \right) \epsilon_{y'y'} \\ & + \left( \frac{e_{31} + 2e_{15}}{2} \sin \theta \sin 2\theta + e_{33} \cos^3 \theta \right) \epsilon_{z'z'} \\ & + [(e_{31} - e_{33}) \cos \theta \sin 2\theta + 2e_{15} \sin \theta \cos 2\theta] \epsilon_{y'z'}. \end{aligned}$$

**Note that the corresponding analytical expression Eq. (18) in [Romanov2006] misses the factor 2 in front of  $e_{15}$  in the 2nd, 3rd and 4th line, and contains a typo in the 3rd line, i.e.  $e_{33}$  has to be  $e_{31}$  in the first term.** Our expression is consistent to eq. (5) in [Schulz2015]. Figure 2.3.4.1 compares the results of the nextnano software with the results of [Romanov2006] and [Schulz2015], respectively. The analytical results in Figure 2.3.4.1 are the plot of the equation above, with an interchange of  $x'$ - and  $z'$ -axes.

From the results in Figure 2.3.4.1 we can see that the piezoelectric polarization vanishes at an intermediate angle around 38 degree and that it is maximized when the inclination angle is zero.



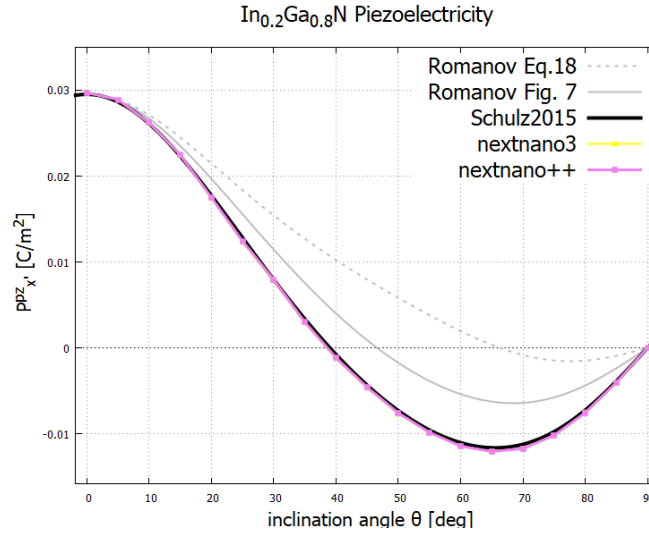


Figure 2.3.4.1: Piezoelectric polarization as a function of inclination angle. The gray dotted curve contains a typo  $e_{33} \leftrightarrow e_{31}$  and misses the factor 2. When the first typo is fixed, the gray solid curve is obtained and looks to be consistent with Figure 7(a) in [Romanov2006]. With the factor 2 the result becomes the black curve. Both *nextnano<sup>3</sup>* and *nextnano++* reproduce the black curve and are thus consistent to [Schulz2015].

### 2.3.5 Post-Processing for polarization

We obtain the angle dependence using ‘**post-processing**’ feature. Here, we collect the polarization components  $P_x$  that is in column 1 of the file `polarization_vector_piezoelectric_simulation.dat`.

- Select file containing values for the piezoelectric components `polarization_vector_piezoelectric_simulation.dat` by clicking on the folder icon below *post-processing*.
- Select 2 for the *Number of relevant column*.
- Select 1 for the *Maximum number of values lines*.
- Click on *Create file with combined data* to generate file `theta_polarization_vector_piezoelectric_simulation_Column1.dat`.
- The post-processing results are contained in the folder `<name_of_input_file>_postprocessing`.
- Finally, the plotted results of the post-processing file can be exported to gnuplot. Add all columns to the Overlay, and then click on: *Create and Open Gnuplot (\*.plt) from Items of Overlay*

### 2.3.6 Alloy content dependence

One can also sweep the alloy content  $x$ . The following results correspond to Figure 7(a) in [Romanov2006]. One can see that the zero point is universal for different alloy contents. The zero point is different compared to [Romanov2006] as he misses the factor of 2 for the strain tensor component. As can be seen in Figure 2.3.4.1 shown above, this mistake is not relevant for 0 and 90 degrees.

### 2.3.7 AlGaIn

- Input file: `Romanov_AlGaIn_theta_nnp.in`

Similarly, piezoelectric polarization of  $\text{Al}_x\text{Ga}_{1-x}\text{N}/\text{GaN}$  structure is calculated and shown in Figure 2.3.7.1. This result corresponds to Figure 8(a) in [Romanov2006]. The piezoelectric effect vanishes at around 38 degree in this case as well. Again, the zero point is different compared to [Romanov2006] as he misses the factor of 2 for the strain tensor component. As can be seen in Figure 2.3.4.1 shown above, this mistake is not relevant for 0 and 90 degrees.

The sign of the piezoelectric polarization in Figure 2.3.7.1 is opposite to the case of  $\text{InGaIn}/\text{GaN}$  composition

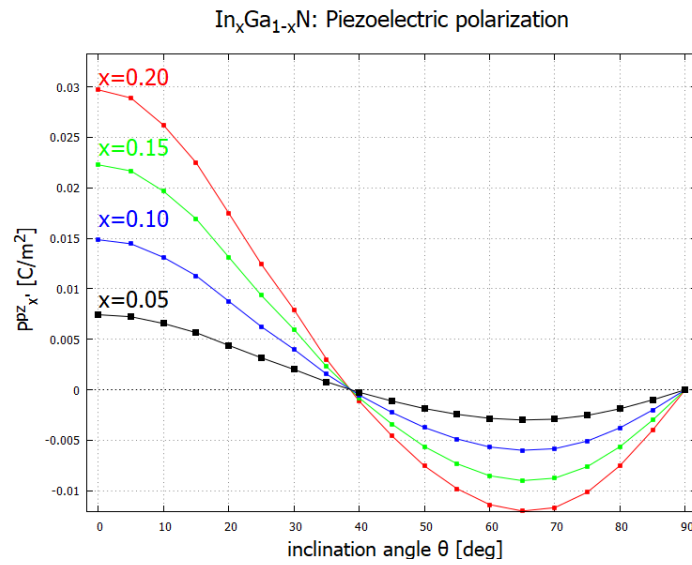


Figure 2.3.6.1: Alloy content dependence of the piezoelectric polarization for  $\text{In}_x\text{Ga}_{1-x}\text{N}/\text{GaN}$  structure.  $\text{In}_x\text{Ga}_{1-x}\text{N}$  is under biaxial compressive strain with respect to GaN.

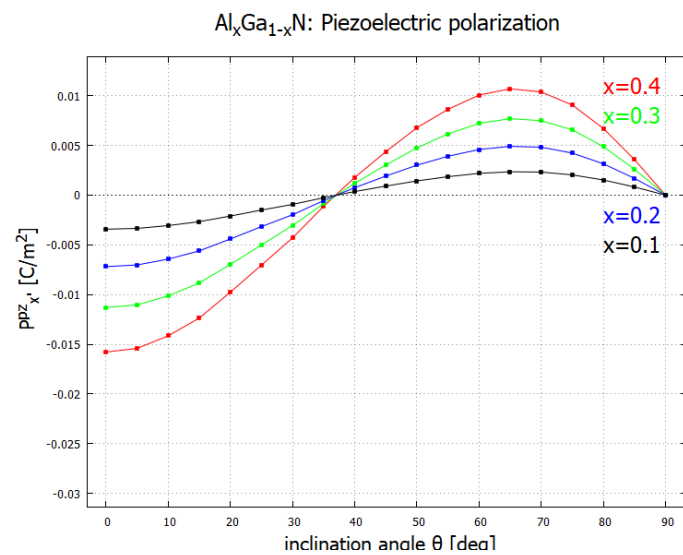


Figure 2.3.7.1: Alloy content dependence of the piezoelectric polarization for  $\text{Al}_x\text{Ga}_{1-x}\text{N}/\text{GaN}$  structure.  $\text{Al}_x\text{Ga}_{1-x}\text{N}$  is under biaxial tensile strain with respect to GaN.

(Figure 2.3.6.1). This is due to the fact that the lattice constants of InN, GaN and AlN obey the following relation

$$a_{\text{InN}} > a_{\text{GaN}} > a_{\text{AlN}}$$

(also for  $c$ ). Since we take GaN as a substitute,  $\text{In}_x\text{Ga}_{1-x}\text{N}$  layer is subject to compressive strain, whereas  $\text{Al}_x\text{Ga}_{1-x}\text{N}$  is under tensile strain [Romanov2006].

### 2.3.8 Piezoelectric effect (second-order)

- Input file: *Romanov\_InGaN\_theta\_nnp\_2nd.in*

Optimization of optoelectronic device design requires an accurate and detailed knowledge of the growth-direction dependence of the built-in electric field. Recently, the second order piezoelectric effect has been reported to be relevant for wurtzite III-N materials, namely GaN, AlN and InN. This potentially affects the electronic and optical properties of the devices. The piezoelectric polarization is generalized in crystal coordinate as [Patra2017]

$$P_{\mu}^{\text{pz}} = \sum_{j=1}^6 e_{\mu j} \epsilon_j + \frac{1}{2} \sum_{j,k=1}^6 B_{\mu jk} \epsilon_j \epsilon_k + \dots,$$

where  $e_{\mu j}$  and  $B_{\mu jk}$  are first- and second-order piezoelectric coefficients, respectively. For binary wurtzite structure, one can show that  $B_{\mu jk}$  has 8 independent components  $B_{311}, B_{312}, B_{313}, B_{333}, B_{115}, B_{125}, B_{135}, B_{344}$ . The explicit expression of the second-order term is given in Eq. (3) in [Patra2017], which is also implemented in *nextnano++* and *nextnano<sup>3</sup>*.

One can turn on the second-order contribution as

```
# nextnano++
strain{
    ...
    second_order_piezo = yes      # default: no
}
```

and in *nextnano3*,

```
! nextnano3
$numeric-control
    ...
    piezo-second-order = 2nd-order    ! [no/2nd-order]
$end_numeric-control
```

Figure 2.3.8.1 shows the results of the *nextnano* software. While the second-order contribution becomes negligible between the orientation  $(10\bar{1}3)$  and  $(10\bar{1}2)$ , and also between 85 and 95 degrees, it enhances the piezo effect up to 14% in other directions. This figure can be qualitatively compared to Figure 1(c) in [Patra2017], but note that they consider binary InN/GaN structure there while we are using  $\text{In}_{0.2}\text{Ga}_{0.8}\text{N}/\text{GaN}$ . The pink curve is different from the one in Figure 2.3.4.1 because we employed the material parameters used in [Patra2017]. The *nextnano<sup>3</sup>* tool also produces a consistent result (input file: *Romanov\_InGaN\_theta\_nn3\_2nd.in*). Within *nextnano<sup>3</sup>* one can also use different formulae for the second-order effect, cf. *nextnano3* documentation.

Last update: nnnn/nn/nn

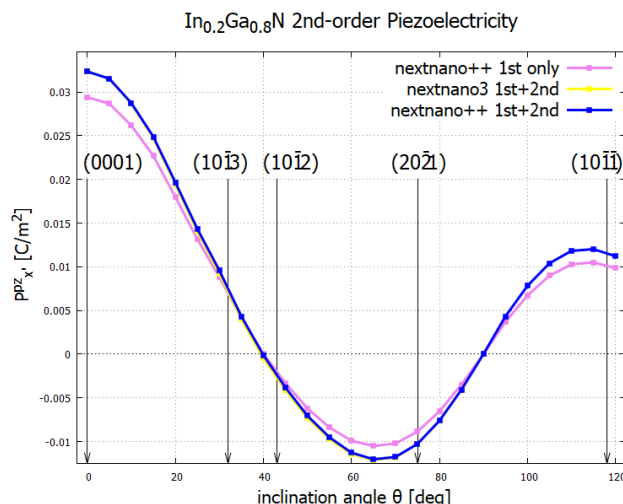


Figure 2.3.8.1: Second-order piezoelectricity. The second-order term enhances the piezoelectric polarization. The *nextnano<sup>3</sup>* result (yellow) is consistent to the *nextnano++* result (blue). Interface planes are indicated at corresponding angles.

## 2.4 Electrostatic potential

### 2.4.1 Poisson Equation

This equation governs the relation between the **electrostatic potential**  $\phi(\mathbf{x})$  and **total charge density distribution**  $\rho(\mathbf{x}, \phi)$  as follows:

$$-\nabla \cdot [\varepsilon_0 \varepsilon_r(\mathbf{x}) \nabla \cdot \phi(\mathbf{x})] = \rho(\mathbf{x}, \phi) \quad (2.4.1.1)$$

where  $\varepsilon_0$  is the vacuum permittivity,  $\varepsilon_r$  is the material dependent static dielectric constant. And the total charge density distribution consists of the **densities of ionized donors**  $N_D^+$ , **ionized acceptors**  $N_A^-$ , **piezoelectric and pyroelectric charge**  $\rho_{pz}$  and  $\rho_{py}$ , besides the **carrier densities**  $n(\mathbf{x}, \phi)$  and  $p(\mathbf{x}, \phi)$ , which are calculated either classically or quantum mechanically:

$$\rho(\mathbf{x}, \phi) = e[-n(\mathbf{x}, \phi) + p(\mathbf{x}, \phi) + N_D^+(\mathbf{x}) - N_A^-(\mathbf{x}) + \rho_{pz}(\mathbf{x}) + \rho_{py}(\mathbf{x})] \quad (2.4.1.2)$$

When the Schrödinger-Poisson equation is solved the carrier densities defined in either multi-band model or single-band model are substituted into this  $\rho(\mathbf{x}, \phi)$  and the Poisson equation is solved accordingly. Then the resulting  $\phi(\mathbf{x})$  is returned into the Schrödinger equation and the carrier densities are calculated once again.

This cycle is continued until the carrier densities satisfies the convergence criteria. The final result of  $n(\mathbf{x}, \phi)$ ,  $p(\mathbf{x}, \phi)$  and  $\phi(\mathbf{x})$  must satisfy both Schrödinger and Poisson equations, or we can say that **the Schrödinger equation and Poisson equation are self-consistent with respect to the resulting carrier densities and electrostatic potential**.

On the other hand, when only the Poisson equation is solved, the carrier densities are calculated according to (2.5.1.1) and (2.5.1.2) instead. We can say in other words that the **carrier density calculation in the context of Thomas-Fermi approximation and the Poisson equation are self-consistent with respect to the resulting carrier densities and electrostatic potential**.

---

*Last update: 04/12/2024*

## 2.5 Charge densities

### 2.5.1 Classical

Things are much more simpler.

When any kind of Schrödinger equation is not solved, the charge carrier densities are estimated from the position-dependent conduction and valence band edges  $E_c^i(\mathbf{x})$  and  $E_v^i(\mathbf{x})$ , quasi-Fermi levels, and the electrostatic potential  $\phi(\mathbf{x})$  in the context of Thomas-Fermi approximation.

These **classical charge carrier densities** are calculated as

$$n(\mathbf{x}) = \sum_{i \in \text{CB}} N_c^i(T) \mathcal{F}_{1/2} \left( [-E_c^i(\mathbf{x}) + e\phi(\mathbf{x}) + E_{F,n}(\mathbf{x})]/kT \right) \quad (2.5.1.1)$$

$$p(\mathbf{x}) = \sum_{i \in \text{VB}} N_v^i(T) \mathcal{F}_{1/2} \left( [E_v^i(\mathbf{x}) - e\phi(\mathbf{x}) - E_{F,p}(\mathbf{x})]/kT \right). \quad (2.5.1.2)$$

Here  $N_v^i(T)$  and  $N_c^i(T)$  are the equivalent density of states at the conduction and valence band edges, which are given by

$$N_l^i(T) = g_l^i \left( \frac{m_{\text{dos},\lambda}^i kT}{2\pi\hbar} \right)^{2/3} \quad (l, \lambda) = (\text{v}, \text{h}), \text{ or } (\text{c}, \text{e}). \quad (2.5.1.3)$$

Here  $m_{\text{dos},\lambda}^i$  is the density-of-mass for  $d = 3$  defined in (2.5.2.10).

This calculation of carrier densities is much faster than the quantum mechanical calculation, but the quantum effect such as energy quantization, carrier leakage into the barrier, etc. cannot be taken into account.

Also in this case, the carrier densities can be written as  $n(\mathbf{x}, \phi)$  and  $p(\mathbf{x}, \phi)$ , which enters into the non-linear Poisson equation introduced next.

Moreover, when the current equation is included in the calculation scheme, seeing the carrier densities as  $n(\mathbf{x}, \phi, E_{F,n})$  and  $p(\mathbf{x}, \phi, E_{F,p})$  makes it easy to understand what the self-consistent calculation is actually doing.

## 2.5.2 Quantum mechanical

### Multi-band model (k · p model)

Once the  $\mu$ -th component envelope function of the  $j$ -th eigenstate of electron ( $l = \text{c}$ ) or hole ( $l = \text{v}$ ) in the  $i$ -th band is obtained as  $(F_\mu)_{l,j}^i(\mathbf{x})$  from the **multi-band Schrödinger equation**, the probability distribution of this  $j$ -th eigenstate reads

$$p_{l,j}^i(\mathbf{x}) = \sum_{\mu} \left| (F_\mu)_{l,j}^i(\mathbf{x}) \right|^2. \quad (2.5.2.1)$$

where we are assuming 3D structure so far.

Then the **quantum mechanical carrier densities for 3D structure** are defined from these probability densities, energy eigenvalues  $E_{c,j}$  and  $E_{v,j}$ , position-dependent quasi-Fermi levels  $E_{F,n}(\mathbf{x})$  and  $E_{F,p}(\mathbf{x})$  as

$$n(\mathbf{x}) = \sum_{i \in \text{CB}} g_c^i \sum_j p_{c,j}^i(\mathbf{x}) f \left( [E_{c,j}^i - E_{F,n}(\mathbf{x})]/kT \right) \quad (2.5.2.2)$$

$$p(\mathbf{x}) = \sum_{i \in \text{VB}} g_v^i \sum_j p_{v,j}^i(\mathbf{x}) f \left( [-E_{v,j}^i + E_{F,p}(\mathbf{x})]/kT \right) \quad (2.5.2.3)$$

where  $f(E)$  is the Fermi-Dirac distribution at temperature  $T$ ,  $g_c^i$  and  $g_v^i$  represent the possible spin and valley degeneracies.

When the simulation is over **1D structure**, the wave function can be separated into the plane wave specified with the lattice wave vector  $\mathbf{k}_{\parallel}$  in the lateral 2D direction and the quantized wave function in the growth direction, which has the  $\mathbf{k}_{\parallel}$ -dependency. Then the charge carrier density is obtained by the following integral over  $\mathbf{k}_{\parallel}$ :

$$n(x) = \sum_{i \in \text{CB}} g_c^i \sum_j \frac{1}{(2\pi)^2} \int_{\Omega_{BZ}} d^2\mathbf{k}_{\parallel} p_{c,j}^i(x, \mathbf{k}_{\parallel}) f \left( [E_{c,j}^i(\mathbf{k}_{\parallel}) - E_{F,n}(x)]/kT \right) \quad (2.5.2.4)$$

$$p(x) = \sum_{i \in \text{VB}} g_v^i \sum_j \frac{1}{(2\pi)^2} \int_{\Omega_{BZ}} d^2\mathbf{k}_{\parallel} p_{v,j}^i(x, \mathbf{k}_{\parallel}) f \left( [-E_{v,j}^i(\mathbf{k}_{\parallel}) + E_{F,p}(x)]/kT \right) \quad (2.5.2.5)$$

Here the integration is over the two-dimensional Brillouin zone  $\Omega_{BZ}$ .

Similarly, the charge carrier densities for **2D structure** is calculated by the integral over the 1-dimensional Brillouin zone as

$$n(\mathbf{x}) = \sum_{i \in \text{CB}} g_c^i \sum_j \frac{1}{2\pi} \int_{\Omega_{BZ}} dk p_{c,j}^i(\mathbf{x}, \mathbf{k}) f\left([E_{c,j}^i(k) - E_{F,n}(\mathbf{x})]/kT\right) \quad (2.5.2.6)$$

$$p(\mathbf{x}) = \sum_{i \in \text{VB}} g_v^i \sum_j \frac{1}{2\pi} \int_{\Omega_{BZ}} dk p_{v,j}^i(\mathbf{x}, \mathbf{k}) f\left([-E_{v,j}^i(k) + E_{F,p}(\mathbf{x})]/kT\right) \quad (2.5.2.7)$$

### Single-band model

Things are simpler.

When the **single-band Schrödinger equation** is set to be solved, the envelope function of the  $j$ -th eigenstate has only one component  $F_{l,j}^i(\mathbf{x})$ . Also, the  $\mathbf{k}$ -integration in (2.5.2.4) to (2.5.2.7) can be done analytically due to the parabolic dispersion according to the effective mass tensor  $\underline{m}_e^{*i}$  and  $\underline{m}_h^{*i}$ .

Thanks to this simplicity the **quantum mechanical charge carrier densities** for  $d$ -dimensional simulation can be written up by the following expression:

$$n(\mathbf{x}) = \sum_{i \in \text{CB}} g_c^i \left(\frac{m_{\text{dos},e} kT}{2\pi\hbar^2}\right)^{(3-d)/2} \sum_j p_{c,j}^i(\mathbf{x}) \mathcal{F}_{(1-d)/2}\left([E_{c,j}^i - E_{F,n}(\mathbf{x})]/kT\right) \quad (2.5.2.8)$$

$$p(\mathbf{x}) = \sum_{i \in \text{VB}} g_v^i \left(\frac{m_{\text{dos},h} kT}{2\pi\hbar^2}\right)^{(3-d)/2} \sum_j p_{v,j}^i(\mathbf{x}) \mathcal{F}_{(1-d)/2}\left([-E_{v,j}^i + E_{F,p}(\mathbf{x})]/kT\right) \quad (2.5.2.9)$$

TODO: The sign in the fermi-dirac integral might be opposite. check the source code.

Here  $\mathcal{F}_n(E)$  denotes the Fermi-Dirac integral of order  $n$  and  $m_{\text{dos},\lambda}^i$  is so-called density-of-states mass defined as

$$m_{\text{dos},\lambda}^i = (\det \bar{m}_{\lambda}^{*i}) \quad \lambda = e, h \quad (2.5.2.10)$$

where  $\bar{m}_{\lambda}^{*i}$  describes the  $2 \times 2$  or  $1 \times 1$  submatrix of the effective mass tensor  $\underline{m}_{\lambda}^{*i}$  in the direction of  $\mathbf{k}_{\parallel}$ .

In any cases, the carrier densities are dependent on the electrostatic potential  $\phi(\mathbf{x})$  through the wave function, which is obtained from the  $\phi$ -dependent Hamiltonian  $H(\phi)$ . Thus we can also write them as  $n(\mathbf{x}, \phi)$  and  $p(\mathbf{x}, \phi)$ , which enters into the non-linear Poisson equation introduced later.

Moreover, when the current equation is included in the calculation scheme, seeing the carrier densities as  $n(\mathbf{x}, \phi, E_{F,n})$  and  $p(\mathbf{x}, \phi, E_{F,p})$  makes it easy to understand what the self-consistent calculation is actually doing.

---

*Last update: 04/12/2024*

## 2.6 Doping

### 2.6.1 Activation Energies

Table 2.6.1.1: Donor levels (n-type) in units of  $eV$  relative to conduction band edge

| Donor Name                                       | En-ergy | Source                                                                        |
|--------------------------------------------------|---------|-------------------------------------------------------------------------------|
| n-As-in-Si                                       | 0.054   |                                                                               |
| n-As-in-Si                                       | 0.049   | American Institute of Physics Handbook, 3rd ed., McGraw-Hill, New York (1972) |
| n-P-in-Si                                        | 0.045   | American Institute of Physics Handbook, 3rd ed., McGraw-Hill, New York (1972) |
| n-Sb-in-Si                                       | 0.039   |                                                                               |
| n-N-in-Si                                        | 0.045   |                                                                               |
| n-As-in-Ge                                       | 0.013   | American Institute of Physics Handbook, 3rd ed., McGraw-Hill, New York (1972) |
| n-P-in-Ge                                        | 0.012   | American Institute of Physics Handbook, 3rd ed., McGraw-Hill, New York (1972) |
| n-N-in-SiC                                       | 0.10    |                                                                               |
| n-Si-in-GaAs                                     | 0.0058  |                                                                               |
| n-Si-in-AlAs                                     | 0.007   | 300 K, Landolt-Boernstein                                                     |
| n-Si-in-Al <sub>0.27</sub> Ga <sub>0.73</sub> As | 0.006   | Landolt-Boernstein                                                            |

More parameters can be found in the *nextnano*<sup>3</sup> database file *database\_nn3.in* or at this [link](#)

Table 2.6.1.2: Acceptor levels (p-type) in units of  $eV$  relative to valence band edge

| Acceptor Name | En-ergy | Source                                                                        |
|---------------|---------|-------------------------------------------------------------------------------|
| p-In-in-Si    | 0.16    |                                                                               |
| p-B-in-Si     | 0.045   | American Institute of Physics Handbook, 3rd ed., McGraw-Hill, New York (1972) |
| p-Al-in-Si    | 0.057   | American Institute of Physics Handbook, 3rd ed., McGraw-Hill, New York (1972) |
| p-B-in-Ge     | 0.010   | American Institute of Physics Handbook, 3rd ed., McGraw-Hill, New York (1972) |
| p-Al-in-Ge    | 0.010   | American Institute of Physics Handbook, 3rd ed., McGraw-Hill, New York (1972) |
| p-Al-in-SiC   | 0.20    |                                                                               |
| p-C-in-GaAs   | 0.027   | Landolt-Boernstein 1982                                                       |

More parameters can be found in the *nextnano*<sup>3</sup> database file *database\_nn3.in* or at this [link](#)

## 2.7 Incomplete ionization

The densities of ionized impurities are calculated in the context of Thomas-Fermi approximation with these formulas:

$$N_D^+(\mathbf{x}) = \sum_{i \in \text{Donors}} \frac{N_{D,i}(\mathbf{x})}{1 + g_{D,i} \exp((E_{F,n}(\mathbf{x}) - E_{D,i}(\mathbf{x}))/k_B T)} \quad (2.7.1)$$

$$N_A^-(\mathbf{x}) = \sum_{i \in \text{Acceptors}} \frac{N_{A,i}(\mathbf{x})}{1 + g_{A,i} \exp((E_{A,i}(\mathbf{x}) - E_{F,p}(\mathbf{x}))/k_B T)} \quad (2.7.2)$$

where the summation is over all different donor or acceptors,  $N_D, N_A$  are the doping concentrations,  $g_D, g_A$  are the degeneracy factors ( $g_D = 2$  and  $g_A = 4$  for shallow impurities), and  $E_D, E_A$  are the energies of the neutral donor and acceptor impurities, respectively.

These energies of neutral impurities  $E_{D,i}, E_{A,i}$  are determined by the ionization energies  $E_{D,i}^{\text{ion}}, E_{A,i}^{\text{ion}}$ , the bulk conduction and valence band edges (including shifts due to strain) and the electrostatic potential.

$$E_{D,i}(\mathbf{x}) = E_c(\mathbf{x}) - e\phi(\mathbf{x}) - E_{D,i}^{\text{ion}}(\mathbf{x}) \quad (2.7.3)$$

$$E_{A,i}(\mathbf{x}) = E_v(\mathbf{x}) - e\phi(\mathbf{x}) + E_{A,i}^{\text{ion}}(\mathbf{x}) \quad (2.7.4)$$

---

Last update: 04/12/2024

## 2.8 Carrier transport

### 2.8.1 Drift-Diffusion Model

The continuity equations in the presence of generation  $G$  recombination  $R$  of electron-hole pairs read

$$\begin{aligned} -e \frac{\partial n}{\partial t} + \nabla \cdot (-e \mathbf{j}_n(\mathbf{x})) &= -e(G(\mathbf{x}) - R(\mathbf{x})), \\ e \frac{\partial p}{\partial t} + \nabla \cdot (e \mathbf{j}_p(\mathbf{x})) &= e(G(\mathbf{x}) - R(\mathbf{x})), \end{aligned} \quad (2.8.1.1)$$

where the current is proportional to the gradient of quasi Fermi levels  $E_{F,n/p}(\mathbf{x})$

$$\begin{aligned} \mathbf{j}_n(\mathbf{x}) &= -\mu_n(\mathbf{x})n(\mathbf{x})\nabla E_{F,n}(\mathbf{x}), \\ \mathbf{j}_p(\mathbf{x}) &= \mu_p(\mathbf{x})p(\mathbf{x})\nabla E_{F,p}(\mathbf{x}). \end{aligned} \quad (2.8.1.2)$$

Here the charge current has the unit of (area)<sup>-1</sup>(time)<sup>-1</sup>.  $\mu_{n/p}$  are the mobilities of each carrier.

Hereafter we consider stationary solutions and set  $\dot{n} = \dot{p} = 0$ . The governing equations then reduce to

$$\begin{aligned} \nabla \cdot \mu_n(\mathbf{x})n(\mathbf{x})\nabla E_{F,n}(\mathbf{x}) &= -(G(\mathbf{x}) - R(\mathbf{x})), \\ \nabla \cdot \mu_p(\mathbf{x})p(\mathbf{x})\nabla E_{F,p}(\mathbf{x}) &= G(\mathbf{x}) - R(\mathbf{x}), \end{aligned} \quad (2.8.1.3)$$

which we call **current equation**.

We can also say that the current equation governs the relationship between the **carrier densities**  $n(\mathbf{x}), p(\mathbf{x})$  and **quasi Fermi levels**  $E_{F,n/p}(\mathbf{x})$ .

The *nextnano++* tool solves this equation and Poisson equation (and also Schrödinger equation) self-consistently.

In their solution, the corresponding calculation of the carrier densities ( $n(\mathbf{x}, \phi, E_{F,n}), p(\mathbf{x}, \phi, E_{F,p})$ ) and Poisson equation are firstly iterated for a given quasi-Fermi levels until the carrier densities converge. Then the resulting carrier densities are substituted into the current equation and the quasi-Fermi levels are updated. This whole cycle is iterated until the quasi-Fermi levels satisfies the convergence criteria, which can be tuned by the users.

---

Last update: 04/12/2024

## 2.9 Mobility

This section describes all mobility models implemented in the [nextnano software](#).



- *Low-field mobility models*
  - *Constant*
  - *Masetti*
  - *Arora*
  - *MINIMOS 6*
  - *Simba*
- *High-Field Mobility Models*
  - *Hänsch*
  - *Extended Canali*
  - *Transferred-Electron*
  - *Eastman-Tiwari-Shur*

### **Note**

If you need more mobility models implemented in *nextnano++*, [contact us](#)

## 2.9.1 Low-field mobility models

Four low-field following mobility models are supported in *nextnano++*.

### Constant

The constant mobility model is due to lattice scattering (phonon scattering) and leads to a constant mobility that depends only on the temperature  $T$ . The lattice atoms oscillate about their equilibrium sites at finite temperature leading to a scattering of carriers which results in a temperature dependent mobility  $\mu_{const}^{n,p}$ .  $\mu_{max}^{n,p}$  is the mobility due to bulk phonon (lattice) scattering. For all semiconductors the temperature dependent lattice mobility is modeled by a power law:

$$\mu_{const}^{n,p}(T) = \mu_{max}^{n,p} \cdot \left( \frac{T}{T_0} \right)^{-exponent}, \quad (2.9.1.1)$$

with temperature  $T$  and reference temperature  $T_0 = 300K$ .

The parameter values used in this model for electrons and holes, respectively, are taken from the PhD thesis of V. Palankovski [Simulation of Heterojunction Bipolar Transistors](#) (TU Vienna). (Note: The exponent has opposite sign in his PhD thesis.)

### Masetti

The Masetti bulk mobility model is used to simulate the doping dependent mobility in Si and takes into account the scattering of the carriers by charged impurity ions which leads to a degradation of the carrier mobility (ionized impurity scattering). It is a model that combines lattice and impurity scattering. This model is temperature independent and the parameters are given for 300 K. Thus it is only valid for 300 K.

Following [\[Masetti1983\]](#), the equation for mobility is :

$$\mu^{n,p} = \mu_{min1}^{n,p} \cdot e^{-\frac{P_c^{n,p}}{N_D+N_A}} + \frac{\mu_{const}^{n,p} - \mu_{min2}^{n,p}}{1 + \left( \frac{N_D+N_A}{C_r^{n,p}} \right)^{\alpha^{n,p}}} - \frac{\mu_1^{n,p}}{1 + \left( \frac{C_s^{n,p}}{N_D+N_A} \right)^{\beta^{n,p}}} \quad (2.9.1.2)$$

with the reference mobility parameters  $\mu_{min1}^{n,p}$ ,  $\mu_{min2}^{n,p}$  and  $\mu_1^{n,p}$ , the reference doping concentration parameters  $P_c^{n,p}$ ,  $C_r^{n,p}$ ,  $C_s^{n,p}$ ,  $\alpha^{n,p}$  and  $\beta^{n,p}$ , and the concentration of ionized donors  $N_D$  and acceptors  $N_A$ . The total concentration of ionized impurities is given by  $N_D + N_A$ . The low-doping reference mobility  $\mu_{const}^{n,p}$  is determined by equation (2.9.1.1) (constant mobility-model), i.e. the values in the database are the same as under this keyword.

## Arora

The Arora mobility model is used to simulate the doping dependent mobility in Si and takes into account the scattering of the carriers by charged impurity ions which leads to a degradation of the carrier mobility (ionized impurity scattering). This model is temperature dependent.

Following [Arora1982], the equation for mobility is:

$$\mu^{n,p} = \mu_{min}^{n,p} \cdot \left(\frac{T}{T_0}\right)^{\alpha_m^{n,p}} + \frac{\mu_d^{n,p} \cdot \left(\frac{T}{T_0}\right)^{\alpha_d^{n,p}}}{1 + \left(\frac{N_D + N_A}{N_0^{n,p} \cdot \left(\frac{T}{T_0}\right)^{\alpha_N^{n,p}}}\right)^{A_a^{n,p} \cdot \left(\frac{T}{T_0}\right)^{\alpha_a^{n,p}}}}, \quad (2.9.1.3)$$

with the reference mobility parameter  $\mu_{min}^{n,p}(T_0)$ , reference mobility parameter  $\mu_d^{n,p}$ , lattice temperature  $T$ , reference temperature  $T_0 = 300K$ , reference exponent parameter  $A_a^{n,p}$ , exponents  $\alpha_m^{n,p}$  and  $\alpha_a^{n,p}$ , reference impurity parameter  $N_0^{n,p}$ , and concentration of ionized donors  $N_D$  and acceptors  $N_A$ . The total concentration of ionized impurities is given by  $N_A + N_D$ .

## MINIMOS 6

The mobility model used in MINIMOS 6 is used to simulate the doping dependent mobility in Si and takes into account the scattering of the carriers by charged impurity ions which leads to a degradation of the carrier mobility (ionized impurity scattering). This model is temperature dependent and takes into account the reduced mobility due to lattice scattering (i.e. the values in the database are the same as under this keyword apart from the sign of the exponent). The formula of Caughey and Thomas [CaugheyThomas1967] is used together with temperature dependent coefficients. This model is well suited for Si. The equation for mobility is:

$$\mu^{n,p} = \mu_{min}^{n,p} + \frac{\mu_{const}^{n,p} - \mu_{min}^{n,p}}{1 + \left(\frac{N_D + N_A}{N_0^{n,p} \cdot \left(\frac{T}{T_0}\right)^{\alpha_N^{n,p}}}\right)^{A_a^{n,p} \cdot \left(\frac{T}{T_0}\right)^{\alpha_a^{n,p}}}}, \quad (2.9.1.4)$$

with lattice temperature  $T$ , reference temperature  $T_0 = 300K$ , reference exponent parameter  $A_a^{n,p}$ , exponents  $\alpha_m^{n,p}$  and  $\alpha_a^{n,p}$ , reference impurity parameter  $N_0^{n,p}$ , and concentration of ionized donors  $N_D$  and acceptors  $N_A$ . The total concentration of ionized impurities is given by  $N_D + N_A$ . The  $\mu_{const}^{n,p}$  is determined by the constant mobility-model: equation (2.9.1.1). The formulas for the reference mobility parameter  $\mu_{const}^{n,p}$  are

$$\mu_{min}^{n,p}(T) = \mu_{min}^{n,p}(T_0) \left(\frac{T}{T_0}\right)^{\alpha_m^{n,p}} \quad (2.9.1.5)$$

$$\mu_{min}^{n,p}(T) = \mu_{min}^{n,p}(T_0) \cdot \left(\frac{2}{3}\right)^{\alpha_m^{n,p}} \left(\frac{T}{200K}\right)^{\alpha_{m2}^{n,p}}, \quad (2.9.1.6)$$

where (2.9.1.5) applies to temperatures  $T \geq 200K$  and (2.9.1.6) to temperatures  $T < 200K$ . The value  $T = 200K$  can be changed by  $T_{Switch}$ . By setting  $\alpha_m^{n,p} = \alpha_{m2}^{n,p}$  and  $\alpha_a^{n,p} = 0$ , (2.9.1.6) reduces to (2.9.1.5) and this model can also be applied to other basic materials.

It is a model that combines lattice and impurity scattering.

The parameter values used in this model for electrons and holes, respectively, are taken from the PhD thesis of V. Palankovski *Simulation of Heterojunction Bipolar Transistors* (TU Vienna). (Note: The exponent has opposite sign in his PhD thesis.)

## Simba

### ⚠ Attention

These models are implemented only in nextnano<sup>3</sup>

This is one possible model for the mobility parameter  $\mu^n$  (for electrons) and  $\mu^p$  (for holes) that is used in the drift-diffusion model. The model is taken from the SIMBA documentation [*CaugheyThomas1967*]. In this model the mobility depends on the three quantities: doping density, temperature and **E**-field. The contributions of these quantities to the mobility are calculated in the following order:

1) Doping concentration:

$$\mu^{n,p}(N_A + N_D) = \mu_{min}^{n,p} + \frac{\mu_D^{n,p}}{1 + \left(\frac{N_A + N_D}{N_{ref}^{n,p}}\right)^{\alpha^{n,p}}} \quad (2.9.1.7)$$

with minimum mobility  $\mu_{min}^{n,p}$ , reference doping density  $N_{ref}^{n,p}$ , reference mobility  $\mu_D^{n,p}$ , exponent  $\alpha^{n,p}$  and concentration of ionized acceptors  $N_A$  and donors  $N_D$ . Note that the nominal doping concentration, as specified in the input file and not the ionized one, is used in *nextnano*<sup>3</sup>.

2) Temperature:

$$\mu^{n,p}(T) = \mu_{max}^{n,p} \cdot \left(\frac{T}{T_0}\right)^{-\gamma^{n,p}}, \quad (2.9.1.8)$$

with temperature  $T$ , reference temperature  $T_0$  and exponent for temperature dependence  $\gamma^{n,p}$ .

3) Electric field (*perpendicular*):

$$\mu^{n,p}(\mathbf{E}_\perp) = \frac{\mu^{n,p}}{\sqrt{1 + \frac{\|\mathbf{E}_\perp\|}{E_T^{n,p}}}} \quad (2.9.1.9)$$

with perpendicular electric field parameter  $E_T^{n,p}$ . It is possible to include/ exclude the perpendicular **E**-field dependence.

4) Electric field (*parallel*):

There are six different SIMBA models for including the impact of the parallel electric field:

#### Model 0

no dependence on parallel electric field

#### Model 1

$$\mu^{n,p}(\mathbf{E}) = \frac{\mu^{n,p}}{\left[1 + \left(\frac{\|\mathbf{E}\|}{E_p^{n,p}}\right)^{\alpha^{n,p}}\right]^{\beta^{n,p}}} \quad (2.9.1.10)$$

with exponents  $\alpha^{n,p}$  and  $\beta^{n,p}$ . The temperature dependency of peak electric field is described by:

$$E_p^{n,p}(T) = E_0^{n,p} - d_E^{n,p} \cdot (T - T_0), \quad (2.9.1.11)$$

with temperature  $T$ , peak electric field  $E_0^{n,p}$ , temperature dependence parameter of peak electric field  $d_E^{n,p}$  and reference temperature  $T_0$ .

#### Model 2

$$\mu^{n,p}(\mathbf{E}) = \frac{\mu^{n,p}}{\left[1 + \left(\mu^{n,p} \frac{\|\mathbf{E}\|}{v_s^{n,p}}\right)^{\kappa^{n,p}}\right]^{1/\kappa^{n,p}}} \quad (2.9.1.12)$$

with exponent  $\kappa^{n,p}$  and saturation velocity  $v_s^{n,p}(T)$ . Temperature dependency of saturation velocity is described by:

$$v_s^{n,p}(T) = v_0^{n,p} - d_v^{n,p} \cdot (T - T_0), \quad (2.9.1.13)$$

with reference saturation velocity  $v_0^{n,p}$ , temperature dependence parameter of saturation velocity  $d_v^{n,p}$  and reference Temperature  $T_0^{n,p}$ .

### Model 3

$$\mu^{n,p}(\mathbf{E}) = \frac{\mu^{n,p} + v_s^{n,p} \frac{\|\mathbf{E}\|^3}{E_p^{n,p}}}{\left[ 1 + \left( \frac{\|\mathbf{E}\|}{E_p^{n,p}} \right)^{\alpha^{n,p}} \right]^{\beta^{n,p}}} \quad (2.9.1.14)$$

### Model 4

$$\mu^{n,p}(\mathbf{E}) = \frac{2\mu^{n,p}}{1 + \left[ 1 + \left( \mu^{n,p} \frac{\|\mathbf{E}\|}{v_s^{n,p}} \right)^{\kappa^{n,p}} \right]^{1/\kappa^{n,p}}} \quad (2.9.1.15)$$

### Model 5

$$\mu^{n,p}(\mathbf{E}) = \frac{\mu^{n,p} + v_s^{n,p} \frac{(\|\mathbf{E}\|)^{\alpha^{n,p}-1}}{(E_p^{n,p})^{\alpha^{n,p}}}}{\left[ 1 + \left( \frac{\|\mathbf{E}\|}{E_p^{n,p}} \right)^{\alpha^{n,p}} \right]^{\beta^{n,p}}} \quad (2.9.1.16)$$

## 2.9.2 High-Field Mobility Models

Four high-field mobility models are currently implemented in *nextnano++*. In our implementation, each of them uses results obtained from selected low-field model passed via  $\mu_{\text{low}}$ .

### Hänsch

As mentioned above, this model is a special case of the Extended Canali model in the limit of strong surface scattering defined by *W. Hänsch and M. Miura-Mattausch*

$$\mu(F) = \frac{2\mu_{\text{low}}}{1 + \left( 1 + \left( 2 \frac{\mu_{\text{low}} F}{v_{\text{sat}}} \right)^2 \right)^{1/2}}$$

where  $\mu_{\text{low}}$  is low-field mobility,  $v_{\text{sat}}$  is saturation velocity, and  $F$  is the driving force.

### Extended Canali

The Extended Canali model is an extended version of Jacoboni-Canali model, originally applied to electron and hole drift-velocity measurements in silicon by *Canali, et al.*.

$$\mu(F) = \frac{(\alpha + 1)\mu_{\text{low}}}{\alpha + \left( 1 + \left( (\alpha + 1) \frac{\mu_{\text{low}} F}{v_{\text{sat}}} \right)^\beta \right)^{1/\beta}}$$

where  $\mu_{\text{low}}$  is low-field mobility,  $v_{\text{sat}}$  is saturation velocity, and  $F$  is the driving force. Parameters  $\alpha$ ,  $\beta$  and  $v_{\text{sat}}$  are defined independently for holes and electrons. The driving force  $F$  of the respective carriers is evaluated as the gradient of the respective quasi-Fermi level. The  $\alpha$  parameter should be set to zero, if one aims at using the Extended Canali model. One can transform it into Hänsch model by setting  $\alpha = 1$  and  $\beta = 2$ .

## Transferred-Electron

The transferred electron model below bases on Monte Carlo simulation of transport in the III-nitride wurtzite materials done by *M. Farahmand, et al.*.

$$\mu(F) = \frac{\mu_{\text{low}} + \frac{v_{\text{sat}}}{F} \left(\frac{F}{E_0}\right)^\beta}{1 + \gamma \left(\frac{F}{E_0}\right)^\alpha + \left(\frac{F}{E_0}\right)^\beta}$$

where  $\mu_{\text{low}}$  is low-field mobility,  $v_{\text{sat}}$  is saturation velocity,  $F$  is the driving force, and  $E_0$  is critical field. Parameters  $\alpha$ ,  $\beta$ ,  $\gamma$  and  $v_{\text{sat}}$  are defined independently for holes and electrons.

## Eastman-Tiwari-Shur

A model based on a modified theory of the high-field domains which takes into account the field dependent diffusion by *L. F. Eastman, et al.* for GaAs MESFETs. Where  $E_s \equiv \frac{v_{\text{sat}}}{\mu_{\text{low}}}$  after work of *J. Chillieri, et al.*.

$$\mu(F) = \frac{\mu_{\text{low}} + \frac{v_{\text{sat}}}{F} \alpha \left(\frac{\mu_{\text{low}} F}{v_{\text{sat}}}\right)^\beta}{1 + \alpha \left(\frac{\mu_{\text{low}} F}{v_{\text{sat}}}\right)^\beta}$$

where  $\mu_{\text{low}}$  is low-field mobility,  $v_{\text{sat}}$  is saturation velocity, and  $F$  is the driving force. Parameters  $\alpha$ ,  $\beta$  and  $v_{\text{sat}}$  are defined independently for holes and electrons. The driving force  $F$  of the respective carriers is evaluated as the gradient of the respective quasi-Fermi level.

Parameters  $\alpha$  and  $\beta$  can be replaced introducing four other parameters  $E_{\text{peak}}$ ,  $E_{\text{mid}}$ ,  $v_{\text{peak}}$ , and  $v_{\text{mid}}$ , all related to the shape of the drift velocity function of the driving force. See *J. Chillieri, et al.* for reference.

$$\beta = \frac{\log\left(\frac{E_{\text{mid}}\mu_{\text{low}} - v_{\text{mid}}}{E_{\text{peak}}\mu_{\text{low}} - v_{\text{peak}}} \cdot \frac{v_{\text{peak}} - v_{\text{sat}}}{v_{\text{mid}} - v_{\text{sat}}}\right)}{\log\left(\frac{E_{\text{mid}}}{E_{\text{peak}}}\right)}$$

$$\alpha = \frac{E_{\text{peak}}\mu_{\text{low}} - v_{\text{peak}}}{v_{\text{peak}} - v_{\text{sat}}} \left(\frac{v_{\text{sat}}}{E_{\text{peak}}\mu_{\text{low}}}\right)^\beta$$

Last update: nnnn/nn/nn

## 2.10 Hamiltonian: 8-band model for zincblende

- *The Model*
- *Offsets*
- *Deformation potentials*
- *k.p parameters*
  - *Default settings*
  - *Luttinger parameters and electron effective mass*
  - *Rescaling S*
- *Zeeman Term*

### 2.10.1 The Model

Our implementation of the 8-band  $\mathbf{k} \cdot \mathbf{p}$  model for bulk crystals is a simplified version of the matrix Hamiltonian described in a PhD thesis [AndlauerPhD2009] obtained from the one-particle Hamiltonian

$$\hat{H} = \frac{\hat{\mathbf{p}}^2}{2m} + V_0(\mathbf{r}) + \frac{\hbar}{4m^2c^2} [\hat{\sigma} \times \nabla V_0(\mathbf{r})] \circ \hat{\mathbf{p}} \quad (2.10.1.1)$$

The description below contains also definitions and relations that can be found in [BirnerPhD2011] and [Bahder-PRB1990].

#### Warning

The Hamiltonian below does not contain terms related to the presence of the magnetic field. Therefore, **proper operator ordering** is neglected to keep formulas as simple as possible. Also, parameters  $N^+$ ,  $N^-$ ,  $\kappa$ , and  $g$  are not included here. Comprehensive documentation will be published elsewhere.

Our model is expressed in a basis of class  $\mathcal{A}$  functions:

$$\{|s \uparrow\rangle, |s \downarrow\rangle, |x_1 \uparrow\rangle, |x_2 \uparrow\rangle, |x_3 \uparrow\rangle, |x_1 \downarrow\rangle, |x_2 \downarrow\rangle, |x_3 \downarrow\rangle\}$$

The Hamiltonian can be concisely written in a block form as follows.

$$\hat{\mathcal{H}}_{\mathbf{k}\cdot\mathbf{p}} = \begin{bmatrix} \hat{\mathcal{H}}_{cc}(\mathbf{k}, \hat{\epsilon}) & 0 & \hat{\mathcal{H}}_{cv}(\mathbf{k}) & 0 \\ 0 & \hat{\mathcal{H}}_{cc}(\mathbf{k}, \hat{\epsilon}) & 0 & \hat{\mathcal{H}}_{cv}(\mathbf{k}) \\ \hat{\mathcal{H}}_{vc}(\mathbf{k}) & 0 & \hat{\mathcal{H}}_{vv}(\mathbf{k}) + \hat{\mathcal{H}}_{vv}(\hat{\epsilon}) + \hat{\mathcal{H}}_{so\uparrow\uparrow} & \hat{\mathcal{H}}_{so\uparrow\downarrow} \\ 0 & \hat{\mathcal{H}}_{vc}(\mathbf{k}) & \hat{\mathcal{H}}_{so\downarrow\uparrow} & \hat{\mathcal{H}}_{vv}(\mathbf{k}) + \hat{\mathcal{H}}_{vv}(\hat{\epsilon}) + \hat{\mathcal{H}}_{so\downarrow\downarrow} \end{bmatrix}$$

where  $\mathbf{k}$  is a wave vector and  $\hat{\epsilon}$  is a strain tensor.

Diagonal elements for the conduction band are defined as

$$\hat{\mathcal{H}}_{cc}(\mathbf{k}, \hat{\epsilon}) = E_c + A_c k^2 + a_c \text{Tr}\{\hat{\epsilon}\},$$

where  $k$  is length of the wave vector,  $E_c$  is conduction-band edge,  $a_c$  is absolute hydrostatic deformation potential for the conduction band,  $\text{Tr}\{\hat{\epsilon}\}$  is trace of the strain tensor,  $A_c$  is defined as

$$A_c = A' + \frac{\hbar^2}{2m_0}.$$

$A'$  is one of Kane parameters. It contains interactions between the conduction band and the remote bands  $\mathcal{B}$  with  $\Gamma_5$  symmetry

$$A' = \frac{\hbar^2}{m_0^2} \sum_{nj}^{\mathcal{B}} \frac{|\langle s | \hat{p}_1 | n\Gamma_5 j \rangle|^2}{E_c - E_{n,\Gamma_5}}.$$

Blocks introducing interaction between conduction and valence bands are given by

$$\hat{\mathcal{H}}_{cv}(\mathbf{k}) = [\imath P_0 k_1 + B k_2 k_3 \quad \imath P_0 k_2 + B k_1 k_3 \quad \imath P_0 k_3 + B k_1 k_2]$$

and

$$\hat{\mathcal{H}}_{vc}(\mathbf{k}) = \begin{bmatrix} -\imath P_0 k_1 + B k_2 k_3 \\ -\imath P_0 k_2 + B k_1 k_3 \\ -\imath P_0 k_3 + B k_1 k_2 \end{bmatrix},$$

where  $k_1, k_2, k_3$  are three components of the wave vector of interest,  $P_0$  is a Kane parameter describing interactions between conduction band and valence bands within the  $\mathcal{A}$  basis

$$P_0 = -\imath \frac{\hbar}{m_0} \langle s | \hat{p}_1 | x_1 \rangle,$$

and  $B$  is a Kane parameter including interaction between the all the bands in class  $\mathcal{A}$  and remote bands  $\mathcal{B}$  of  $\Gamma_5$  symmetry

$$B = 2 \frac{\hbar^2}{m_0^2} \sum_{nj}^{\mathcal{B}} \frac{\langle s | \hat{p}_1 | n\Gamma_{5j} \rangle \langle n\Gamma_{5j} | \hat{p}_1 | x_3 \rangle}{[E_c + E_v]/2 - E_{n,\Gamma_5}}.$$

with top valence band energy  $E_v = E_{v,av} + \Delta_0$ .

Blocks for the valence bands without the strain included are defined as

$$\hat{\mathcal{H}}_{vv}(\mathbf{k}) = \begin{bmatrix} E_{v,av} + \frac{\hbar^2}{2m_0} k^2 & 0 & 0 \\ 0 & E_{v,av} + \frac{\hbar^2}{2m_0} k^2 & 0 \\ 0 & 0 & E_{v,av} + \frac{\hbar^2}{2m_0} k^2 \end{bmatrix} + \begin{bmatrix} L'k_1^2 + Mk_2^2 + Mk_3^2 & N'k_1k_2 & N'k_1k_3 \\ N'k_1k_2 & Mk_1^2 + L'k_2^2 + Mk_3^2 & N'k_2k_3 \\ N'k_1k_3 & N'k_2k_3 & Mk_1^2 + Mk_2^2 + L'k_3^2 \end{bmatrix},$$

where  $E_{v,av}$  is average energy of valence bands at  $\Gamma$  point,  $M$ ,  $N'$ , and  $L'$  are Kane parameters introducing interactions between the valence bands in  $\mathcal{A}$  and remote bands  $\mathcal{B}$  of  $\Gamma_1$ ,  $\Gamma_3$ ,  $\Gamma_4$ ,  $\Gamma_5$  symmetries

$$\begin{aligned} M &= H_1 + H_2 \\ N' &= F' - G + H_1 - H_2 \\ L' &= F' + 2G \end{aligned}$$

where

$$\begin{aligned} G &= \frac{\hbar^2}{2m_0^2} \sum_{nj}^{\mathcal{B}} \frac{|\langle x_1 | \hat{p}_1 | n\Gamma_{3j} \rangle|^2}{E_v - E_{n,\Gamma_3}} \\ F' &= \frac{\hbar^2}{2m_0^2} \sum_{nj}^{\mathcal{B}} \frac{|\langle x_1 | \hat{p}_1 | n\Gamma_{1j} \rangle|^2}{E_v - E_{n,\Gamma_1}} \\ H_1 &= \frac{\hbar^2}{2m_0^2} \sum_{nj}^{\mathcal{B}} \frac{|\langle x_1 | \hat{p}_1 | n\Gamma_{5j} \rangle|^2}{E_v - E_{n,\Gamma_5}} \\ H_2 &= \frac{\hbar^2}{2m_0^2} \sum_{nj}^{\mathcal{B}} \frac{|\langle x_1 | \hat{p}_1 | n\Gamma_{4j} \rangle|^2}{E_v - E_{n,\Gamma_4}} \end{aligned}$$

Spin-orbit interaction within the valence bands is introduced by

$$\hat{\mathcal{H}}_{so\uparrow\uparrow} = \frac{\Delta_0}{3} \begin{bmatrix} 0 & -i & 0 \\ i & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} = [\hat{\mathcal{H}}_{so\downarrow\downarrow}]^\dagger \quad \text{and} \quad \hat{\mathcal{H}}_{so\uparrow\downarrow} = \frac{\Delta_0}{3} \begin{bmatrix} 0 & 0 & 1 \\ 0 & 0 & -i \\ -1 & i & 0 \end{bmatrix} = [\hat{\mathcal{H}}_{so\downarrow\uparrow}]^\dagger,$$

where spin-orbit interaction energy  $\Delta_0$  is defined by

$$\frac{\Delta_0}{3} = -i \frac{\hbar}{4m_0^2 c^2} \langle x_1 | [\nabla V_0(\mathbf{r}) \times \hat{\mathbf{p}}]_2 | x_3 \rangle.$$

The strain is introduced to the valence bands by

$$\hat{\mathcal{H}}_{vv}(\hat{\epsilon}) = \begin{bmatrix} l\epsilon_{11} + m\epsilon_{22} + m\epsilon_{33} & n\epsilon_{21} & n\epsilon_{31} \\ n\epsilon_{21} & m\epsilon_{11} + l\epsilon_{22} + m\epsilon_{33} & n\epsilon_{32} \\ n\epsilon_{31} & n\epsilon_{32} & m\epsilon_{11} + m\epsilon_{22} + l\epsilon_{33} \end{bmatrix}$$

where  $\epsilon_{ij}$  are elements of the strain tensor  $\hat{\epsilon}$  and  $m$ ,  $n$ ,  $l$  are matrix elements of a strain-dependent interaction operator, further defining deformation potentials for the valence bands.

**Note**

All sections below may be moved elsewhere in near future

## 2.10.2 Offsets

$$E_c = E_g^{(db)} + E_{v,av}^{(db)} + \frac{1}{3}\Delta_0^{(db)}, \quad E_{v,av} = E_{v,av}^{(db)}, \quad \Delta_0 = \Delta_0^{(db)}$$

Where the following mapping to our database is applied.

Table 2.10.2.1: Mapping of offsets to the database

| parameter         | value in the database                                      |
|-------------------|------------------------------------------------------------|
| $E_g^{(db)}$      | database{ ..._zb{ conduction_bands{ Gamma{ bandgap } } } } |
| $E_{v,av}^{(db)}$ | database{ ..._zb{ valence_bands{ bandoffset } } }          |
| $\Delta_0^{(db)}$ | database{ ..._zb{ valence_bands{ delta_S0 } } }            |

**Attention**

If temperature dependence is triggered then the Varshni formula is applied to the energy gap such that  $E_c \rightarrow E_c(T)$  and  $E_{v,av} \rightarrow E_{v,av}(T)$ .

## 2.10.3 Deformation potentials

$$\begin{aligned} a_c &= a_c^{(db)}, \\ m &= a_v^{(db)} - b^{(db)}, \\ n &= \sqrt{3}d^{(db)}, \\ l &= a_v^{(db)} + 2b^{(db)}, \end{aligned}$$

Where the following mapping to our database is applied.

Table 2.10.3.1: Mapping of deformation potentials to the database

| parameter    | value in the database                                  |
|--------------|--------------------------------------------------------|
| $a_c^{(db)}$ | database{ ..._zb{ Gamma{ defpot_absolute } } }         |
| $a_v^{(db)}$ | database{ ..._zb{ valence_bands{ defpot_absolute } }   |
| $b^{(db)}$   | database{ ..._zb{ valence_bands{ defpot_uniaxial_b } } |
| $d^{(db)}$   | database{ ..._zb{ valence_bands{ defpot_uniaxial_d } } |

## 2.10.4 k.p parameters

**Attention**

In this section we assume that `rescale_S_to` is not defined in the input file at all, like in the examples below. The topic of rescaling S parameter and it's influence on the Hamiltonian will be discussed elsewhere.

As the  $\mathbf{k} \cdot \mathbf{p}$  models have been derived in the literature on numerous ways, there are couple of parameterisation standards available of which preference is not clear. Also, depending on the method applied to obtaining parameters some of them are easier accessible than the others. Therefore, depending on the source and the material of interest different schemes of parametrisation may be preferred by the user. For this purpose multiple possibilities of connecting our database to this model are available.



## Default settings

The default settings are equivalent to setting all the attributes `use_Luttinger_parameters`, `from_6band_parameters`, `approximate_kappa`, `evaluate_S` to no.

### Examples

1. Controlling parameters of the Hamiltonian for computation of electronic energy dispersion for a bulk crystal

```
classical{
  bulk_dispersion{
    KP8{
      from_6band_parameters = no
      use_Luttinger_parameters = no
      approximate_kappa = no
      evaluate_S = no
    }
  }
}
```

2. Controlling parameters of the Hamiltonian for which h Schrödinger equation is solved

```
quantum {
  region{
    kp_8band{
      kp_parameters{
        from_6band_parameters = no
        use_Luttinger_parameters = no
        approximate_kappa = no
        evaluate_S = no
      }
    }
  }
}
```

Then the Kane parameters are defined by

$$M = \frac{\hbar^2}{2m_0} M^{(\text{db})} \quad , \quad N' = \frac{\hbar^2}{2m_0} N'^{(\text{db})} \quad , \quad L' = \frac{\hbar^2}{2m_0} L'^{(\text{db})}$$

$$A_c = \frac{\hbar^2}{2m_0} S^{(\text{db})} \quad , \quad B = \frac{\hbar^2}{2m_0} B^{(\text{db})} \quad , \quad P = \sqrt{\frac{\hbar^2}{2m_0} E_p^{(\text{db})}}$$

where the following mapping to our database is applied.

Table 2.10.4.1: Mapping of Kane parameters to the database

| parameter           | value in the database                   |
|---------------------|-----------------------------------------|
| $M^{(\text{db})}$   | database{ ..._zb{ kp_8_bands{ M } } }   |
| $L'^{(\text{db})}$  | database{ ..._zb{ kp_8_bands{ L } } }   |
| $N'^{(\text{db})}$  | database{ ..._zb{ kp_8_bands{ N } } }   |
| $S^{(\text{db})}$   | database{ ..._zb{ kp_8_bands{ S } } }   |
| $B^{(\text{db})}$   | database{ ..._zb{ kp_8_bands{ B } } }   |
| $E_p^{(\text{db})}$ | database{ ..._zb{ kp_8_bands{ E_P } } } |

## Luttinger parameters and electron effective mass

One needs to set all three parameters `from_6band_parameters`, `use_Luttinger_parameters`, `evaluate_S` to yes to use the Luttinger parameters (as defined for 6-band  $\mathbf{k} \cdot \mathbf{p}$  model) and the effective mass of electrons.

### Examples

1. Controlling parameters of the Hamiltonian for computation of electronic energy dispersion for a bulk crystal

```
classical{
  bulk_dispersion{
    KP8{
      from_6band_parameters = yes
      use_Luttinger_parameters = yes
      approximate_kappa = no
      evaluate_S = yes
    }
  }
}
```

2. Controlling parameters of the Hamiltonian for which h Schrödinger equation is solved

```
quantum {
  region{
    kp_8band{
      kp_parameters{
        from_6band_parameters = yes
        use_Luttinger_parameters = yes
        approximate_kappa = no
        evaluate_S = yes
      }
    }
  }
}
```

Then the Kane parameters are defined by

$$\begin{aligned} M &= \frac{\hbar^2}{2m_0} \left[ -\gamma_1^{(\text{db})} + 2\gamma_2^{(\text{db})} - 1 \right] \\ N' &= \frac{\hbar^2}{2m_0} \left[ -6\gamma_3^{(\text{db})} \right] + \frac{E_p^{(\text{db})}}{E_g} \\ L' &= \frac{\hbar^2}{2m_0} \left[ -\gamma_1^{(\text{db})} - 4\gamma_2^{(\text{db})} - 1 \right] + \frac{E_p^{(\text{db})}}{E_g} \\ A_c &= \frac{\hbar^2}{2m_0} \left[ \frac{1}{m_e^{(\text{db})}} - \frac{2E_p^{(\text{db})}}{3E_g} - \frac{E_p^{(\text{db})}}{3 \left[ E_g + \Delta_0^{(\text{db})} \right]} \right] \\ B &= \frac{\hbar^2}{2m_0} B^{(\text{db})} \\ P &= \sqrt{\frac{\hbar^2}{2m_0} E_p^{(\text{db})}}, \end{aligned}$$

where the following mapping to our database is applied.

Table 2.10.4.2: Mapping to the database

| parameter         | value in the database                                   |
|-------------------|---------------------------------------------------------|
| $\gamma_1^{(db)}$ | database{ ..._zb{ kp_6_bands{ gamma_1 } } }             |
| $\gamma_2^{(db)}$ | database{ ..._zb{ kp_6_bands{ gamma_2 } } }             |
| $\gamma_3^{(db)}$ | database{ ..._zb{ kp_6_bands{ gamma_3 } } }             |
| $m_e^{(db)}$      | database{ ..._zb{ conduction_bands{ Gamma{ mass } } } } |
| $\Delta_0^{(db)}$ | database{ ..._zb{ valence_bands{ delta_SO } } }         |
| $E_p^{(db)}$      | database{ ..._zb{ kp_8_bands{ E_P } } }                 |
| $B^{(db)}$        | database{ ..._zb{ kp_8_bands{ B } } }                   |

## Rescaling S

One of ways to get rid of spurious solutions in quantum structures is to rescale  $S$  parameter to 0 or 1. The  $S$  defines  $A_c$  as

$$A_c = \frac{\hbar^2}{2m_0} S$$

### Examples

1. Rescaling  $S$  in the Hamiltonian for computation of electronic energy dispersion for a bulk crystal

```
classical{
  bulk_dispersion{
    KP8{
      rescale_S_to = 1
    }
  }
}
```

2. Rescaling  $S$  in the Hamiltonian for which the Schrödinger equation is solved

```
quantum {
  region{
    kp_8band{
      kp_parameters{
        rescale_S_to = 1
      }
    }
  }
}
```

The initial value of  $S$  is determined according to choices described before. If one chose `evaluate_S = no` then

$$S = S^{(db)},$$

otherwise, if one chose `evaluate_S = yes` then

$$S = \frac{1}{m_e^{(db)}} - \frac{2E_p^{(db)}}{3E_g} - \frac{E_p^{(db)}}{3[E_g + \Delta_0^{(db)}]}.$$

In the input file, one can request consistent rescaling the model such that  $S \rightarrow S^{(\text{new})}$  resulting in

$$A_c = \frac{\hbar^2}{2m_0} S^{(\text{new})}.$$

A rescaled Kane energy  $E_p^{(\text{new})}$  is evaluated to ensure that the model gives the same electronic band structure (ideally) as before the rescaling, but without spurious solutions. It is done directly from the assumption the  $S = S^{(\text{new})}$ .

$$E_p^{(\text{new})} = E_p^{(\text{db})} + \left[ S - S^{(\text{new})} \right] \frac{E_g \left[ E_g + \Delta_0^{(\text{db})} \right]}{E_g + \frac{2}{3} \Delta_0^{(\text{db})}}$$

After the rescaled Kane energy is evaluated, it is used to update or redefine other relevant Kane parameters entering the model.

$$\begin{aligned} L' &\rightarrow L' + \frac{E_p^{(\text{new})} - E_p^{(\text{db})}}{E_g}, \\ N' &\rightarrow N' + \frac{E_p^{(\text{new})} - E_p^{(\text{db})}}{E_g}, \\ P &= \sqrt{\frac{\hbar^2}{2m_0} E_p^{(\text{new})}}. \end{aligned}$$

Where  $L'$  and  $N'$  are initially evaluated according to the choices in the `kp_parameters{ }` group as described in previous sections.

## 2.10.5 Zeeman Term

Magnetic effects are included via the Zeeman term

$$\hat{H}_n = \frac{\mu_B}{2} \cdot \hat{\sigma} \cdot \hat{g}_n \cdot \mathbf{B}$$

for both electrons and holes. The  $\hat{g}_n$  for electrons is diagonal and can be replaced with a scalar

$$g_c = \frac{(E_c^\uparrow - E_c^\downarrow)}{\mu_B B}$$

Free electron has the g factor equal 2, while in the crystal, due to spin-orbit splitting is different and can be approximated by so-called Roth's formula.

$$g_c = 2 - \frac{2E_p \Delta_0}{3E_g (E_g + \Delta_0)}$$

Parameter `classical{ bulk_dispersion{ KP8{ correct_electron_gfactor } } }` controls evaluation of the  $g_c$  for the bulk model as follows.

If `correct_electron_gfactor` is negative, then

$$g_c = 2$$

and

$$\hat{H}_n = \frac{\mu_B}{2} \cdot \hat{\sigma} \cdot 2 \cdot \mathbf{B}.$$

If `correct_electron_gfactor = 0`, then

$$m_{eff} = S + P^2 * \frac{E_g + \frac{2}{3} \Delta_0}{E_g (E_g + \Delta_0)}$$

where,  $S$ ,  $P$ ,  $E_g$ , and  $\Delta_0$  are taken from the database or estimated as defined in `kp_parameters{ }` group. A new parameter  $P_{new}$  is defined

$$P_{new} = \sqrt{\left( m_{eff} - \frac{\hbar^2}{2m_0} \right) \frac{E_g (E_g + \Delta_0)}{E_g + \frac{2}{3} \Delta_0}}$$

and used to compute corrected g-factor as

$$g_s = (P_{new}^2 - P^2) \frac{2m_0}{\hbar^2} \frac{2\Delta_0}{3E_g(E_g + \Delta_0)}.$$

This g-factor is then used in the Zeeman term.

$$\hat{H}_n = \frac{\mu_B}{2} \cdot \hat{\sigma} \cdot (2 + g_s) \cdot \mathbf{B}$$

If `classical{ bulk_dispersion{ KP8{ rescale_kp_everywhere } } }` is set to `yes`, then the Kane parameters  $L$  and  $N^+$  are rescaled

$$L \rightarrow L + \frac{P_{new}^2 - P^2}{E_g}$$

$$N^+ \rightarrow N^+ + \frac{P_{new}^2 - P^2}{E_g}$$

which results in replacing the  $P$  with a new one

$$P \rightarrow P = P_{new}$$

to be used in the remaining part of the Hamiltonian.

If `correct_electron_gfactor` is positive, then the algorithm is the same as in the case of having it zero, with a change that the band gap has assigned the value of `correct_electron_gfactor`.

$$E_g \rightarrow E_g = G$$

at the very beginning. This assignment influences only the band gap used within this algorithm. The band gap in other parts of the model is included normally, based on the parameters from the database.

---

*Last update: 27/05/2025*

## 2.11 Interface Hamiltonian 8-band Zinc-Blende

- *The Model*
- *Implementation and usage*

### 2.11.1 The Model

The interface Hamiltonian is used to model the heterostructures with atomic sharp interfaces. This is an extension of the standard 8-band  $\mathbf{k} \cdot \mathbf{p}$  model (described [here](#)). The model, originally derived in [\[Kilpstein2010\]](#), includes  $l = 2$  perturbation terms, which have been omitted in our implementation, consistent with the approach in [\[Livneh-PRB2012\]](#).

The interface Hamiltonian in the block form is given by:

$$\hat{\mathcal{H}}_{interface} = \delta(z - z_i) \begin{bmatrix} D_S & 0 & 0 & \pi\beta \\ 0 & D_X & \pi\alpha & 0 \\ 0 & \pi\alpha & D_X & 0 \\ \pi\beta & 0 & 0 & D_Z \end{bmatrix}$$

where  $\delta$  is the Dirac delta function,  $z_i$  is the position of the interface,  $D_S$ ,  $D_X$ ,  $D_Z$ ,  $\alpha$ , and  $\beta$  are the interface Hamiltonian parameters,  $\pi$  is the parameter that can take values of  $\pm 1$ . Parameter  $\pi$  is used to distinguish different ordering of materials at the interface, for example InAs/GaSb and GaSb/InAs interfaces would have same interface parameters, but opposite  $\pi$  values.

### 2.11.2 Implementation and usage

The interface Hamiltonian is available only for 1D simulations. For the best results, ensure that there is a grid point at the interface position, for example like so:

```
grid{
  xgrid{
    ...
    line{ pos = $x_1          spacing = $x_interface } # material1
    line{ pos = $x_interface spacing = $x_2 }          # material2
    ...
  }
}
```

To add the interface Hamiltonian to the simulation, use the `interface` keyword in `quantum{ region{ kp8_band{ } } }` section of input file.

Example:

```
quantum{
  region{
    kp_8band{
      ...
      interface{
        position = $x_interface
        D_S = 1.0
        D_X = 2.0
        D_Z = 1.5
        alpha = 0.1
        beta = 0.1
        reverse = no
      }
      ...
    }
  }
}
```

`reverse=no` corresponds to  $\pi = 1$ , and `reverse=yes` corresponds to  $\pi = -1$ . Other interface parameters are expressed in  $eV \cdot nm$ .

One can add multiple interfaces to the simulation by adding multiple `interface` blocks to the input file. Example below has 2 interfaces with the same paramters, but different order of materials (common case for modelling of quantum well).

```
quantum{
  region{
    kp_8band{
      ...
      interface{
        position = $x_interface1
        D_S = 1.0
        D_X = 2.0
        D_Z = 1.5
        alpha = 0.1
        beta = 0.1
        reverse = no
      }
      interface{
        position = $x_interface2
```

(continues on next page)

(continued from previous page)

```

        D_S = 1.0
        D_X = 2.0
        D_Z = 1.5
        alpha = 0.1
        beta = 0.1
        reverse = yes
    }
}
}

```

To set up repeating interfaces, one can use the `array_x` keyword in the `interface` block.

```

quantum{
    region{
        kp_8band{
            ...
            interface{
                position = 10.0
                ... # parameters here
                array_x{
                    shift = 2.0
                    min = -2
                    max = 3
                }
            }
        }
    }
}

```

The above example will create 6 interfaces with the same parameters, at positions 6.0, 8.0, 10.0, 12.0, 14.0 and 16.0 nm.

*Last update: 23/01/2025*

## 2.12 Excitons

The exciton states are computed using the approach from [ChuangOpto1995]. The explanation below only covers the most important aspects of the model, for detailed derivation please refer to the [ChuangOpto1995] book. The exciton computation is only available for 1D systems.

### 2.12.1 Model

Assuming the effective mass approximation for the electron and hole, the Schrödinger equation for the exciton can be written as:

$$(H_e(\bar{r}_e) + H_h(\bar{r}_h) - \frac{e^2}{4\pi\epsilon|\bar{r}_e - \bar{r}_h|})\Phi(r_e, r_h) = E\Phi(r_e, r_h) \quad (2.12.1.1)$$

where  $H_e$  and  $H_h$  are the Hamiltonians for the electron and hole, respectively,  $\bar{r}_e$  and  $\bar{r}_h$  are the coordinates of the electron and hole, respectively,  $\epsilon$  is the dielectric constant, and  $E$  is the energy of the exciton.

The wavefunction of the exciton, formed by electron  $n$  and hole  $m$ , will be calculated in the form:

$$\Phi(r_e, r_h) = \exp(i\bar{K}_t\bar{R}_t)F(\rho, z_e, z_h) = \exp(i\bar{K}_t\bar{R}_t)\phi_{nm}(\rho)f_n(z_e)g_m(z_h) \quad (2.12.1.2)$$

where  $\vec{K}_t$  is the in-plane wavevector of the exciton,  $\vec{R}_t$  is the in-plane coordinate of the exciton,  $F(\rho, z_e, z_h)$  is the exciton envelope function,  $f_n(z_e)$  and  $g_m(z_h)$  are the single particle envelope wavefunctions of electron and hole in the growth direction. Then, the equation for the unknown  $\phi_{nm}(\rho)$  is given by:

$$\left(-\frac{\hbar^2}{2m_r}\nabla_\rho^2 - V_{nm}(\rho)\right)\phi_{nm}(\rho) = E_{binding}\phi_{nm}(\rho) \quad (2.12.1.3)$$

where  $m_r$  is the reduced mass of the exciton,  $E_{binding}$  is the binding energy of the exciton, and  $V_{nm}(\rho)$  is expressed as:

$$V_{nm} = \int dz_e |f_n(z_e)|^2 \int dz_h |g_m(z_h)|^2 \frac{e^2}{4\pi\epsilon_s(\rho^2 + |z_e - z_h|^2)} \quad (2.12.1.4)$$

The solution of the equation for  $\phi_{nm}(\rho)$  can be found variationally by minimizing the binding energy of the exciton. The form of the solution is assumed to be similar to 1S state of 2D hydrogen atom:

$$\phi(\rho) = \sqrt{\frac{2}{\pi}} \frac{1}{\lambda} \exp(-\rho/\lambda) \quad (2.12.1.5)$$

where  $\lambda$  is the variational parameter, which has an interpretation of exciton inplane Bohr radius. The variational parameter  $\lambda$  is determined by minimizing the binding energy of the exciton from equation (2.12.1.3).

## 2.12.2 Averaging model parameters

The model depends on dielectric constant  $\epsilon$ , effective masses of the electron and hole  $m_e$  and  $m_h$ , which are not constant in heterostructures. If not given in the input file, the volume averaged values of these parameters are used. For effective masses, density weighted average is also possible.

## 2.12.3 Excitons in multiband Hamiltonians

The computation of the exciton in the case of 8-band  $\mathbf{k} \cdot \mathbf{p}$  Hamiltonian is complicated by the fact, that the electron and hole Hamiltonians are no longer separable. In that case the equations derived from effective mass Hamiltonians are used, using wavefunctions computed with the 8-band Hamiltonian. As the effective masses are not longer parameters of the Hamiltonian, the effective masses used are computed from the parameters for the 8-band Hamiltonian:  $L, M, N, E_P, S, E_{gap}$ . The same approach is used for 6-band  $\mathbf{k} \cdot \mathbf{p}$  Hamiltonian, where the effective masses are computed from the parameters  $L, M, N$ .

## 2.13 Optoelectronic characterization

- *Current and Power*
  - *Photocurrent*
  - *Power*
  - *Absorbed-power*
  - *Emitted-power*
- *Efficiencies*
  - *IQE - internal quantum efficiency*
  - *RQE - volume quantum efficiency*

### 2.13.1 Current and Power

#### Photocurrent

Then the **photocurrent**  $I_{photo}$  is calculated as the summation of the integration of these “radiative” and “fixed”:

$$I_{photo} = e \cdot \left( \int d\mathbf{x} R_{rad,net}^{stim}(\mathbf{x}) + \int d\mathbf{x} R_{fixed}(\mathbf{x}) \right) \quad (2.13.1.1)$$



## Power

$$\sum_i V_{i\text{-th contact}} \cdot I_{i\text{-th contact}} \quad (2.13.1.2)$$

## Absorbed-power

$$\int dE d\mathbf{x} E \cdot G(E, \mathbf{x}) \quad (2.13.1.3)$$

where  $G(E, x)$  is the generation rate calculated.

## Emitted-power

$$\int dE dx E \cdot R_{\text{rad}}^{\text{spon}}(E, x) \quad (2.13.1.4)$$

## 2.13.2 Efficiencies

### IQE - internal quantum efficiency

is calculated as

$$\eta_{IQE} = \frac{I_{\text{photo}}}{I_{\text{total}}} \quad (2.13.2.1)$$

where  $I_{\text{total}}$  is the total injected current consisted of both electron and hole currents.

The electrical power and optical power are calculated and output in *power.dat*:

### RQE - volume quantum efficiency

, which is also called as **radiative quantum efficiency**, is calculated as

$$\eta_{VQE} = \frac{R_{\text{rad,net}}^{\text{stim}} + R_{\text{fixed}}}{R_{\text{total}}} \quad (2.13.2.2)$$

where  $R_{\text{total}} = R_{\text{rad,net}}^{\text{stim}} + R_{\text{fixed}} + R_{\text{Auger}} + R_{\text{SRH}}$  is the total recombination rate including both radiative and non-radiative recombination.

Both  $\eta_{IQE}$  and  $\eta_{VQE}$  agree if the electrons and holes injected into the active region are fully consumed up by the recombination there. However, if they are not consumed up,  $e \cdot R_{\text{total}} < I_{\text{charge}}$  and this results in  $\eta_{IQE1} > \eta_{IQE2}$

---

*Last update: 09/12/2024*



## TUTORIALS

Many of the *nextnano++* tutorials have equivalent input files for *nextnano*<sup>3</sup>. However, some features are only available in *nextnano*<sup>3</sup>, e.g. tight-binding, electrolytes, graphene and mobility calculations.

### 3.1 Band structure

#### 3.1.1 Empirical tight-binding $sp^3s^*$ band structure of GaAs, GaP, AlAs, InAs, C (diamond) and Si

The input files to be used are:

- *1D\_TightBinding\_bulk\_GaAs.in*
- *1D\_TightBinding\_bulk\_GaAs\_so.in*
- *1D\_TightBinding\_bulk\_Al0.3Ga0.7As.in*
- *1D\_TightBinding\_bulk\_GaP.in*
- *1D\_TightBinding\_bulk\_GaP\_so.in*
- *1D\_TightBinding\_bulk\_AlAs.in*
- *1D\_TightBinding\_bulk\_AlAs\_so.in*
- *1D\_TightBinding\_bulk\_C.in*
- *1D\_TightBinding\_bulk\_Si.in*
- *1D\_TightBinding\_bulk\_Ge.in*
- *1D\_TightBinding\_bulk\_InAs\_so.in*
- *1D\_TightBinding\_bulk\_AlSb\_so.in*
- *1D\_TightBinding\_bulk\_InSb\_so.in*
- *1D\_TightBinding\_bulk\_Al0.5In0.5Sb.in*

#### Empirical tight-binding $sp^3s^*$ band structure of GaAs and GaP

The empirical tight-binding model that is used here is based on the  $sp^3s^*$  Hamiltonian, i.e. the 10 x 10 matrix given in Table (A) of [\[VoglJPCS1983\]](#).

In addition, we include spin-orbit coupling leading to a 20 x 20 matrix. The additional terms arising due to spin-orbit coupling are given for instance on p. R5 of [\[CarloSST2003\]](#).

We note that nowadays much better theoretical methods are available for calculating the band structure of bulk materials. However, for educational purposes, the chosen  $sp^3s^*$  method should be sufficient.

In this tutorial, we calculate the bulk band structure of

- GaAs, GaP and AlAs *without* spin-orbit coupling using the parameters of [\[VoglJPCS1983\]](#) at T = 0 K
- GaAs, GaP and AlAs *including* spin-orbit coupling using the parameters of [\[KlimeckSM2000\]](#) at T = 300 K

## Input

The values for the tight binding parametrization have to be specified in the input file:

```
$numeric-control
...
!-----
! Tight-binding parameters for GaAs (values of [Klimeck]). The units are [eV].
!-----
!tight-binding-parameters = -3.53284d0      ! Esa (GaAs)
                                0.27772d0      ! Epa
                                -8.11499d0      ! Esc
                                4.57341d0      ! Epc
                                12.33930d0      ! Es_a
                                4.31241d0      ! Es_c
                                -6.87653d0      ! Vss
                                1.33572d0      ! Vxx
                                5.07596d0      ! Vxy
                                0d0            ! Vs_s_
                                2.85929d0      ! Vsa_pc
                                11.09774d0      ! Vsc_pa
                                6.31619d0      ! Vs_a_pc
                                5.02335d0      ! Vs_c_pa
                                0.32703d0  0.12000d0 ! Delta_so_a Delta_so_c
! Note: a = anion, c = cation
!      s_ = s*
```

For more information about the meaning of these parameters, refer to the above cited references.

## Output

The output of the calculated tight-binding band structure can be found in the following file: `TightBinding/BandStructure.dat`

The first column contains the number of the grid point in the Brillouin zone. These grid points run

- from L point to Gamma point (along Lambda)
- from Gamma point to X point (along Delta)
- from X point to the U, K points
- from U,K points to Gamma point (along Sigma)

The next columns are the eigenvalues of the tight-binding Hamiltonian in units of [eV] for each grid point in  $\mathbf{k} = (k_x, k_y, k_z)$  space.

The file `TightBinding/BandStructure_without_so.dat` contains the tight-binding band structure without spin-orbit coupling.

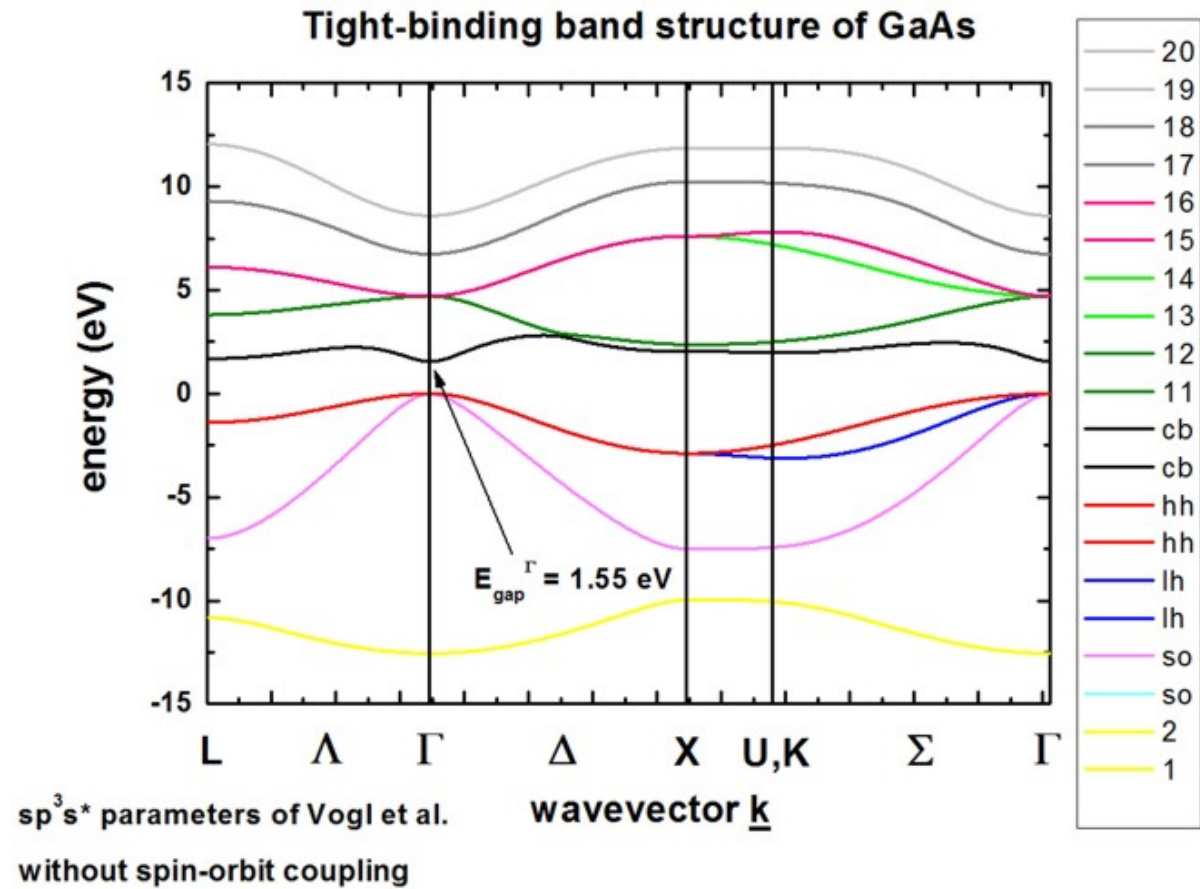
The file `TightBinding/k_vectors.dat` contains for each point the information to which k point it belongs to.

| no. | kx         | ky         | kz         | k          | kx [2pi/a] | ky [2pi/a] | kz [2pi/a] | k  [2pi/a]   |
|-----|------------|------------|------------|------------|------------|------------|------------|--------------|
| 1   | 0.314159E+ | 0.314159E+ | 0.314159E+ | 0.544140E+ | 0.500000E+ | 0.500000E+ | 0.500000E+ | 0.866025E+00 |

Note: Currently the units of  $k_x$ ,  $k_y$  and  $k_z$  do not take into account the lattice constant  $a$ . This should be modified. The values for  $k_x$ ,  $k_y$  and  $k_z$  in units of  $[2\pi/a]$  are correct, however. Another improvement would be to calculate and output the three-dimensional energy dispersion  $E(k_x, k_y, k_z)$  and two-dimensional slices  $E(k_x, k_z, 0)$  through the three-dimensional energy dispersion  $E(k_x, k_z, k_z)$  for a constant value of  $k_y$ , e.g.  $k_y = 0$ .

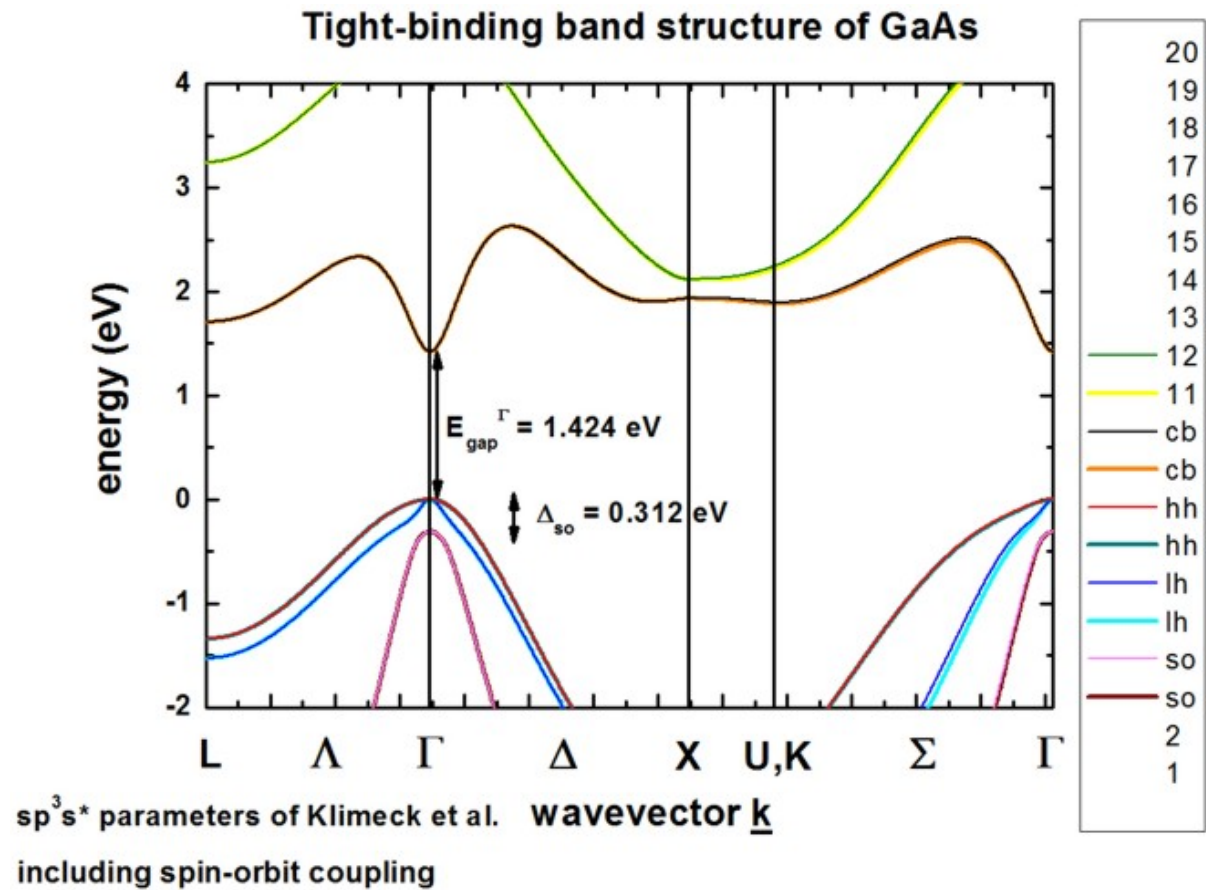
## Results

GaAs without spin-orbit coupling from 1D\_TightBinding\_bulk\_GaAs.in



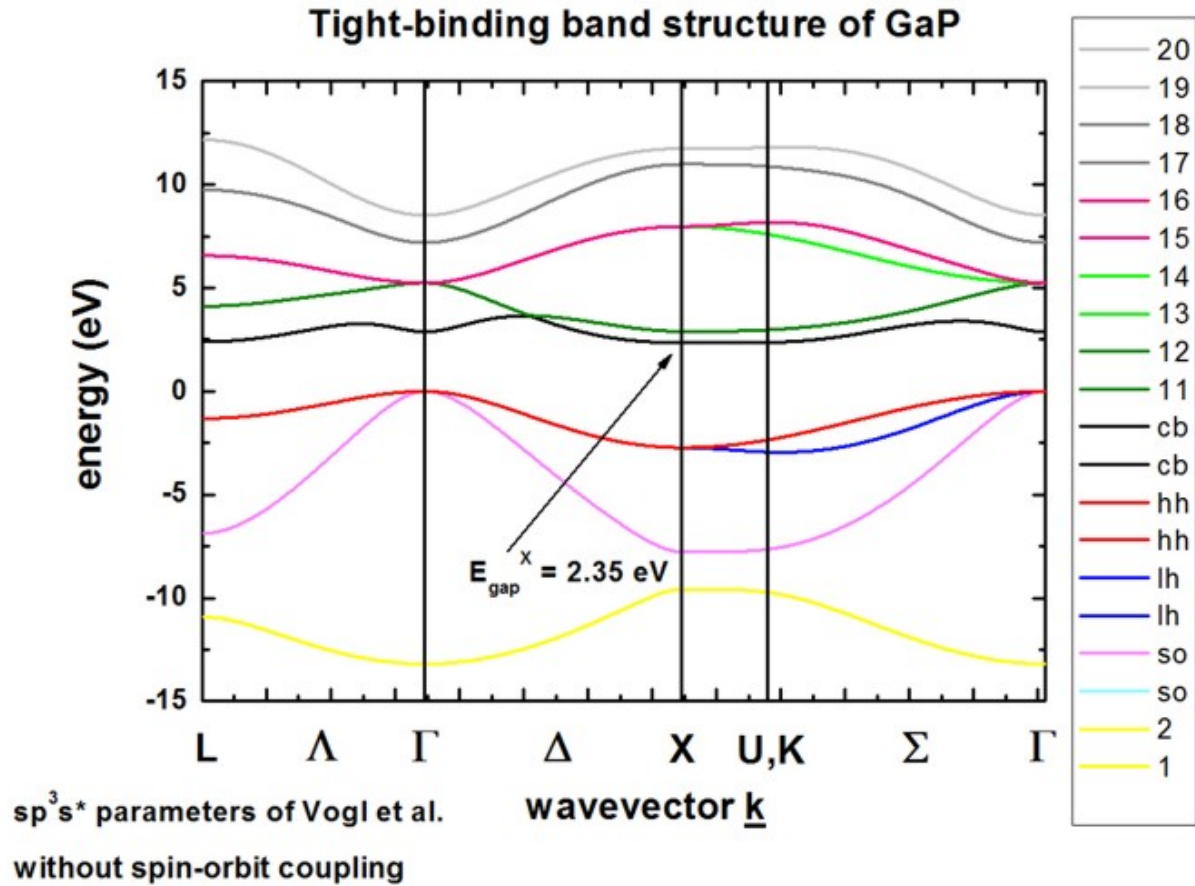
The calculated band structure is in excellent agreement with Fig. 11(d) of [VoglJPCS1983]. The conduction band minimum is at the Gamma point (direct band gap). Because spin-orbit coupling is not included in the Hamiltonian, the sp<sup>3</sup>s\* empirical tight-binding parameters were taken from [VoglJPCS1983] at T = 0 K.

GaAs including spin-orbit coupling from 1D\_TightBinding\_bulk\_GaAs\_so.in



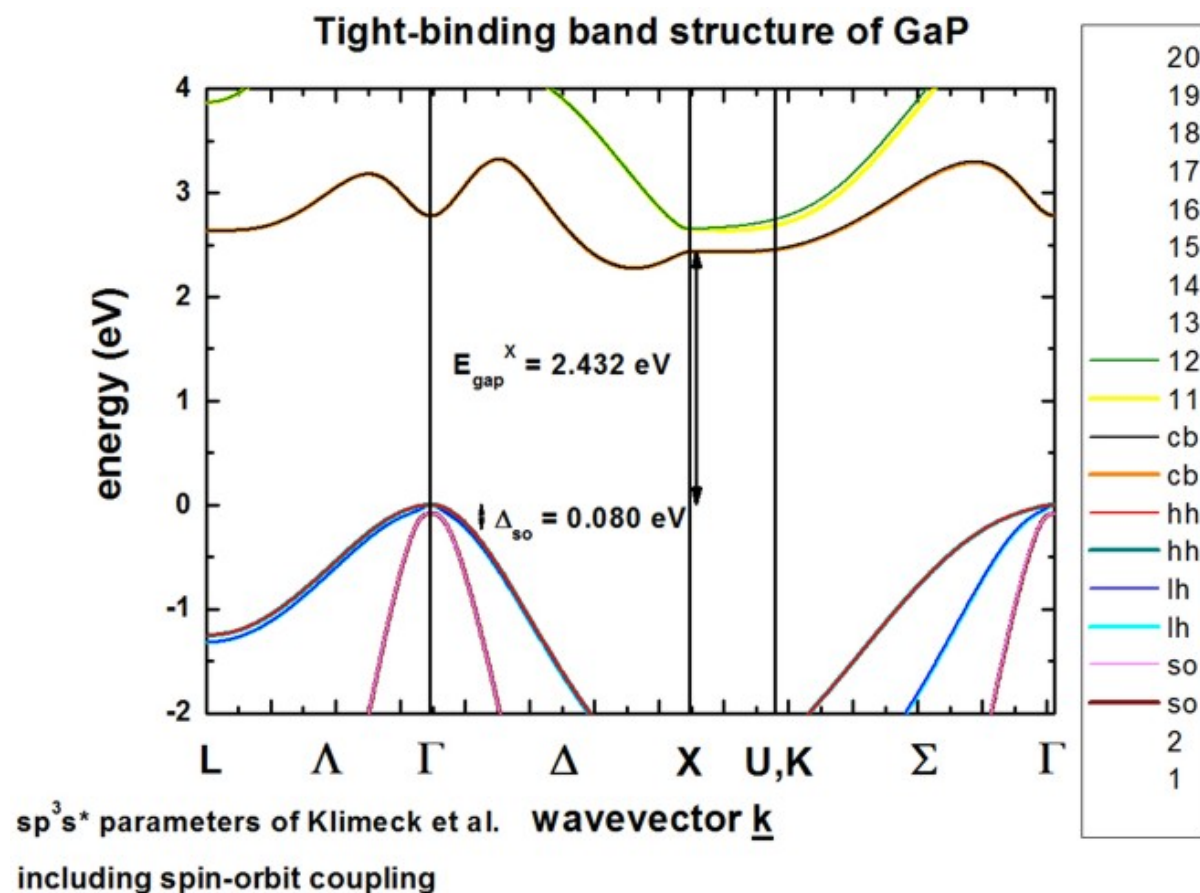
The calculated band structure is in excellent agreement with Fig. 1 of [KlimeckSM2000]. The conduction band minimum is at the Gamma point (irect band gap). Spin-orbit coupling lifts the degeneracy of heavy/light hole and split-off hole at the Gamma point. Heavy and light hole are still degerate at the Gamma point. The sp<sup>3</sup>s\* empirical tight-binding parameters were taken from [KlimeckSM2000] at T = 300 K.

**GaP without spin-orbit coupling** from 1D\_TightBinding\_bulk\_GaP.in



The calculated band structure is in excellent agreement with Fig. 2 of [VoglJPCS1983]. The conduction band minimum is calculated to be at the X point (indirect band gap). Because spin-orbit coupling is not included in the Hamiltonian, heavy, light and the split-off hole are degenerate at the Gamma point, i.e. at  $\mathbf{k} = (k_x, k_y, k_z) = 0$ . The sp<sup>3</sup>s\* empirical tight-binding parameters were taken from [VoglJPCS1983] at T = 0 K.

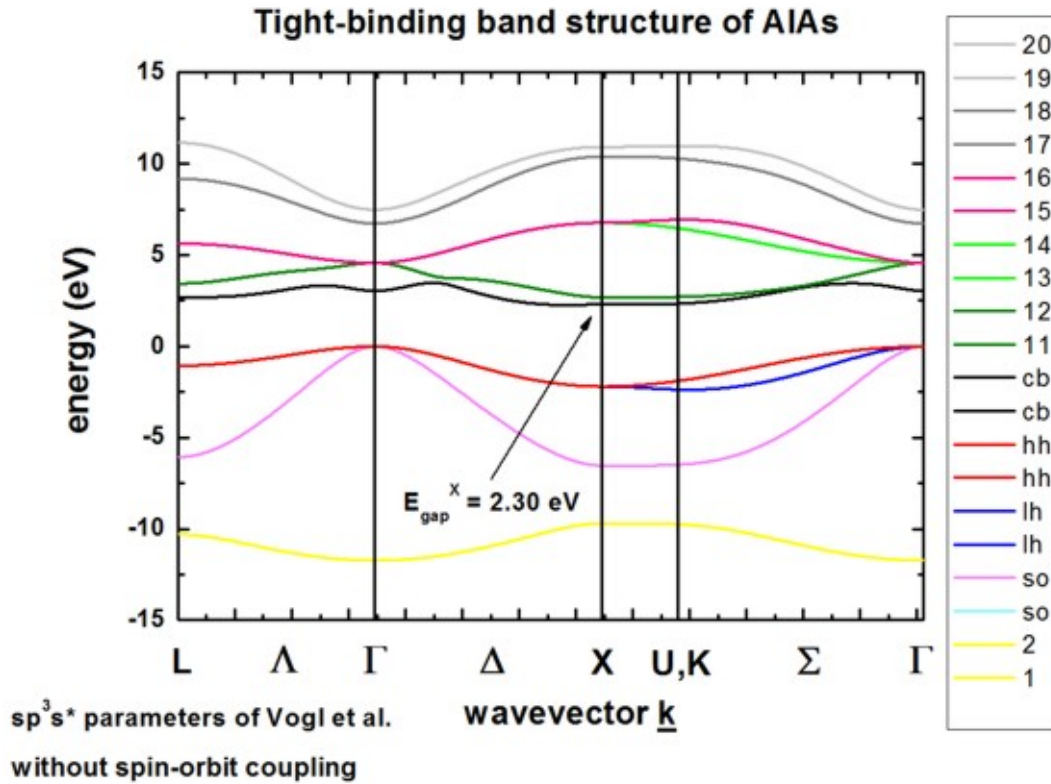
**GaP including spin-orbit coupling** from 1D\_TightBinding\_bulk\_GaP\_so.in



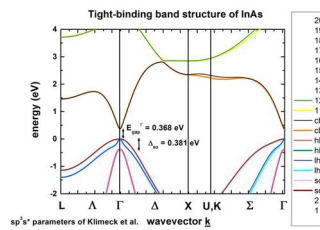
The calculated band structure is in excellent agreement with Fig. 1 of [KlimeckSM2000]. The conduction band minimum is in the vicinity of the X point at the Delta line (indirect band gap), so-called *camel's back*. Spin-orbit coupling lifts the degeneracy of heavy/light hole and split-off hole at the Gamma point. Heavy and light hole are still degenerate at the Gamma point. The sp<sup>3</sup>s\* empirical tight-binding parameters were taken from [KlimeckSM2000] at T = 300 K.

AlAs without spin-orbit coupling from 1D\_TightBinding\_bulk\_AlAs.in

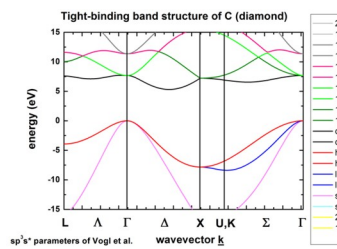




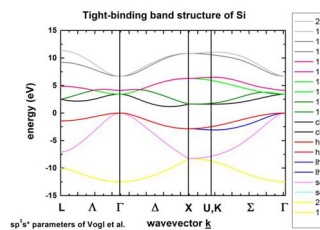
InAs including spin-orbit coupling from 1D\_TightBinding\_bulk\_InAs\_so.in



C (diamond) without spin-orbit coupling from 1D\_TightBinding\_bulk\_C.in



Si (silicon) without spin-orbit coupling from 1D\_TightBinding\_bulk\_Si.in



The  $k$  space resolution, i.e. the number of grid points on the axis of these plots can be adjusted. This can be done with:

```
$tighten
calculate-tight-binding-tighten = no
destination-directory           = TightBinding/
number-of-k-points              = 50      ! This number corresponds to 50 k_
↪points between the Gamma point and the X point      ! The number of k points along the_
↪other directions are scaled accordingly.
```

Author: Stefan Birner, Reinhard Scholz

### 3.1.2 Tight-binding band structure of graphene

The input file used is:

- *1D\_TightBinding\_graphene.in*

#### Nearest-neighbor tight-binding approximation

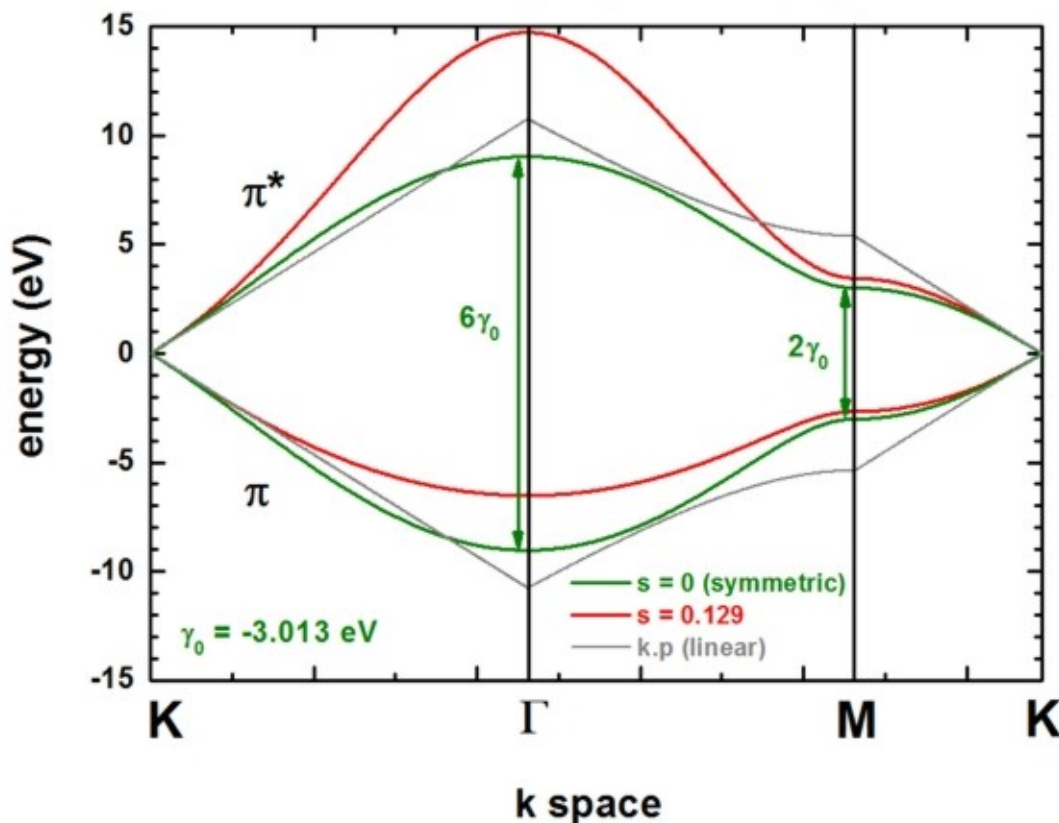
In this tutorial we calculate the bulk band structure of graphene which is a two-dimensional crystal (i.e. a monolayer of graphite) using a standard tight-binding approach. For more details, see for example the article [\[SaitoS2001\]](#)

The following figure shows the conduction band  $\pi^*$  (upper part) and valence band  $\pi$  (lower part) of graphene along special high-symmetry directions in the two-dimensional hexagonal Brillouin zone ( $\mathbf{k}$  space).

The high symmetry points that are used in this graph (from left to right) are:

- K:  $\mathbf{k} = (k_x, k_y) = (0, \frac{2}{3}) \times \frac{2\pi}{a}$
- Gamma:  $\mathbf{k} = (k_x, k_y) = (0, 0)$
- M:  $\mathbf{k} = (k_x, k_y) = (\frac{1}{3}^{1/2}, 0) \times \frac{2\pi}{a}$
- K':  $\mathbf{k} = (k_x, k_y) = (\frac{1}{3}^{1/2}, \frac{1}{3}) \times \frac{2\pi}{a}$

## Band structure of graphene (tight-binding)



Two lines correspond to the case where the  $s_0$  parameter is set to zero, i.e. in that case the dispersion of both  $\pi^*$  and  $\pi$  is the same (apart from the sign, i.e. they are symmetric with respect to the Fermi level  $E_F = 0$  eV). In this case the splitting energy at Gamma is three times as large as at the M point:

- splitting at Gamma:  $6\gamma_0$  (for  $s_0 = 0$ )
- splitting at M:  $2\gamma_0$  (for  $s_0 = 0$ )

Two lines correspond to the case where  $s_0 = 0.129$ .  $\pi^*$  and  $\pi$  are then nonsymmetric and are close to calculations from first principles and experimental data.

The data points are contained in the following file: `TightBinding/BandStructureGraphene.dat`. The first column contains integers which refer to the x axis (i.e. numbering of k points), the second column contains the eigenvalue of  $\pi^*$  in units of [eV] (conduction band), the third column contains the eigenvalue of  $\pi$  in units of [eV] (valence band). The file `TightBinding/k_vectors.dat` contains information about which integer corresponds to which  $k_x$  and  $k_y$  value.

The general formula for these lines read:

$$E_{+,-} = \frac{[E_{2p} \pm \Gamma_0 w(\mathbf{k})]}{1 \pm s_0 w(\mathbf{k})}$$

The parameters can be specified in the input file:

```
$numeric-control
...
(nearest-neighbor approximation, i.e. 3 parameters)
tight-binding-parameters = 0d0      ! [eV] E2p: site energy of the 2pz atomic orbital
↪(orbital energy)
                                -3.013d0 ! [eV] gamma0: C-C transfer energy (usually it
↪holds: -3 eV < gamma0 < -2.5 eV)
                                0.129d0 ! []   s0 = 0.129: denotes the overlap of the
```

(continues on next page)

(continued from previous page)

↪electronic wave function on adjacent sites  
 $s_0 = 0$ : denotes the overlap of the  
 ↪electronic wave function on adjacent sites  
 ! (usually it holds:  $s_0 < 0.1$ . Since this  
 ↪value is small, very often, it is neglected.)

Then there are two lines that are linear around the K point ( $\mathbf{k} \cdot \mathbf{p}$  approximation or linear expansion). Their linear dispersion is independent of the parameter  $s_0$ . Thus for small values of  $k$  (i.e. with respect to the K point), the energy dispersion can be approximated by a linear dispersion relation.

$$E(k) = E_{2p} \pm \hbar V_F |\mathbf{k}| = E_{2p} \pm 3^{1/2} \Gamma_0 \frac{ka}{2}$$

where

- $a$  is the latticer constant of graphene ( $a = 0.24612 \text{ nm}$ )
- the Fermi velocity of the charge varriers is given by  $V_F = 3^{1/2} |\Gamma| \frac{a}{2\hbar} = 0.98 \times 10^6 \text{ m s}^{-1} \simeq 0.003c$
- $c$  is the velocity of light

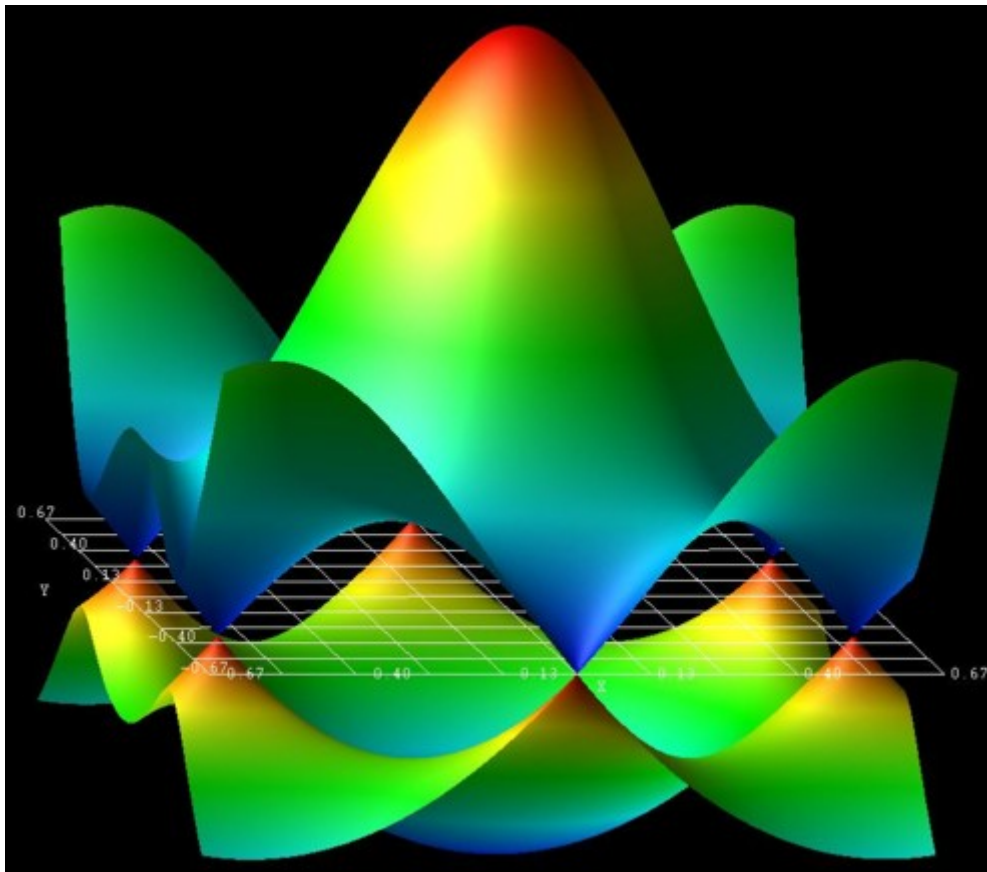
These data points are contained in this file: `TightBinding/BandStructureGraphene_kp.dat`

At the K point, the band gap is zero.

The following figure shows the energy dispersion  $E(k_x, k_y)$  of graphene for  $s_0 = 0.129$ . At the K points, the band gap is zero. The point in the middle is the Gamma point.  $k_x$  is from  $[-\frac{2}{3}, \frac{2}{3}] \frac{2\pi}{a}$ , the same holds for  $k_y$ .

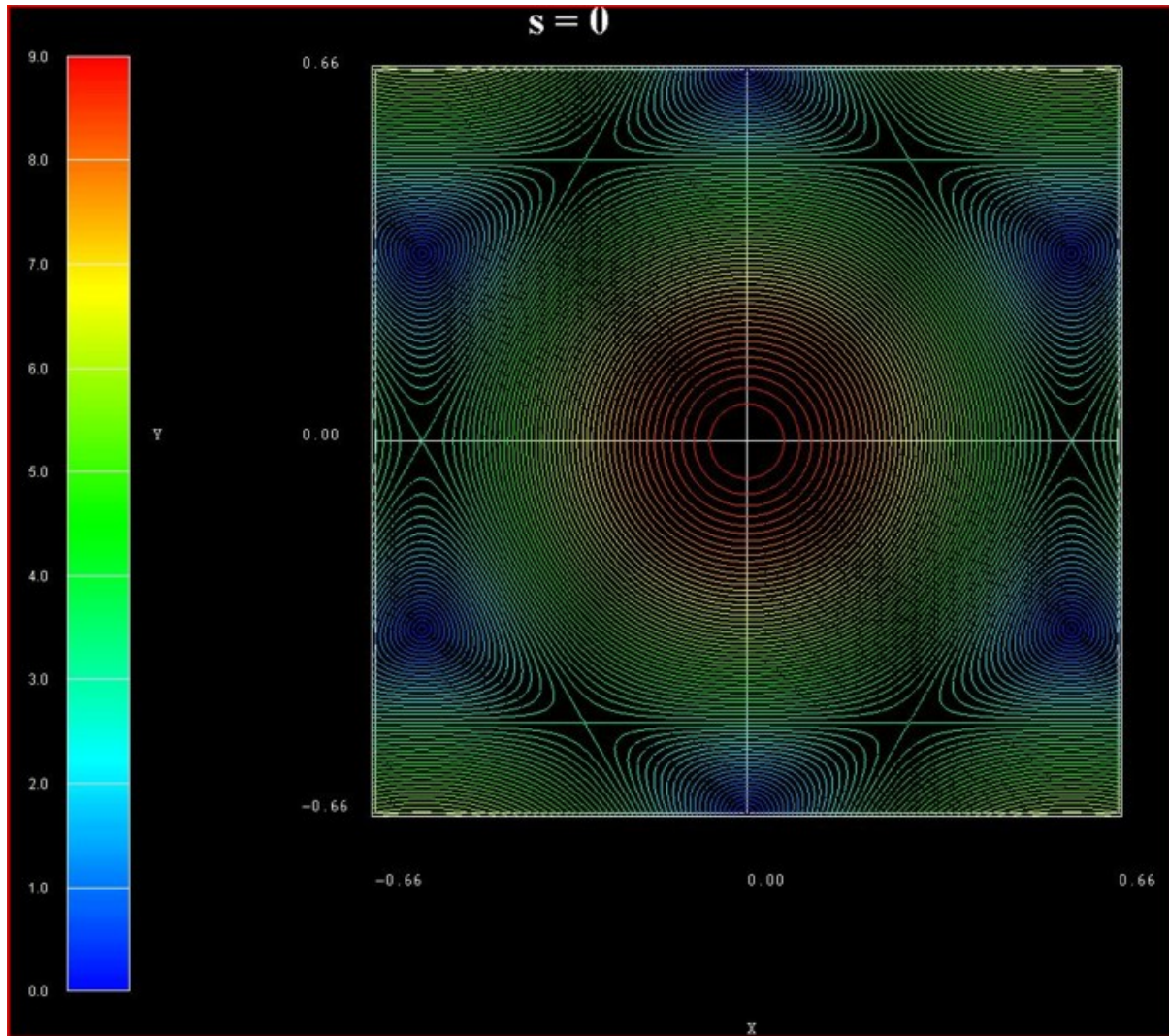
These data points are contained in the files:

- `TightBinding/BandStructureGraphene_cb.vtr`
- `TightBinding/BandStructureGraphene_vb.vtr`



The figure has been generated using the AVS/Express software.

The following figure shows the contour plot of the energy dispersion  $E(k_x, k_y)$  of graphene for  $s_0 = 0$  for the conduction band. (Note that the valence band dispersion is identical, apart from the sign, for  $s_0 = 0$ .) These data points are contained in the file: `TightBinding/BandStructureGraphene_cb.vtr`



The following figure shows the contour plot of the energy dispersion  $E(k_x, k_y)$  of graphene for  $s_0 = 0.129$  for the conduction band. These data points are contained in the file: `TightBinding/BandStructureGraphene_cb.vtr`

The following figure shows the contour plot of the energy dispersion  $E(k_x, k_y)$  of graphene for  $s_0 = 0.129$  for the valence band. These data points are contained in the file: `TightBinding/BandStructureGraphene_vb.vtr`

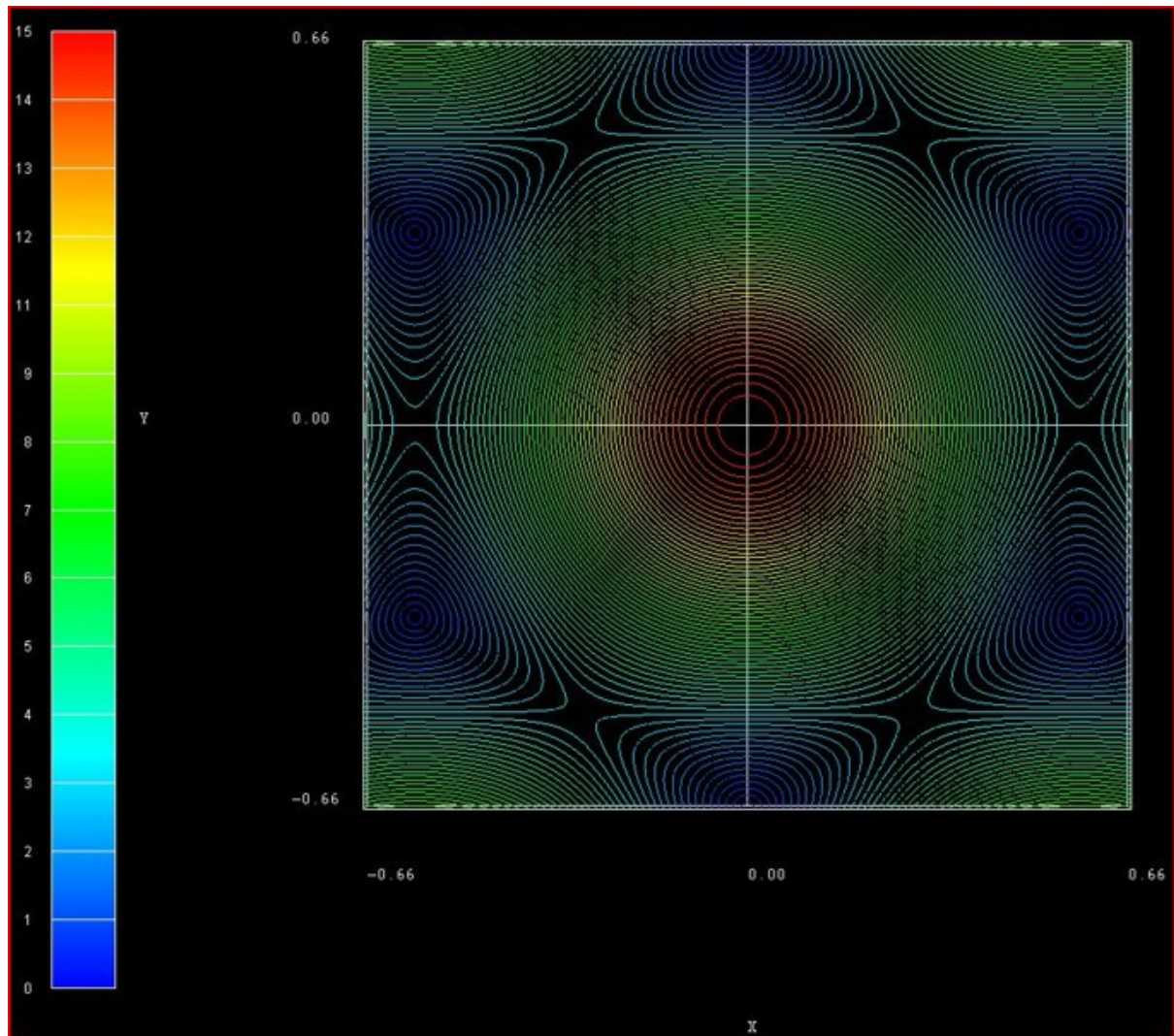
The following figure is the *nextnanomat* screenshot for the conduction band dispersion  $E(k_x, k_y)$ . The six dark areas correspond to the Dirac points in graphene.

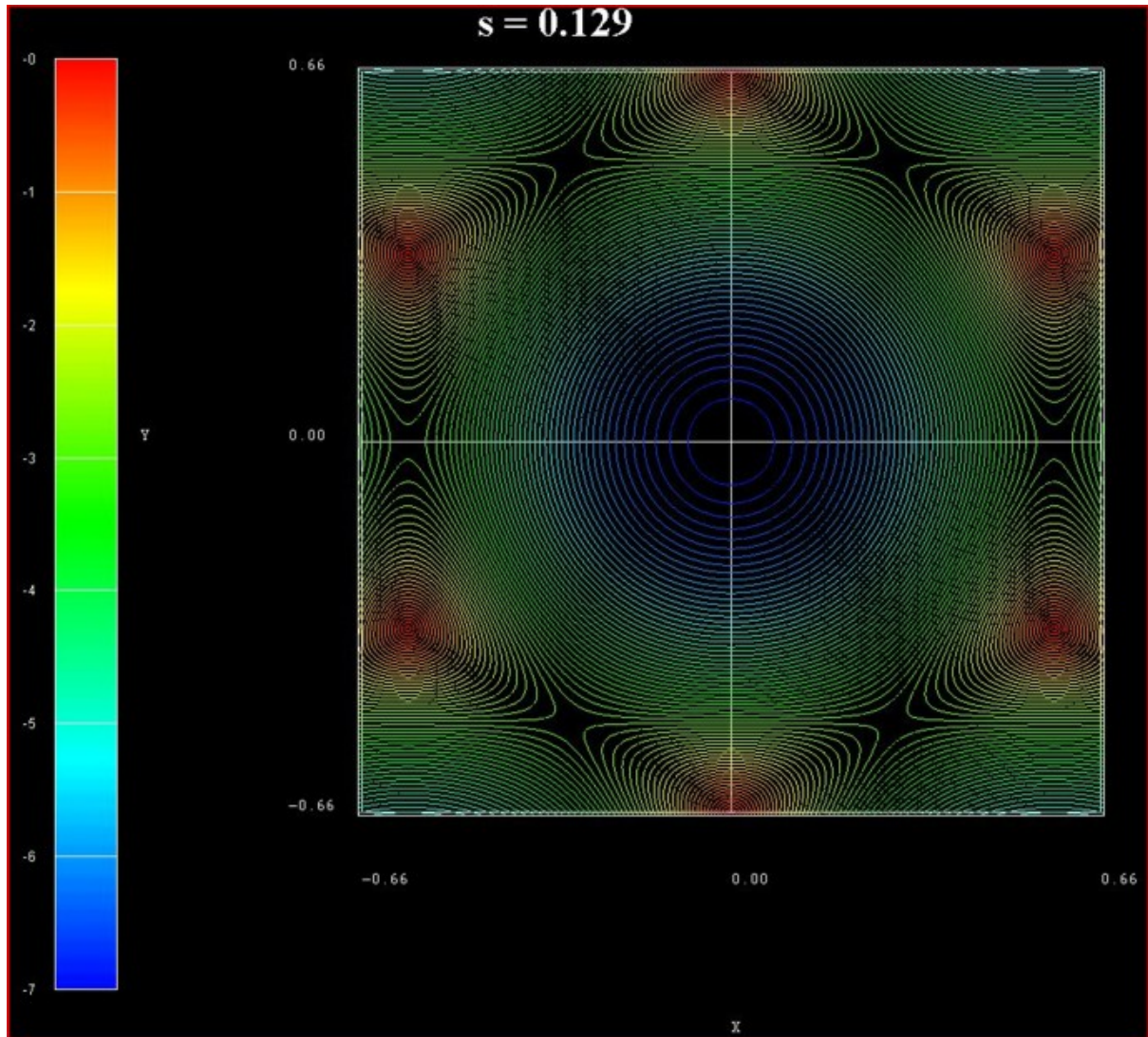
## Output Options

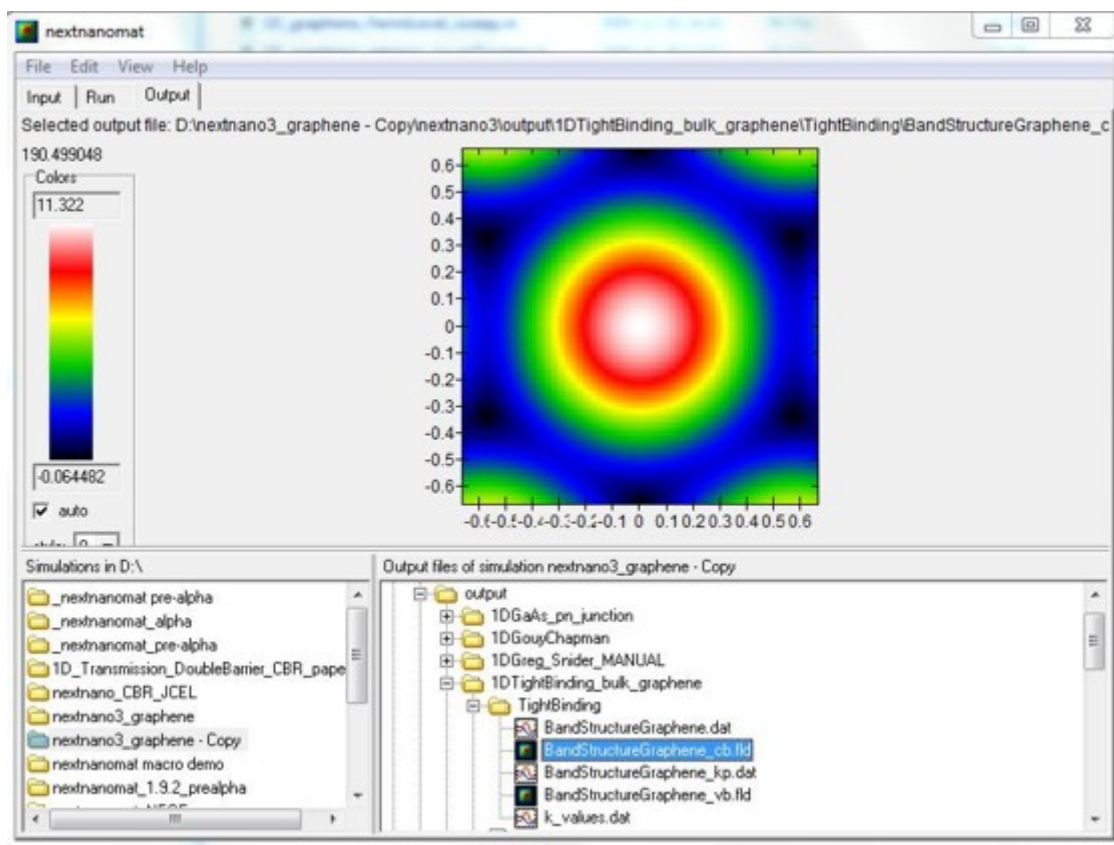
```
!-----!
$output-kp-data
...
bulk-kp-dispersion-3D = yes ! In this case, this refers to the bulk 2D tight-binding_
↪energy dispersion E(kx,ky).
                             ! If yes, then the two-dimensional energy dispersion_
↪E(kx,ky) is written out.
```

(continues on next page)









(continued from previous page)

**\$tighten**

destination-directory = TightBinding/

```
!-----!
! This parameter defines the resolution of the E(kx,ky) energy dispersion.
! E.g. if 'number-of-k-points = 100' then the kx gridding has a total of
! [-100,...,0,...,100] grid points,
! i.e. a total of 2 * 100 + 1 = 201 grid points along the kx direction.
! The same applies for ky direction.
!-----!
```

number-of-k-points = 100

! ==> 2 \*  $n$  $\rightarrow n + 1$ **Third-nearest-neighbor tight-binding approximation**

The following figure shows the band structure of graphene. All lines are identical to the ones shown already above with the exception of the blue lines which is the third-nearest-neighbor tight-binding approximation.

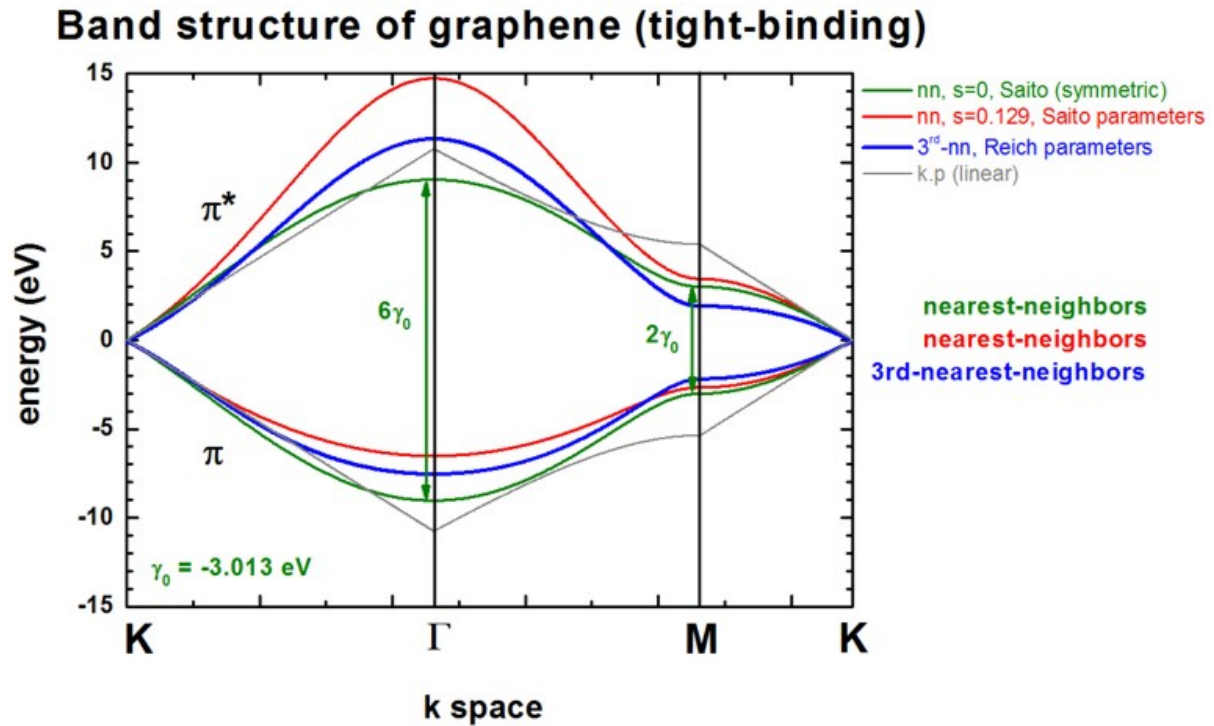
The third-nearest-neighbor tight-binding approximation is described in *ReichPR2002*

The following parameters have been used for the third-nearest neighbor approximation, i.e. 7 parameters:

```
tight-binding-parameters = -0.28d0 ! [eV] E2p: site energy of the 2pz atomic orbital
 $\rightarrow$  (orbital energy)
                        -2.97d0 ! [eV] gamma0: C-C transfer energy (nearest-
 $\rightarrow$  neighbor, nn)
                        0.073d0 ! [] s0: denotes the overlap of the electronic
```

(continues on next page)





(continued from previous page)

```

↪ wave function on adjacent sites (nn)
-0.073d0 ! [eV] gamma1: (2nd-nn)
0.018d0 ! [] s1: (2nd-nn)
-0.33d0 ! [eV] gamma2: (3rd-nn)
0.026d0 ! [] s2: (3rd-nn)

```

The parameters are taken from Reich et al. Note that the band gap at the K point is not exactly zero when using these parameters.)

### More Options

Inside the code several options, i.e. several algorithms, exist to setup the tight-binding Hamiltonian:

```

tight-binding-method = bulk-graphene-Saito-nn      ! nn = nearest-neighbor
                    = bulk-graphene-Scholz-nn      ! nn = nearest-neighbor
                    = bulk-graphene-Scholz-3rd-nn  ! 3rd-nn = third-nearest-neighbor
                    = bulk-graphene-Reich-3rd-nn   ! 3rd-nn = third-nearest-neighbor

```

Special option for third-nearest-neighbor tight-binding approximation in graphene:

```

tight-binding-calculate-parameter = no      ! (default) use 7 parameters (E2p, gamma0, ↪
↪ gamma1, gamma2, s0, s1, s2)
                                = E2p      ! use 6 parameters (gamma0, ↪
↪ gamma1, gamma2, s0, s1, s2)
                                = gamma1    ! use 6 parameters (E2p, ↪
↪ gamma0, gamma2, s0, s1, s2)

```

$E_{2p}$  or gamma1 can be calculated internally in order to force  $E(K) = 0 \text{ eV}$  where K is the K point in the Brillouin zone. In that case, only 6 parameters are used (although 7 parameters have to be present in the input file). The parameter to be calculated is simply ignored inside the code.

The k space resolution, i.e. the number of grid points on the axis of these plots can be adjusted.

```
$tighten
calculate-tight-binding-tighten = no           !
destination-directory           = TightBinding/
number-of-k-points              = 50           ! This corresponds to 50 k points.
↪ between the Gamma point and the M point.
                                   ! The number of k points along the
↪ other directions are scaled correspondingly.
```

### 3.1.3 30-band $k \cdot p$ band structure calculation

#### Note

This tutorial is under in development.

#### Attention

The 30-band model is under development in our tool, therefore related syntax and available functionalities are expected to be changing. Strain effects are not included yet.

#### Input Files:

*bulk\_kp\_dispersion\_Si\_SiGe\_Ge\_30band\_nn3.in*

#### Scope of the tutorial:

- Band structure of bulk Si, SiGe, and Ge within 30-band  $k \cdot p$

#### Relevant keywords:

- bulk-kp-dispersion
- kp-parameters
- k-vectors-sample-type

#### Main adjustable parameters in the input file:

- set of parameters - %30band\_parameters
- path of the band structure - %Bandstructure
- parameters controlling mole fraction of the modeled alloy %AlloySweepActive, %AlloySweepSize, %AlloySweepSteps

#### Relevant output Files:

- kp\_bulk\bulk\_kp30kp\_dispersion\*.dat (band structure)

## Introduction

This tutorial shows how to calculate band structure of  $\text{Si}_{1-x}\text{Ge}_x$  alloys within 30-band  $k \cdot p$  model. It answers the following questions:

- How to trigger 30-band model in *nextnano*<sup>3</sup>?
- How to define path on which you want to calculate the band structure?
- Which output file contain the band structure computed within the 30-band model?

The band structures presented in this tutorial are in agreement with those reported by *M. Cardona, F. H. Pollak* and *Rideau et al.* (except vicinity of the *K* point for the latter, which is under investigation.)

## The Input File

30-band  $k \cdot p$  model can be called to compute bulk dispersion by specifying **bulk-kp-dispersion = 30-band** inside the group **\$output-kp-data**.

```
$output-kp-data
bulk-kp-dispersion = 30-band
$end_output-kp-data
```

The path along which the band structure is computed can be specified inside the group **\$tighten** by assigning one of available paths (use autocomplete feature in *nextnanomat*) to the attribute **k-vectors-sample-type**.

```
$tighten
k-vectors-sample-type = L-Gamma-X-W-K-L-W-X-K-Gamma
$end_tighten
```

Parameters are currently hard-coded and available only for SiGe alloys. In this tutorial two sets of parameters are used to reproduce results from *M. Cardona, F. H. Pollak* and *Rideau et al.*. Switching between the different sets of parameters is done by setting **kp-parameters** to **kp-parameters = Rideau** or to **kp-parameters = Cardona-Pollak** inside the group **\$numeric-control**.

```
$numeric-control
kp-parameters = Rideau
$end_numeric-control
```

## Output files

The band structure generated after running the input file *bulk\_kp\_dispersion\_Si\_SiGe\_Ge\_30band\_nn3.in* can be found in a file *...kp\_bulkbulk\_kp30kp\_dispersion\_BrillouinZone1\_L-Gamma-X-W-K-L-W-X-K-Gamma.dat*, where the first column contains indexes of following wave vectors along the path and all the following columns contain eigenvalues starting with the highest ones. Exact coordinates of wave vectors for each point of the computed band structure can be found in a related file *...kp\_bulkbulk\_kp30kp\_dispersion\_BrillouinZone1\_L-Gamma-X-W-K-L-W-X-K-Gamma\_k\_vectors.dat*. There, the first column contains corresponding indexes of each k-point, the three following contain exact coordinates of each wave vector, and the last column stores the length of each wave vector.

## Results

### Cardona-Pollak parameters: 15 band Hamiltonian

In the paper of *M. Cardona and F. H. Pollak*, 15-band  $kp$  Hamiltonian is introduced to model band structures of Si and Ge. The 15-band Hamiltonian disregards spin-orbit interactions, resulting in spin degeneracy across all bands. For this tutorial, nextnano computes all 30 bands, while keeping spin-orbit couplings at zero.

To use the parameters from this paper, change **kp-parameters** to **Cardona-Pollak**

```
$numeric-control
kp-parameters = Cardona-Pollak
$end_numeric-control
```

The variable `%AlloyContent` in the input file can be adjusted to select between 0 (pure Germanium) and 1 (pure Silicon).

The resulting band structures for both pure Germanium and pure Silicon are illustrated in Figures [Figure 3.1.3.1](#), and [Figure 3.1.3.2](#), respectively.

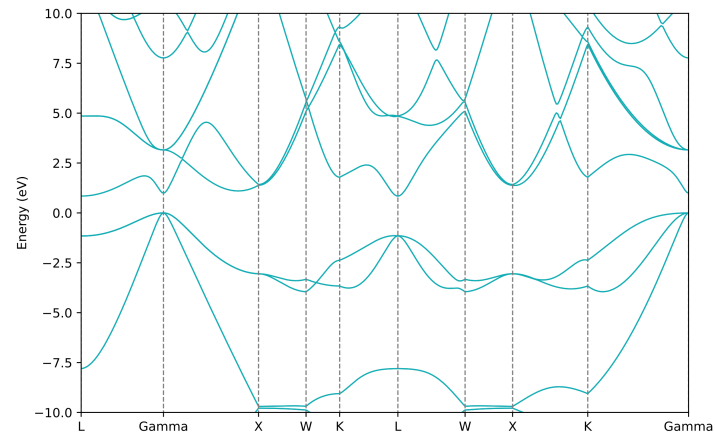


Figure 3.1.3.1: Band structure of Germanium (Ge) computed with Cardona-Pollak set of parameters , %AlloyContent=1.0

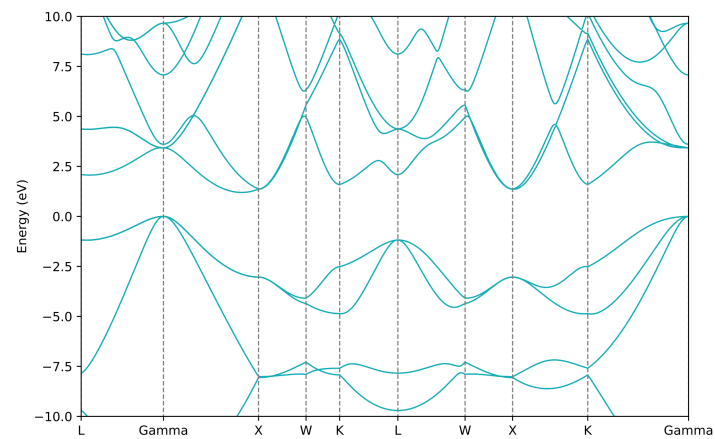


Figure 3.1.3.2: Band structure of Silicon (Si) computed with Cardona-Pollak set of parameters, %AlloyContent=0.0

### Rideau et al parameters.

In the paper of [Rideau et al.](#), the full set of parameters for 30-band Hamiltonian is given, including the spin-orbit couplings. The parameters are also proved to be valid for any alloy content in  $\text{Si}_{1-x}\text{Ge}_x$  alloys.

Band structures of pure Germanium, Silicon-Germanium alloy and pure Silicon generated within the input file are shown in the figures [Figure 3.1.3.3](#), [Figure 3.1.3.4](#) and [Figure 3.1.3.5](#), respectively.

To use the parameters from this paper, change **kp-parameters** to **Rideau**

```
$numeric-control
kp-parameters = Rideau
$end_numeric-control
```

The variable `%AlloyContent` should be adjusted accordingly in the input file to choose the material.

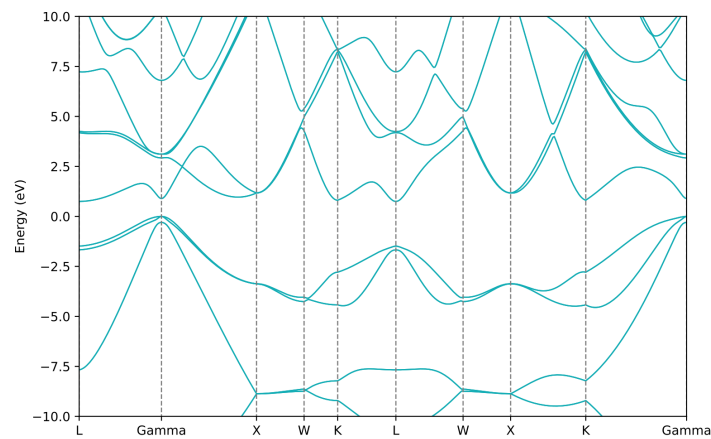


Figure 3.1.3.3: Band structure of Germanium (Ge), `%AlloyContent=1.0`, spin-orbit coupling included.

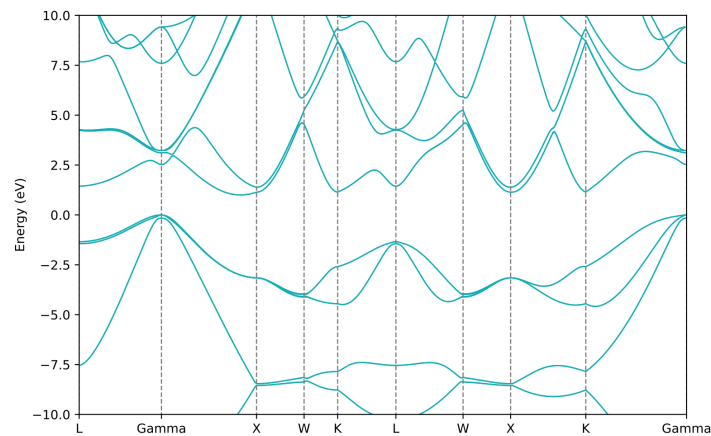


Figure 3.1.3.4: Band structure of Silicon Germanium ( $\text{Si}_{0.5}\text{Ge}_{0.5}$ ), `%AlloyContent=0.5`, spin-orbit coupling included.

## 3.2 Semiconductor Physics

### 3.2.1 Doping in semiconductors: Incomplete ionization

The input files used in this tutorial are:

- `1D_doped_bulk_Si_doping_sweep_Altermatt.in`

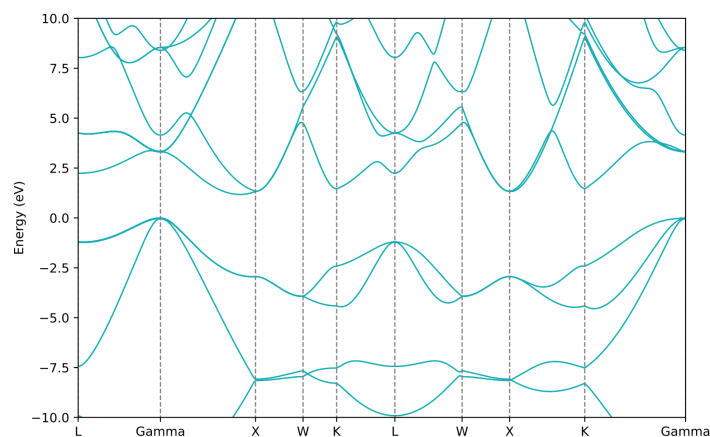


Figure 3.1.3.5: Band structure of Silicon (Si), %AlloyContent=0.0, spin-orbit coupling included.

#### Acknowledgment

This tutorial is based on the nextnano GmbH collaboration in the scope of the [SiPho-G Project](#) aiming at development of ultrahigh-speed optical components for next-generation photonic integrated circuits, and it is funded by the European Union's Horizon 2020 research and innovation program under grant agreement No 101017194.



**SiPho-G**  
Advanced GeSi components for next-generation silicon photonics applications

- `IncompleteIonization_Altermatt_nextnano3.py`

#### Table of Contents

##### Doping in semiconductors: Incomplete ionization in semiconductors (Altermatt model)

This tutorial tries to reproduce Fig. 5 of [\[AltermattI2006\]](#). We use the incomplete ionization model of Altermatt et al. of References [\[AltermattI2006\]](#) and [\[AltermattII2006\]](#). The relevant keywords and models are described in [\\$impurity-parameters](#).

We use one Python script and use nextnano.py to generate the following figure.

Our results differ from the original Altermatt paper. The minimum for Si:P (n-type) is close ( $\sim 0.78$  (nextnano) vs.  $\sim 0.76$  (Altermatt)) while Si:As (n-type) differs more ( $\sim 0.72$  (nextnano) vs.  $\sim 0.66$  (Altermatt)). The minimum for Si:B (p-type) also differs ( $\sim 0.61$  (nextnano) vs.  $\sim 0.73$  (Altermatt)). It seems that Altermatt uses the Maxwell–Boltzmann approximation for the density in eq. (7) of [\[AltermattII2006\]](#) while our implementation is using Fermi–Dirac statistics based on his eqs. (5a) and (5b) of [\[AltermattII2006\]](#). In fact, our model is not using his eqs. (6a), (6b) and (7), i.e. we are using the Fermi levels instead of the densities which might explain the discrepancy. It is not clear to us how his approach takes into account heavy holes, light holes and split-off holes as his eq. (7) indicates the usage of one type of holes only.

The parameters for the Altermatt model are specified as follows.

```
$doping-function
doping-function-number    = 1
impurity-number           = 1
doping-concentration       = 2.0      # 2.0 * [10^18 cm^-3]
only-region               = 0.0 20.0 # x_min x_max [nm]
$end_doping-function
```

(continues on next page)

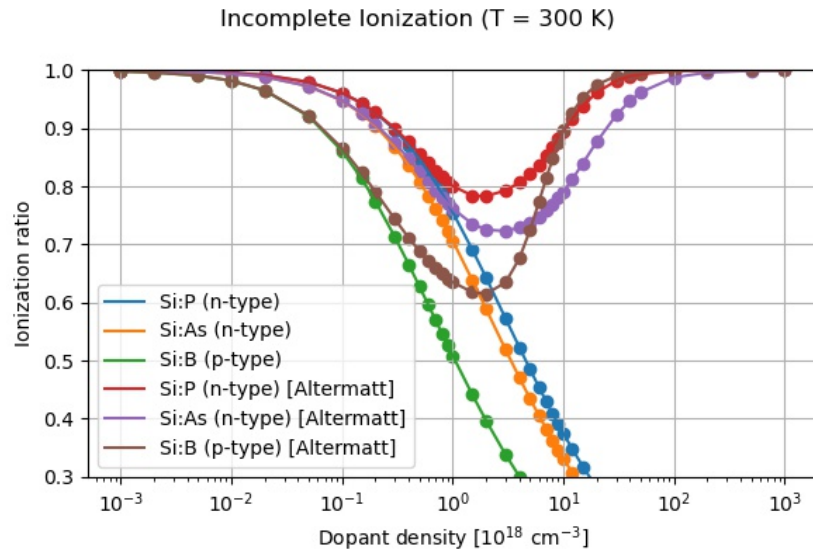


Figure 3.2.1.1: Ionization ratio of the donors P and As (n-type) and the acceptor B (p-type) in Si as a function of dopant density at room temperature. The figure compares our default model with the improved Altermatt model. At low dopant densities both models agree while for higher dopant densities the models differ substantially.

(continued from previous page)

```
$impurity-parameters
impurity-number           = 1
number-of-energy-levels    = 1
impurity-type             = n-type
degeneracy-of-energy-levels = 2
energy-levels-relative    = 0.0455 # n-P-in-Si [eV] ! [eV]
incomplete-ionization-N-ref = 2.2 # 2.2 * 10^18 cm^-3 [10^18 cm^-3]
incomplete-ionization-c    = 2.0 # 2 []
incomplete-ionization-N-b  = 6.0 # 6 * 10^18 cm^-3 [10^18 cm^-3]
incomplete-ionization-d    = 2.3 # 2.3 []
$end_impurity-parameters
```

If the 4 Altermatt specifiers (incomplete-ionization-N-ref, incomplete-ionization-c, incomplete-ionization-N-b, incomplete-ionization-d) are not present, the default model for incomplete ionization is used.

## 3.3 Heterostructures

### 3.3.1 Quantum Confined Stark Effect (QCSE)

Author: Stefan Birner

The input file used is:

- *1DQuantumConfinedStarkEffect\_nn3.in*
- *1DQuantumConfinedStarkEffect\_nnp.in*

This tutorial aims to reproduce Figure 3.22 (p. 96) of Paul Harrison’s excellent book “*Quantum Wells, Wires and Dots*” (Section 3.12 “Quantum Confined Stark Effect”), thus the following description is based on the explanations made therein. We are grateful that the book comes along with a CD so that we are able to look up the relevant material parameters and to check the results for consistency.



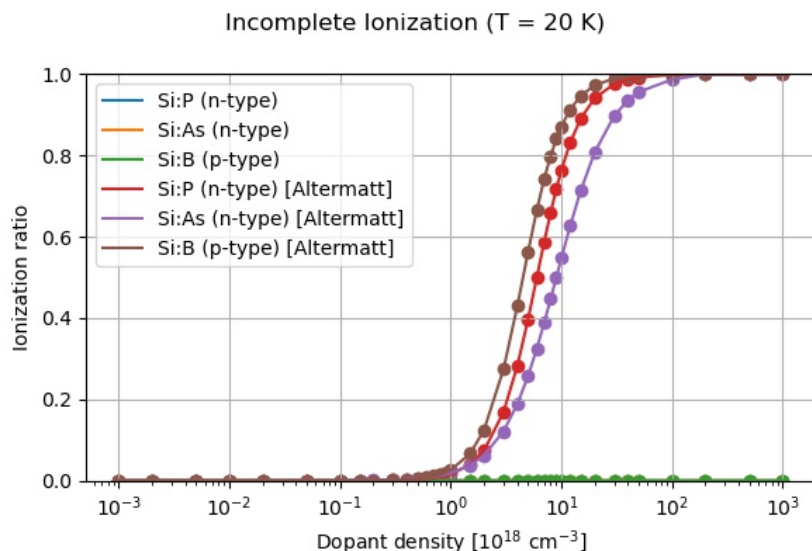


Figure 3.2.1.2: Ionization ratio of the donors P and As (n-type) and the acceptor B (p-type) in Si as a function of dopant density at T = 20 K. For low temperatures, only the Altermatt model leads to ionization at higher dopant densities.

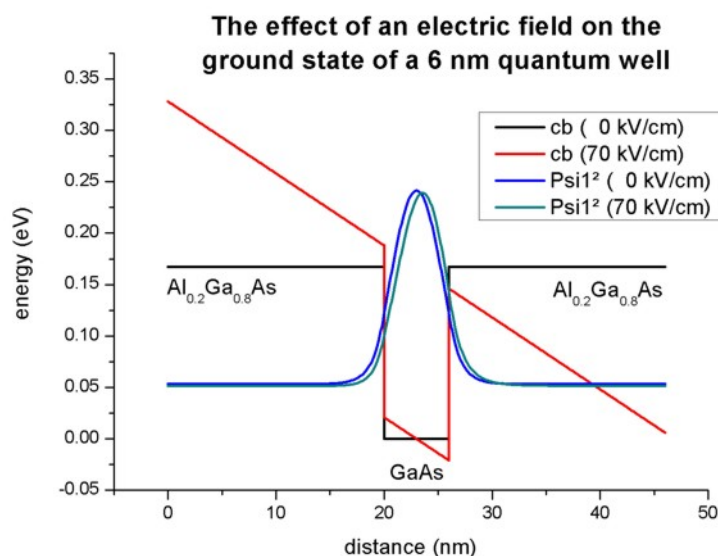
### Single quantum well: 20 nm AlGaAs / 6 nm GaAs / 20 nm AlGaAs

Our structure consists of a 6 nm GaAs quantum well that is surrounded by 20 nm Al<sub>0.2</sub>Ga<sub>0.8</sub>As barriers on each side. We thus have the following sequence: **20 nm** Al<sub>0.2</sub>Ga<sub>0.8</sub>As / 6 nm GaAs / **20 nm** Al<sub>0.2</sub>Ga<sub>0.8</sub>As. The barriers are printed in bold.

The figure shows the **conduction band edge** and the **square of the ground state electron wave function ( $\psi^2$ )** that is confined inside the well for two cases:

- No applied electric field (0 kV/cm)
- Applied electric field (-70 kV/cm)

In the case of an applied electric field, the wave function moves to the right and its ground state energy decreases slightly. The reason is that a charged particle prefers to move to areas of lower potential in order to lower the total energy. (Note that the energies were shifted so that the conduction band edge of GaAs equals 0 eV.) The origin of the electric field is chosen automatically to be the center of the well. This makes it possible to compare energies by varying the applied electric field as shown in this tutorial. For holes, the wave function would move to the left (not shown here) thus making it possible to produce a space charge or a polarization of the charge carriers.





## Technical Details

We use

```
$simulation-flow-control
flow-scheme = 21          ! apply constant electric field
...
```

and

```
$numeric-control
...
zero-potential = yes
```

This flow scheme includes the following:

1. We calculate the strain (if any)
2. We calculate the piezo and pyroelectric charges (if any)
3. **We do not solve Poisson's equation** (This is the difference with flow-scheme = 20)
4. We apply the electric field
5. We calculate the eigenstates and wave functions by solving Schrödinger's equation (either single-band or  $\mathbf{k} \cdot \mathbf{p}$ )

Note that in this case, this is not a self-consistent calculation of the Poisson-Schrödinger equation.

## Output

- a. The **conduction** band edge of the Gamma conduction band can be found at band\_structure / cb1D\_Gamma.dat
- b. This file contains the **eigenenergies** of the ground state. The units are in [eV] Schroedinger\_1band / ev\_cb1\_qc1\_sg1\_deg1.dat

For example, the values for zero applied electric field read:

|                      | num_ev | eigenvalue [eV]          |
|----------------------|--------|--------------------------|
| nextnano             | 1      | 0.053287                 |
| Paul Harrison's book | 1      | 0.05326045 (or 0.053310) |

- c. This file contains the **eigenenergies** and the **squared wave functions** ( $\psi^2$ ): Schroedinger\_1band / cb1\_sg1\_deg1\_psi\_squared.dat
  - d. This file contains the **eigenenergies** and the **wave functions** ( $\psi$ ): Schroedinger\_1band / cb1\_sg1\_deg1\_psi.dat
- a) and b) can be used to plot the data as shown in the figure above.

## Varying the electric field

There are two ways to vary the applied electric field from 0 kV/cm to -70 kV/cm.

### Possibility 1

We perform several individual calculations and vary the strength of the electric field by specifying its value for each individual calculation.

```
$electric-field
electric-field-on          = yes      ! 'yes' / 'no'
electric-field-strength    = 0.0      ! in units of [V/m] - Here:  0 kV/cm
```

(continues on next page)

(continued from previous page)

```

! electric-field-strength = -5e5 ! in units of [V/m] - Here: -5 kV/cm
! electric-field-strength = -10e5 ! in units of [V/m] - Here: -10 kV/cm
! electric-field-strength = -15e5 ! in units of [V/m] - Here: -15 kV/cm
! electric-field-strength = -20e5 ! in units of [V/m] - Here: -20 kV/cm
! electric-field-strength = -25e5 ! in units of [V/m] - Here: -25 kV/cm
! electric-field-strength = -30e5 ! in units of [V/m] - Here: -30 kV/cm
! electric-field-strength = -40e5 ! in units of [V/m] - Here: -40 kV/cm
! electric-field-strength = -50e5 ! in units of [V/m] - Here: -50 kV/cm
! electric-field-strength = -60e5 ! in units of [V/m] - Here: -60 kV/cm
! electric-field-strength = -70e5 ! in units of [V/m] - Here: -70 kV/cm
    electric-field-direction = 0 0 1 ! [001] direction, i.e. along z axis
$end_electric-field

```

## Possibility 2

An **alternative** (and much more user friendly approach) would be the usage of an “electric field sweep”. Here, only one calculation is necessary. The variation of the electric field strength is done automatically.

```

$electric-field
electric-field-on = yes ! 'yes' / 'no'
electric-field-strength = 0.0 ! in units of [V/m] - Here: 0 kV/cm
electric-field-direction = 0 0 1 ! [001] direction, i.e. along z axis.
!
electric-field-sweep-active = yes ! 'yes' / 'no'
electric-field-sweep-step-size = -5e5 ! in units of [V/m] - Here: -5 * 10^5
↪ V/m = -5 kV/cm
electric-field-sweep-number-of-steps = 14 ! 14 steps, starting from 0 kV/cm to -
↪ 70 kV/cm
$end_electric-field

```

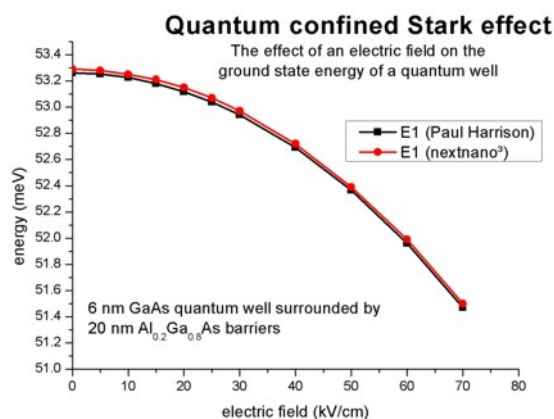
Here, the electric field is varied from 0 kV/cm to -70 kV/cm, in steps of -5 kV/cm. The output of the eigenvalues is then contained in `Schroedinger_1band / electric_ev_cb1_sg1_deg1.dat`.

The first column contains the strength of the electric field in units of [kV/cm]. The second column contains the 1<sup>st</sup> eigenvalue for the specified electric field in units of [eV]. The third column contains the 2<sup>nd</sup> eigenvalue for the specified electric field in units of [eV].

The following figure shows the ground state energy of the 6 nm quantum well as a function of the applied electric field strength  $F$ . The calculated energies can be represented by a parabolic fit. Over the range of electric fields investigated, the ground state energy can be represented by the parabola:

$$E_1(F) = E_1(0) - 0.000365F^2$$

where  $E_1(0)$  refers to the ground state energy at zero electric field (in units of meV). Here, the electric field strength  $F$  is given in units of kV/cm. (Note: Paul Harrison’s value is 0.00036)



This suppression of the confined energy level by an electric field is called “Quantum Confined Stark Effect (QCSE)”. The data that were plotted here are contained in this file. `Schroedinger_1bandelectric[kV/cm]_ev1D_cb1_qc1_sg1_deg1_dir.dat`. The energy levels are contained as a function of the sweep variable “electric field”.

The plot is (almost) in agreement with Fig. 3.22 (p. 96) of Paul Harrison’s book “*Quantum Wells, Wires and Dots*”. The energies differ slightly although we used

- identical effective masses
- identical conduction band offset
- identical grid resolution (0.1 nm)

The only difference is that we use multiple points at the heterointerfaces but this should not explain the difference. The answer lies probably in a tiny inconsistency in the book. On page 96 it says: “where  $E_1(0)$  refers to the ground-state energy (53.310 meV) at zero field” However, the value that was used in Fig. 3.22 is  $E_1(0) = 53.26045$  meV. (This value can be found on the CD that accompanies the book.) The nextnano<sup>3</sup> value is 53.287 meV which lies in the middle between these two values.

## Output

The energy values were taken from the file `Schroedinger_1band / ev_cb1_sg1_deg1.dat`.

For example, the values for zero applied electric field read:

|                      | num_ev | eigenvalue [eV]          |
|----------------------|--------|--------------------------|
| nextnano++           | 1      | 0.053287                 |
| Paul Harrison’s book | 1      | 0.05326045 (or 0.053310) |

The values for an applied electric field of -70 kV/cm read:

|                      | num_ev | eigenvalue [eV] |
|----------------------|--------|-----------------|
| nextnano++           | 1      | 0.051497        |
| Paul Harrison’s book | 1      | 0.051472        |

## 3.3.2 Optical interband absorption in a quantum well including excitonic effects

This tutorial presents calculation of interband absorption spectrum in a quantum well including excitonic effects.

There is a separate tutorial that discusses the calculation of the exciton binding energy and exciton Bohr radius of an infinite quantum well: [Exciton energy in quantum wells - Tutorial](#)

In this tutorial we calculate the absorption spectrum of a 10 nm GaAs quantum well. The purpose is to calculate the absorption spectrum for a simple model and model that includes *excitonic effects* on the absorption spectrum.

The absorption spectrum has been calculated using a simple model assuming a parabolic energy dispersion. In order to keep things simple, i.e. to be able to compare our results with analytical formula, we used the same effective mass for electrons and holes ( $m_e = m_h = 0.065 m_0$ ).

The excitonic binding energy  $E_b$  has been calculated to be -9.5 meV. Therefore, the absorption spectrum that includes excitonic contributions starts at an energy roughly 10 meV below than band gap. The exciton Bohr radius  $\lambda$  was found to be 13.1 nm.

For Lorentzian broadening we use a linewidth of FWHM = 6 meV, and for Gaussian broadening we use FWHM = 10 meV. The FWHM(Voigt) depends in a complicated way on FWHM(Lorentzian) and FWHM(Gaussian).

| Property                    | Symbol     | unit  | analytical calculation      | nextnano |
|-----------------------------|------------|-------|-----------------------------|----------|
| quantum well width          | L          | nm    | 10.0                        | 10.0     |
| barrier height              | $E_b$      | eV    | infinite quantum well model | 1000     |
| effective electron mass     | $m_e$      | $m_0$ | 0.0665                      | 0.0665   |
| effective hole mass         | $m_h$      | $m_0$ | 0.0665                      | 0.0665   |
| refractive index            | $n_r$      |       | 3.3                         | 3.3      |
| linewidth (FWHM) Lorentzian | $\Gamma_L$ | meV   | n/a                         | 6        |
| linewidth (FWHM) Gaussian   | $\Gamma_G$ | meV   | n/a                         | 10       |
| temperature                 | T          | K     | 300                         | 300      |

The Coulomb enhancement factor is given by  $S_{2D} = \frac{\exp(\pi/\sqrt{\Delta})}{\cosh(\pi/\sqrt{\Delta})}$ , where  $\Delta$  is the total excess kinetic energy of the electron–hole pair normalized to  $E_b/4$  [LeverJLT2010].

We observe two major contributions to the absorption spectrum:

- A distinct peak a few meV (corresponding to the exciton binding energy  $E_b$ ) lower than the absorption edge (band gap). This is the signature of the bound exciton.
- Sommerfeld enhancement: In the continuum part of the absorption spectrum, it is scaled via the Coulomb enhancement factor  $S_{2D}$ .

---

— Begin —

**Automatic documentation: Running simulations, generating figures and reStructured Text (\*.rst) using nextnanopy**

The following documentation and figures were generated automatically using *nextnanopy*.

The following Python script was used: 1D\_InterbandAbsorption\_InfiniteWell\_Exciton\_nextnano3.py

---

The following figures have been generated using *nextnano*<sup>3</sup>.

The absorption spectrum has been calculated using a simple model assuming a parabolic energy dispersion.

#### Infinite QW (single-band)

##### Optical absorption spectrum of bulk crystal and of a quantum well

##### Optical absorption of a 10 nm quantum well

##### Optical absorption of a 10 nm quantum well using different broadening functions

##### Optical absorption of a 10 nm quantum well showing the different contributions to the excitonic absorption

We acknowledge funding from the European Union’s Horizon 2020 research and innovation programme under grant agreement No. 101017194 (SiPho-G).

---

**Automatic documentation: Running simulations, generating figures and reStructured Text (\*.rst) using nextnanopy**

---

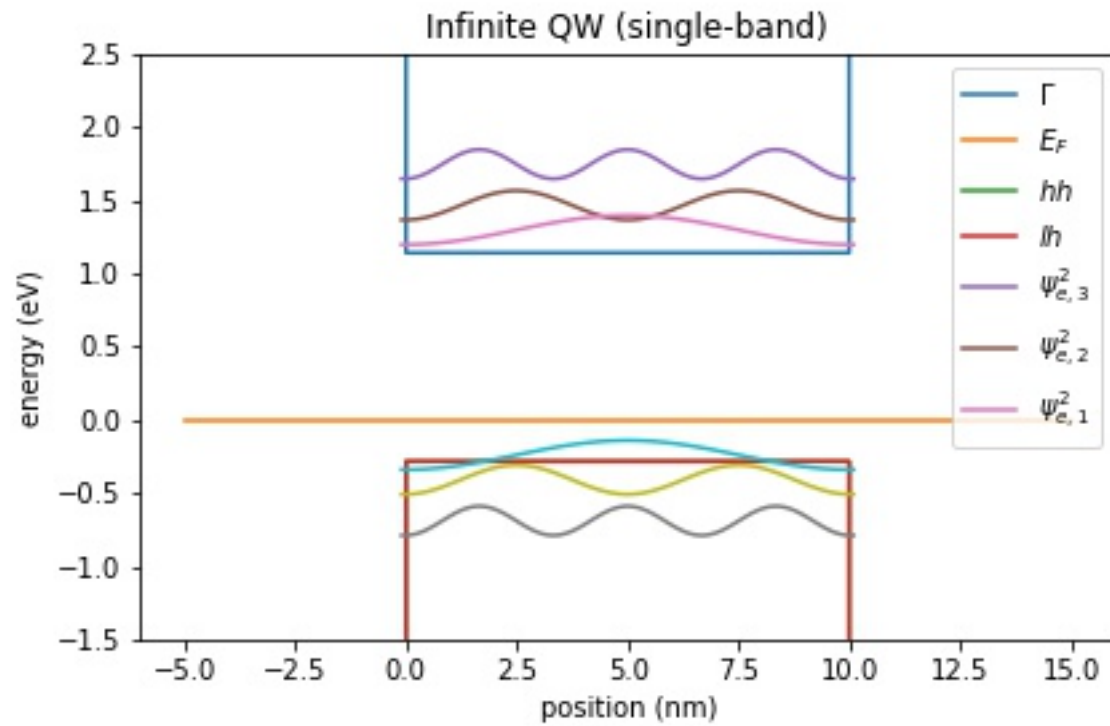


Figure 3.3.2.1: Conduction and valence band edges, Fermi level, electron and holes states of a quantum well

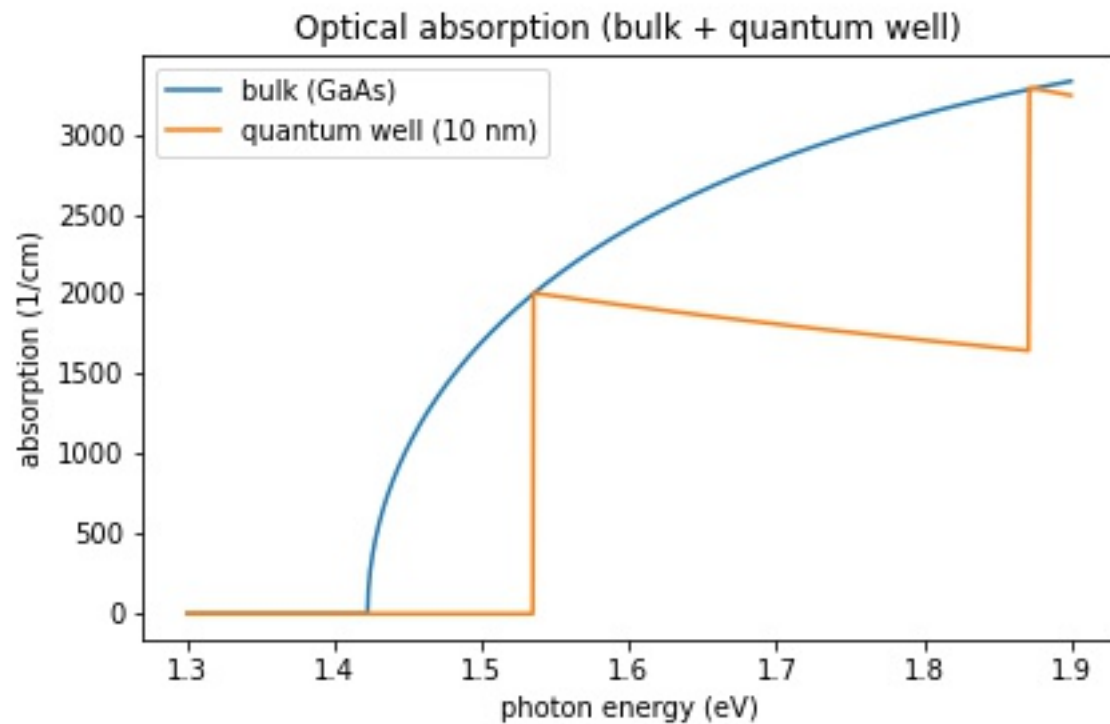


Figure 3.3.2.2: The absorption spectrum of bulk GaAs and a 10 nm infinite quantum well.

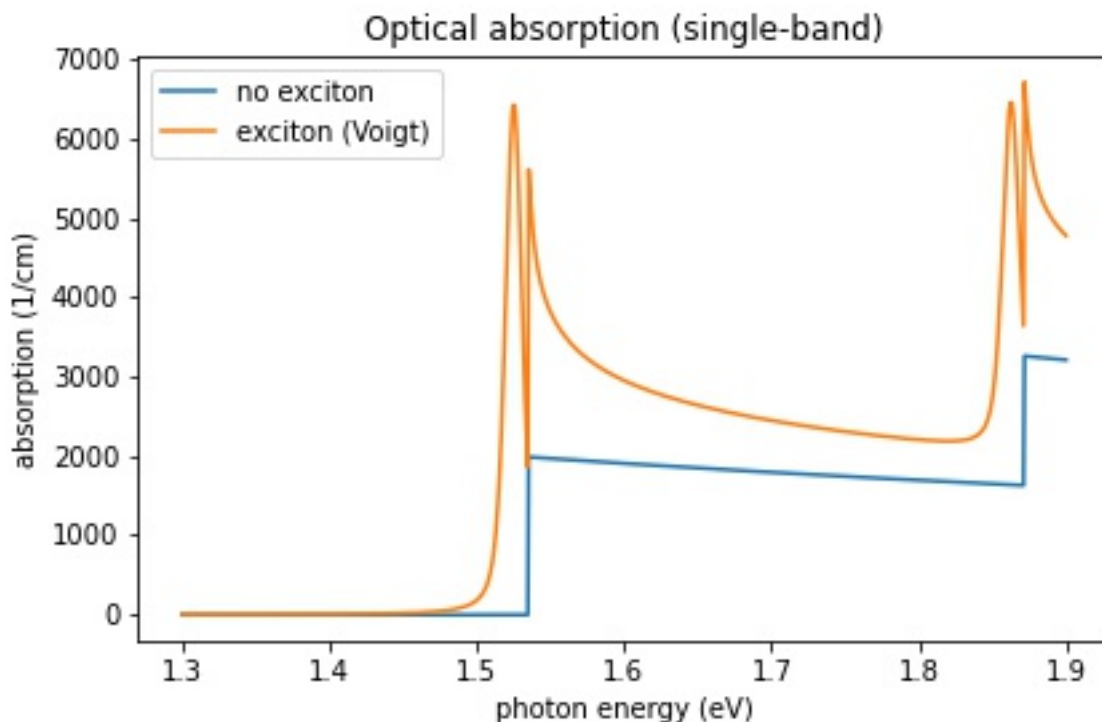


Figure 3.3.2.3: The absorption spectrum of a 10 nm infinite quantum well consisting of GaAs. The absorption spectrum has been calculated with and without excitonic contributions to the spectrum.

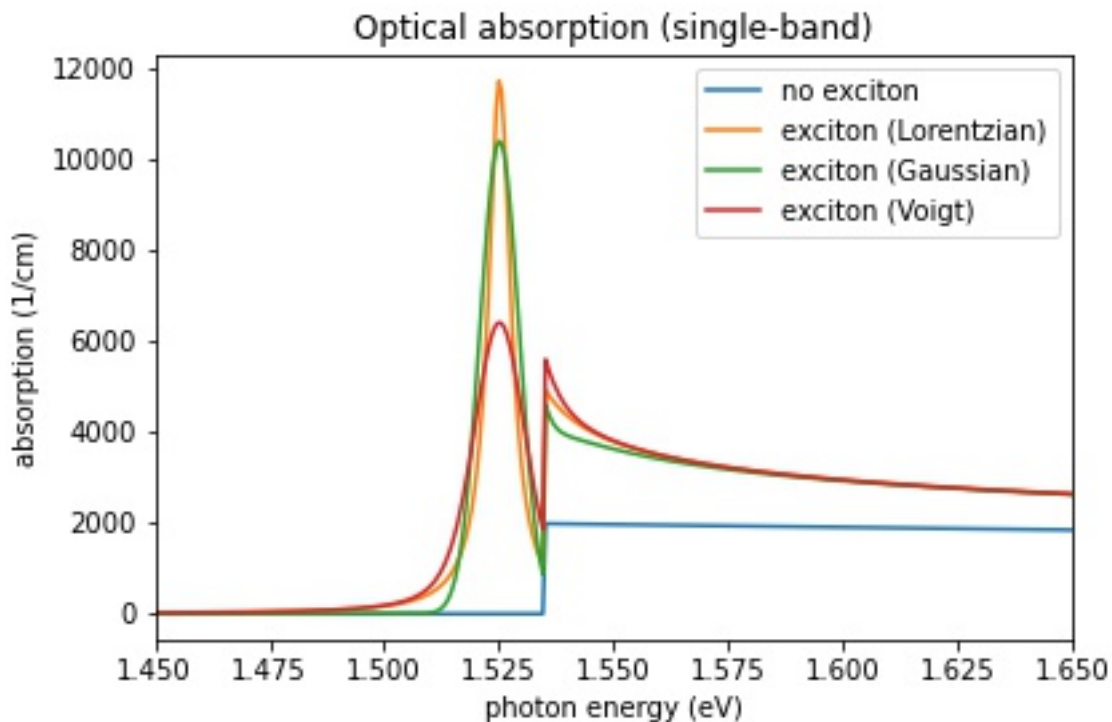


Figure 3.3.2.4: The absorption spectrum of a 10 nm infinite quantum well consisting of GaAs. The absorption spectrum has been calculated with Lorentzian, Gaussian and Voigt broadening function, and without excitonic contributions to the spectrum.

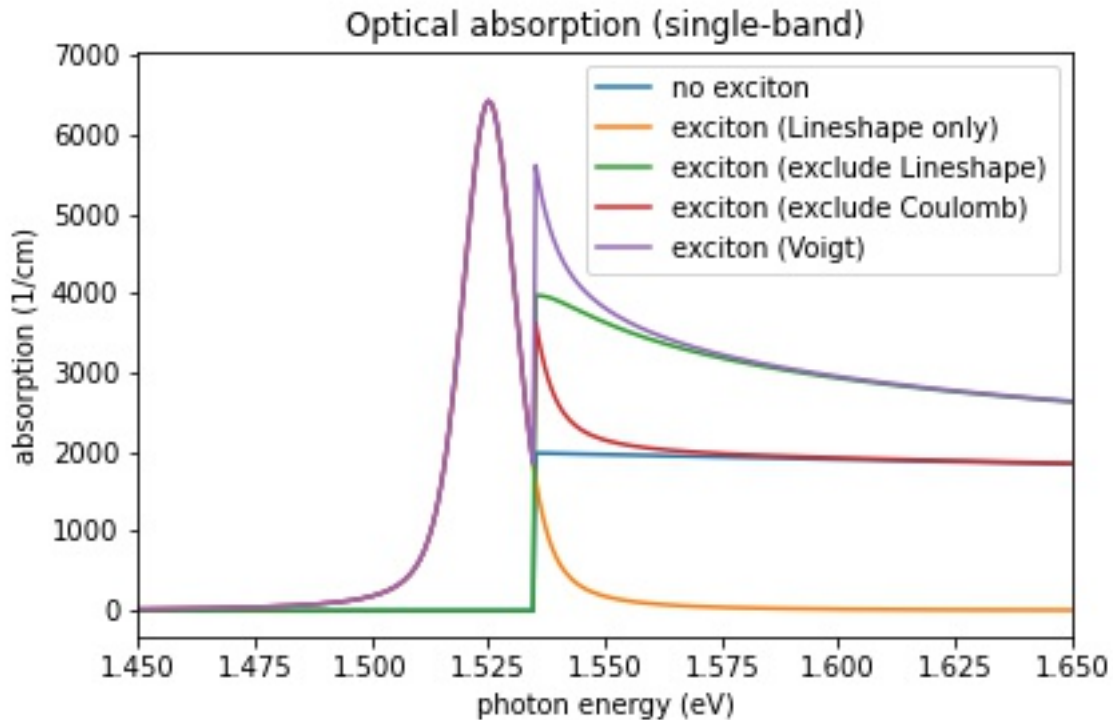


Figure 3.3.2.5: The absorption spectrum of a 10 nm infinite quantum well consisting of GaAs showing the different excitonic contributions. The absorption spectrum has been calculated with a Voigt broadening functions, and without excitonic contributions to the spectrum. In order to see the contributions of the Coulomb enhancement factor and the lineshape peak, each of these contributions can be switched off.

— End —

### 3.3.3 Schottky barrier

#### Input Files:

- *1DSchottky\_barrier\_GaAs\_nn3.in*
- *2DSchottky\_barrier\_GaAs\_nn3.in*
- *1DSchottky\_barrier\_GaAs\_ohmic\_nn3.in*
- *1DSchottky\_barrier\_GaAs\_SchottkyBarrier0V\_nn3.in*
- *1DSchottky\_barrier\_GaAs\_surface\_density\_nn3.in*
- *1DSchottky\_barrier\_GaAs\_surface\_states\_acceptor\_nn3.in*

#### Scope:

In this tutorial we simulate Schottky contacts.

#### Introduction

When a metal is in contact with a semiconductor, a potential barrier is formed at the metal-semiconductor interface. In 1938, Walter Schottky suggested that this potential barrier arises due to stable space charges in the semiconductor. At thermal equilibrium, the Fermi levels of the metal and the semiconductor must coincide. There are two limiting cases:

- Ideal Schottky barrier:**

- metal/n-type semiconductor: The barrier height  $\phi_B$  is the difference of the metal work function  $\phi_M$  and the electron affinity ( $\chi$ ) in the semiconductor.

$$e\phi_B = e(\phi_M - \chi_s)$$

- metal/p-type semiconductor: The barrier height  $\phi_{B,p}$  is given by:

$$e\phi_{B,p} = e(\phi_M - \chi_s) - E_{\text{gap}}$$

#### b) Fermi level pinning:

If surface states on the semiconductor surface are present: The barrier height is determined by the property of the semiconductor surface and is independent of the metal work function.

#### Attention

Note that this approach have physical sense only for structures that are not biased, in global equilibrium.

Consequence: The Schottky barrier sets a (Dirichlet) boundary condition for the electrostatic potential, i.e. the solution of the Poisson equation in the semiconductor, because the conduction and valence band edge energies are in a definite energy relationship with the Fermi level of the metal.

```
contacts{
  schottky{
    name = contact           # Schottky barrier
    bias = 0.0               # [V] apply voltage
    barrier = 0.53           # [V] GaAs, S.M. Sze, "Physics of Semiconductor_
    ↪Devices", p. 275 (2nd ed.)
  }
}
```

The n-type donor concentration in *GaAs* has been taken to be  $1 \cdot 10^{18} \text{ cm}^{-3}$  (fully ionized). The temperature is set to 300 K.

Figure 3.3.3.1 shows the conduction band edge profile for n-type *GaAs* in equilibrium with

- a Schottky barrier of 0.53 V, i.e. the conduction band edge is pinned 0.53 eV above the Fermi level (which is at 0 eV)
- a Schottky barrier of 0 V
- an ohmic contact at 10 nm.

(The contact region is from 0 nm to 10 nm but no equations are solved inside the contact region.)

Note that in equilibrium the Fermi level is constant and equal to 0 eV in the whole device. If the semiconductor is doped, the conduction and valence band edges are shifted with respect to this Fermi level, i.e. relative to 0 eV and are thus dependent on doping. This is a bulk property and independent of surface effects, like ohmic contacts or Schottky barrier height (see right part of the figure). At the left boundary, however, the band profile is affected by the type of contact.

#### Note

A Schottky barrier of 0 V is not equivalent to an ohmic contact!

An ohmic contact corresponds to a Neumann boundary condition for the Poisson equation, i.e.  $d\phi/dx = 0$  (constant electrostatic potential) or flat band condition which is equivalent to  $E = 0$ . A Schottky barrier  $\phi_B$  is a Dirichlet



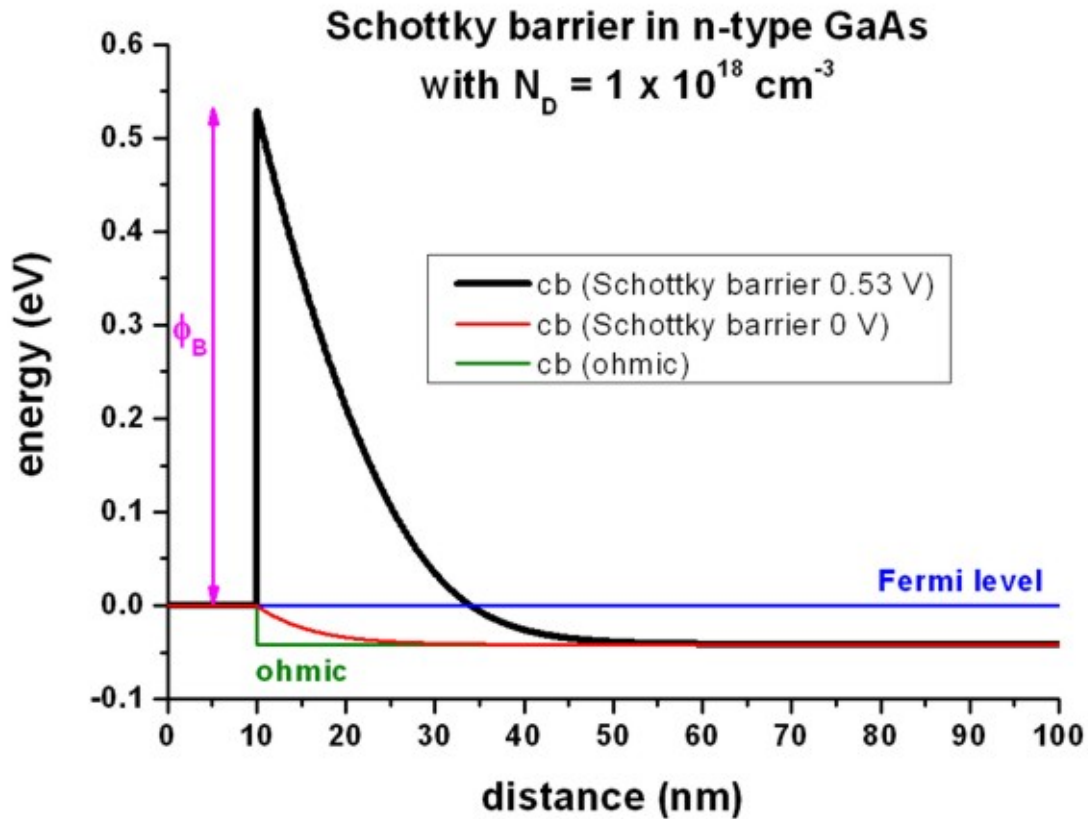


Figure 3.3.3.1: Calculated conduction band profile

boundary condition for the Poisson equation, i.e. the value of the conduction band edge at the boundary is fixed with respect to the Fermi level:

$$E_c - E_F = e\phi_B$$

In this particular example, an artificial Schottky barrier of -0.04184 V would be equivalent to an ohmic contact, (i.e. flat band condition), but only for the same temperature and the same doping concentration.

### Interface charges (surface states)

Input file: *1DSchottky\_barrier\_GaAs\_surface\_density.in*

Instead of specifying a Schottky barrier, the user can alternatively specify a fixed surface charge density.

```
structure{
  ...
  region{ # charge sheet
    line{ x = [10E0, 10E0 + $Width]}           # between contact/GaAs at 10 nm
    doping{
      constant{
        name = "negative-interface-charge"      # name of impurity
        conc = 2.7675e20                       # doping concentration [cm-3]
      }
    }
  }
}
```

```
impurities{
  ...
}
```

(continues on next page)

(continued from previous page)

```

charge{
    name = "negative-interface-charge"           # refer to region with name_
    negative-interface-charge
    type = negative
}

```

Figure 3.3.3.2 shows that the red curve (= “ohmic” contact with interface charge density  $\sigma$  (surface states) of  $-2.7675 \cdot 10^{12} |e|/\text{cm}^2 = -4.4340 \cdot 10^{-3} \text{ C/m}^2$ ) is equivalent to the black curve (Schottky barrier of 0.53 eV).

A sheet charge density of  $-2.7675 \cdot 10^{12} \text{ cm}^{-2}$  corresponds to a volume charge of  $-2.7675 \cdot 10^{20} \text{ cm}^{-3}$  if one assumes this charge to be distributed over a grid spacing of 0.1 nm. In this case, the interface charge density corresponds to a Neumann boundary condition for the derivative of the electrostatic potential  $\phi$ :

$$\frac{d\phi}{dx} = -E_x = \text{const},$$

where  $E_x$  is the electric field component along the x direction.  $E_x$  is related to the interface charge as follows:

$$E_x = \frac{\sigma}{\epsilon_r \epsilon_0}$$

where  $\epsilon_0$  is the permittivity of vacuum and  $\epsilon_r$  is the dielectric constant of the semiconductor. In this example:

- $\epsilon_r = 12.93$  for GaAs
- $E_x = 387.3 \text{ kV/cm}$

The output for the electric field (in units of [kV/cm]) can be found in this file: *electric\_field.dat*

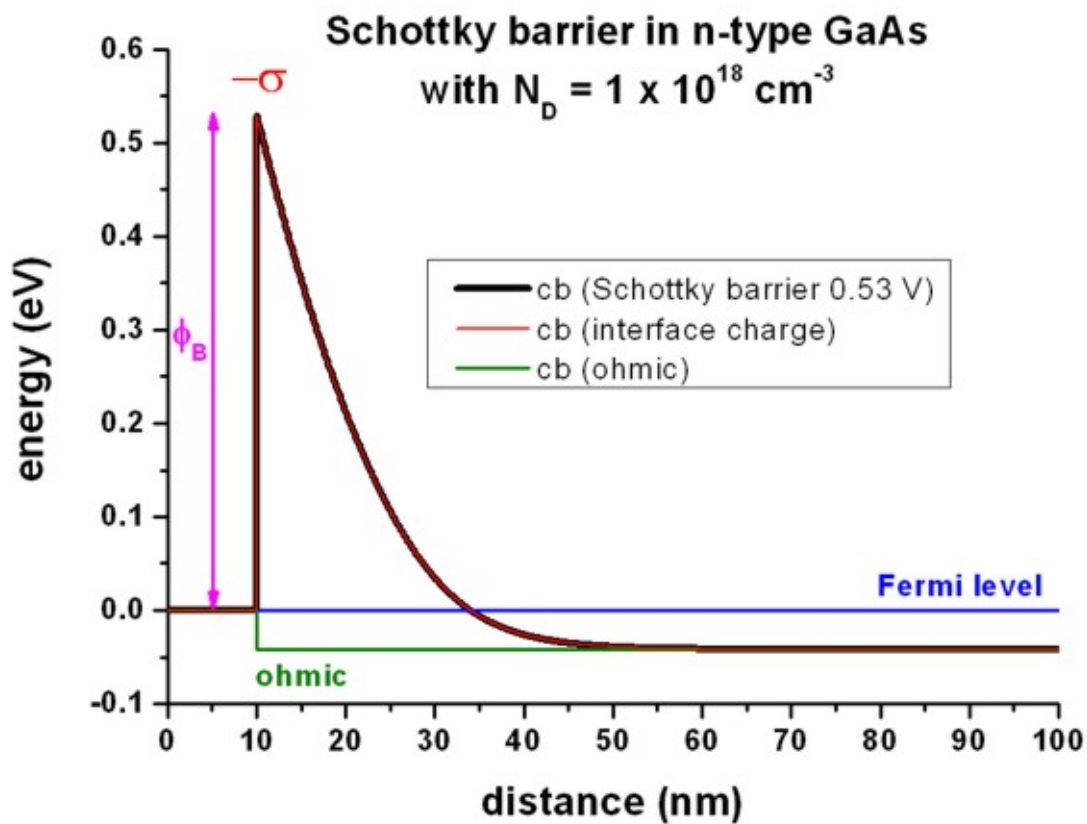


Figure 3.3.3.2: Calculated conduction band profile

The output for the interface densities can be found in this file: *materialdensity\_fixed\_charge.dat*.

## Surface states - Acceptors

Input file: *1DSchottky\_barrier\_GaAs\_surface\_states\_acceptor\_nnp.in*

Instead of specifying a Schottky barrier, the user can alternatively specify a density of acceptor surface states (p-type doping). Essentially, this can be done by specifying a p-type doping region that is very thin, i.e. the doping is specified only on one grid point.

In this example, we use a doping area of 0.1 nm at the surface that we dope p-type with a volume density of  $276.75 \cdot 10^{18} \text{ cm}^{-3}$ . This corresponds to a sheet charge density of  $2.7675 \cdot 10^{12} \text{ cm}^{-2}$  where we assume the states to be fully ionized.

```
impurities{
  ...
  acceptor{ # p-type
    name = "impurity" # refer to region with name impurity
    energy = -1000E0 # all ionized
    degeneracy = 4 # degeneracy of energy levels, 2 for n-type, 4 for p-type
  }
}
```

The results are the same as shown in Figure 3.3.3.2 for the interface charges.

## 3.4 Transmission

### 3.4.1 Efficient method for the calculation of ballistic quantum transport - The CBR method (2D example)

In this tutorial we apply the Contact Block Reduction (CBR) method to a Aharonov-Bohm-type structure with a large barrier in the middle of the device.

The following two *nextnano<sup>3</sup>* input files are based on the paper [MamaluyCBR2003]

- 2D\_CBR\_MamaluySabathilJAP2003.in
- 2D\_CBR\_MamaluySabathilJAP2003\_holes.in

the latter of which simulates holes instead of electrons.

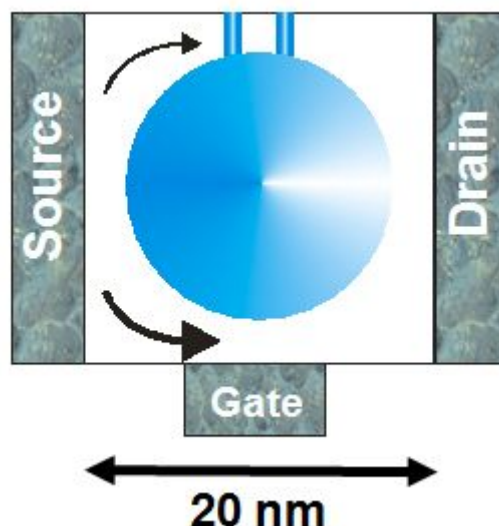
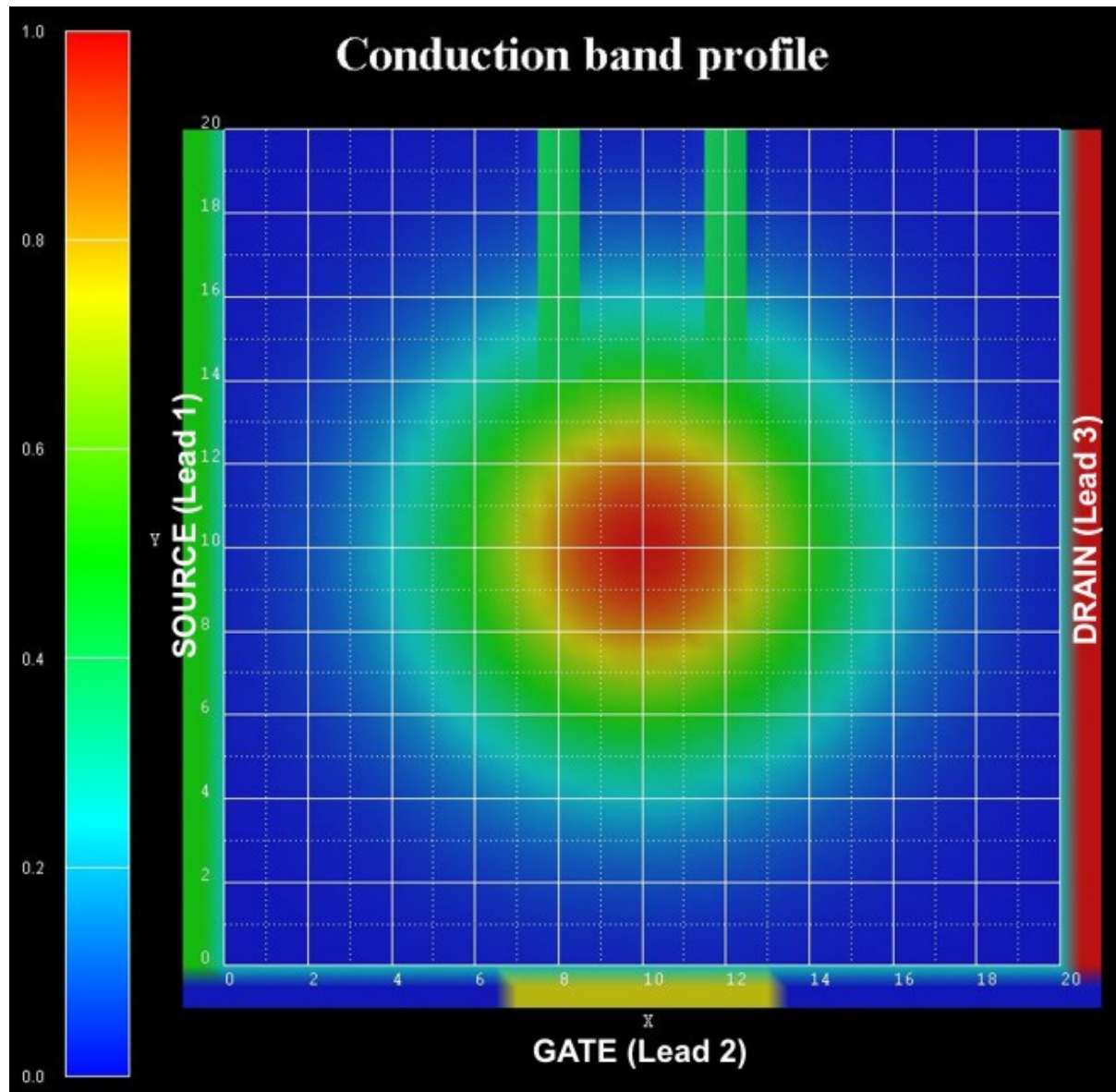


Figure 3.4.1.1: Schematic sketch of the device showing two possible paths of the electrons

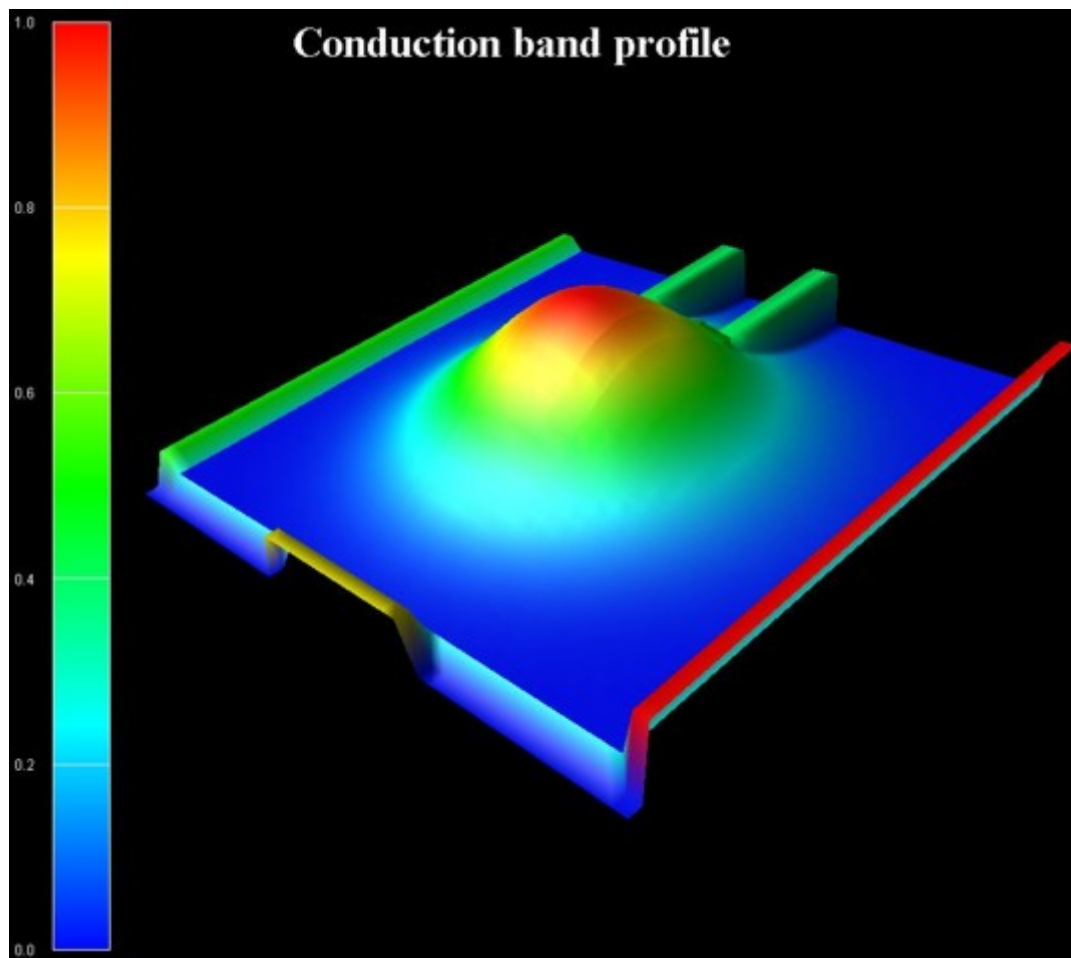
- The device consists of three leads that are called ‘source’, ‘gate’ and ‘drain’ in this example.

- In the middle of the device a potential barrier of two-dimensional Gaussian shape effectively expels the electrons from the center.
- In the upper part of the device, a thin tunneling double barrier is present.
- The device dimensions are 20 nm x 20 nm.

A detailed description of the device can be found in Section V of [MamalyCBR2003].



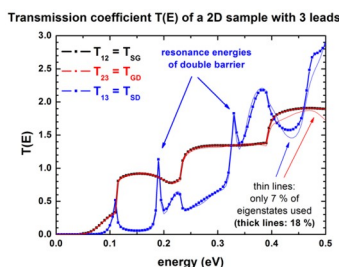
- The effective electron mass is assumed to be constant throughout the device and equal to  $0.3 m_0$ .
- The device region consists of  $41 \times 41 = 1681$  grid points, which is equivalent to a grid spacing of 0.5 nm. This means that the device Hamiltonian is a matrix of size  $1681 \times 1681$ .
- The tunneling barriers have a width of 1 nm each and are separated by 3 nm.
- The maximum height of the Gaussian barrier is  $E_{c,0} = 1$  eV, the height of the double barriers is 0.4 eV.
- The conduction band profile is given by  $E_c = E_{c,0} \exp[-(x^2 + y^2)/a^2]$  where  $x$  and  $y$  are with respect to the center of the device, and  $a = 5$  nm.
- The conduction band profile is achieved by using an appropriate ternary material having a 2D Gaussian alloy profile.
- The lower gate is 6 nm long, all other leads are 20 nm long.



The following figure shows the calculated transmission coefficients of the various lead combinations  $T_{12}$ ,  $T_{23}$  and  $T_{13}$ . For the thick lines 18 % (303 of 1681) of all eigenvectors were used whereas for the thin lines only 7 % (118 of 1681) had to be calculated, i.e. one does not have to calculate all eigenvalues of the device Hamiltonian which grossly reduces CPU time. A small percentage of eigenvalues suffices for  $T(E)$  in relevant energy range of interest.

The transmission coefficient can be found in this file:

CBR\_data1/transmission2D\_cb\_sg001\_ind000\_CBR.dat



The *nextnano<sup>3</sup>* results differ slightly from the [MamalyCBR2003] paper. Reason: The potential energy profile in the device and in the leads is not exactly identical, as well as the dimensions of the barriers. Therefore the eigenenergies and wave functions in the device, and in the leads differ slightly which explains the small deviations.

The eigenstates # 16 is a resonance state of the lower transmission path.

- 1st resonance: # 16: 0.123 eV
- Its square of the wave function is shown below.

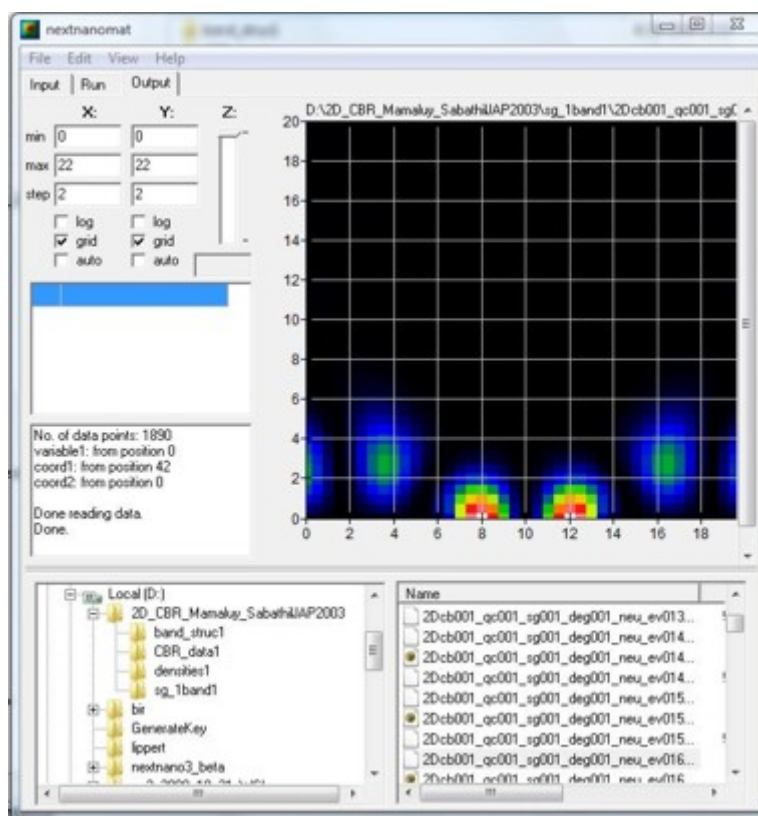


Figure 3.4.1.2: The *nextnanomat* screenshot (nextnanomat has been developed by Jörg Ehehalt)

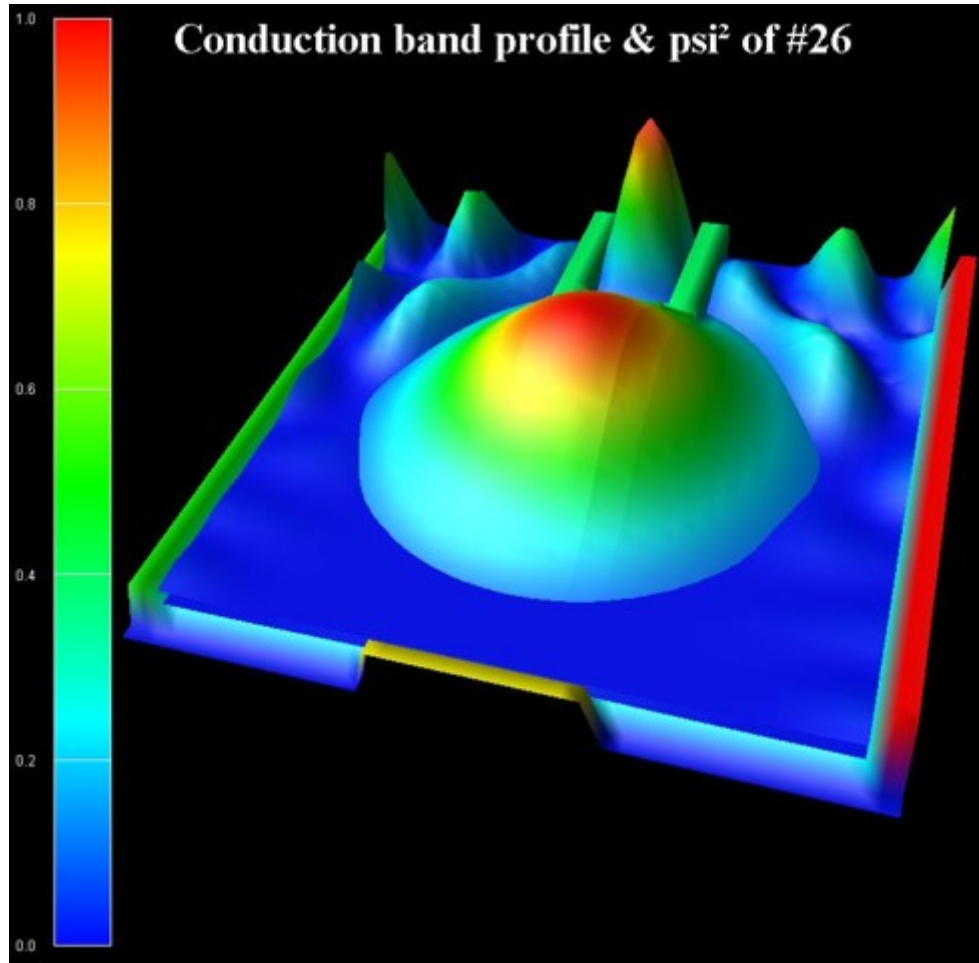
The eigenstates # 26 and # 29 are resonance states of the double barrier.

- **1st resonance:**
  - # 26: 0.182 eV (delocalized)



- # 29: 0.196 eV (more localized)
- **2nd resonance:**
  - # 55: 0.322 eV (delocalized)
  - # 57: 0.333 eV (more localized)
  - # 60: 0.345 eV (delocalized)
  - # 63: 0.359 eV (more localized)

The following figure shows the conduction band profile together with the square of the wave function of the 26th eigenstate. One can clearly see that it is a resonance state of the double barrier and corresponds to the second peak in the blue transmission curve  $T_{13}$  from source to drain around 180 meV.



### Lead modes

The following two figures show the lead modes of the gate, and the source (which is identical to the drain). In the transmission curve  $T_{12}(E) = T_{23}(E)$ , the transmission shows a step-like behavior which is related to the energies of lead no. 2 ('gate').

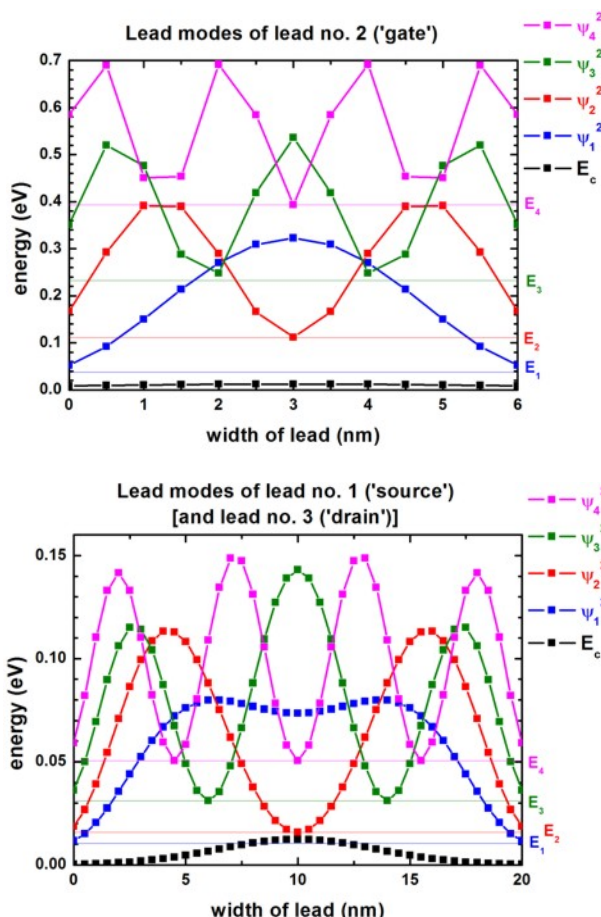
**The lead modes (eigenvalues, psi, psi<sup>2</sup>, band edge profile, ...) can be found in these files:**

CBR\_data1/modes\_lead00\*\_sg001\_\*.dat

### Technical details

#### Definition of contacts

For each contact (lead), a quantum cluster ("lead quantum cluster") has to be defined because in each lead, a one-dimensional Schrödinger equation has to be solved which gives us the lead modes (i.e. energies and eigenvectors).



of the leads). In addition, a quantum cluster is required for the device itself (“main quantum cluster”).

```
!-----!
$quantum-regions                                     !

region-number   = 1                                  ! 'device'
base-geometry   = rectangle                           !
region-priority = 2                                  !
x-coordinates   = 0d0 20d0                           ! [nm] width of 'device' = 20 nm
y-coordinates   = 0d0 20d0                           ! [nm] width of 'device' = 20 nm

region-number   = 2                                  ! 'source'
base-geometry   = rectangle                           !
region-priority = 1                                  !
x-coordinates   = -1d0 -0.5d0                         ! [nm] (including 2 grid points,
↪along this direction)
y-coordinates   = 0d0 20d0                           ! [nm] length of 'lead 1' = 20 nm

region-number   = 3                                  ! 'gate'
base-geometry   = rectangle                           !
region-priority = 1                                  !
x-coordinates   = 7d0 13d0                            ! [nm] length of 'lead 2' = 6 nm
y-coordinates   = -1d0 -0.5d0                         ! [nm] (including 2 grid points,
↪along this direction)

region-number   = 4                                  ! 'drain'
base-geometry   = rectangle                           !
```

(continues on next page)



(continued from previous page)

```

region-priority = 1                                !
x-coordinates   = 20.5d0 21d0                      ! [nm] (including 2 grid points,
↪along this direction)
y-coordinates   = 0d0    20d0                      ! [nm] length of 'lead 3' = 20 nm

$end_quantum-regions                                !
!-----!

```

For each quantum cluster, the number of eigenstates to be calculated and its boundary conditions have to be specified.

For the main quantum cluster it holds: For each grid point in the main quantum cluster it is checked if it is at the boundary and if it is in contact to a lead. If it is at the boundary, and if it is in contact to a lead, a Neumann boundary condition is set. If it is at the boundary, and if it is not in contact to a lead, a Dirichlet boundary condition is set.

```

!-----!
$quantum-model-electrons                                !
model-number      = 1                                  !
model-name        = effective-mass                     ! single band effective-mass,
↪Schrödinger equation
cluster-numbers   = 1                                  !
conduction-band-numbers = 1                            ! Gamma band
!number-of-eigenvalues-per-band = 118                  ! corresponds to 7.0 % of 1681
number-of-eigenvalues-per-band = 303                  ! corresponds to 18.0 % of 1681
!number-of-eigenvalues-per-band = 1681                ! corresponds to 100.0 % of 1681
quantization-along-axes = 1 1 0                       ! (x,y)-plane
boundary-condition-100 = Neumann                       ! Neumann along propagation direction
boundary-condition-010 = Neumann                       ! Neumann along propagation direction

lead 1 = source: number of modes per lead = 41 = maximum number of relevant quantum,
↪grid points in lead 1
!-----!
model-number      = 2                                  !
...
cluster-numbers   = 2                                  !
number-of-eigenvalues-per-band = 41                   ! calculate 41 lead modes
boundary-condition-010 = Dirichlet                     ! Dirichlet perpendicular to,
↪propagation direction

!-----!
! lead 2 = gate:  number of modes per lead = 13 = maximum number of relevant quantum,
↪grid points in lead 2
!-----!
model-number      = 3                                  !
...
cluster-numbers   = 2                                  !
number-of-eigenvalues-per-band = 13                   ! calculate 13 lead modes
boundary-condition-100 = Dirichlet                     ! Dirichlet perpendicular to,
↪propagation direction

!-----!
! lead 3 = drain:  number of modes per lead = 41 = maximum number of relevant quantum,
↪grid points in lead 3
!-----!
model-number      = 4                                  !
...
cluster-numbers   = 2                                  !

```

(continues on next page)

(continued from previous page)

```

number-of-eigenvalues-per-band = 41           ! calculate 41 lead modes
boundary-condition-010          = Dirichlet    ! Dirichlet perpendicular to
↪propagation direction

$end_quantum-model-electrons    !
!-----!

!-----!
$CBR-current                    !
!
destination-directory = CBR_data1/           ! directory for output and data files
calculate-CBR         = yes                  ! flag: "yes"/"no"
!
main-qr-num           = 1                   ! number of main quantum cluster for
↪which transport is calculated: 'device'
num-leads             = 3                   ! total number of leads attached to
↪main region
lead-qr-numbers       = 2 3 4               ! quantum cluster numbers
↪corresponding to each lead
propagation-direction = 1 2 1 ! '1'=x, '2'=y ! direction of propagation (1,2,3)
↪for each lead
num-modes-per-lead    = 41 13 41           ! number of modes per lead used for
↪CBR calculation
! must be <= number of eigenvalues
↪specified in corresponding quantum model
!num-eigenvectors-used = 118                ! corresponds to 7.0 % of 1681
num-eigenvectors-used = 303                 ! corresponds to 18.0 % of 1681
!num-eigenvectors-used = 1681 ! 1681=41*41=N_x*N_y! number of eigenvectors in main
↪quantum cluster used for CBR calculation
E-min                 = 0.0d0               ! [eV] lower boundary for
↪transmission energy interval
E-max                 = 0.5d0               ! [eV] upper boundary for
↪transmission energy interval
num-energy-steps       = 100                ! number of energy steps in T(E)
!
$end_CBR-current       !
!-----!

```

For each energy  $E$  (num-energy-steps = 100) where the transmission coefficient  $T(E)$  has to be calculated, a matrix of size 95 x 95 has to be inverted. The size of 95 is determined by the sum of the number of grid points in each lead that are in contact to the device.

- Lead 1 (Source): 41 grid points
- Lead 2 (Gate): 13 grid points
- Lead 3 (Drain): 41 grid points
  - in total: 95 grid points
  - The total CPU time for calculation of the transmission  $T(E)$  in this example was about 30 seconds for 303 eigenstates.

For further information, please study this section: \$CBR-current .

### 3.4.2 Transmission through a 3D nanowire (3D example)

In this tutorial we apply the Contact Block Reduction (CBR) method to a simple GaAs nanowire of cuboidal shape, using the input files

- `transmission-nanowire_GaAs_3D_nnp.in`

the latter of which is for holes instead of electrons.

The corresponding tutorial for *nextnano++* can be found [here](#).

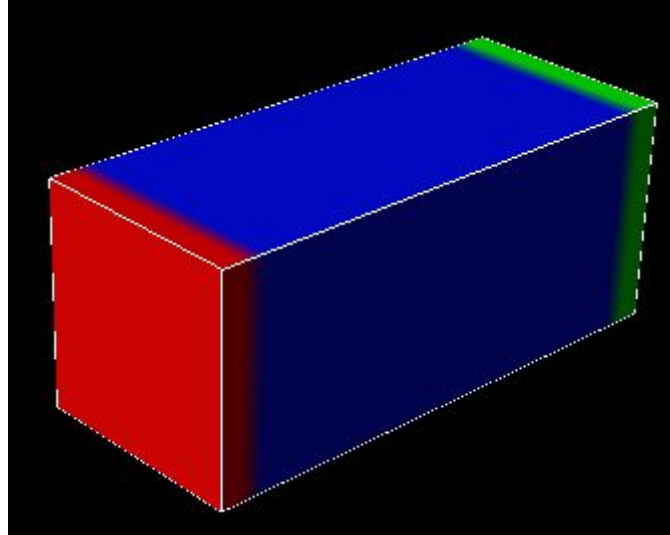


Figure 3.4.2.1: Schematic sketch of the 3D device showing the GaAs region that is placed between two contacts (red and green leads)

- The device consists of two leads of 10 nm x 10 nm each. Each lead has a total of 121 grid points (11 x 11 grid points).
- The leads are described as quantum regions, i.e. in each lead (quantum region) a two-dimensional Schrödinger equation has to be solved to obtain the eigenenergies and wave functions of the lead modes.
- The device dimensions are 10 nm x 10 nm x 20 nm.
- The grid spacing is 1 nm in all directions.
- The effective electron mass is assumed to be constant throughout the device and equal to  $0.067 m_0$ .
- The device region consists of  $11 \times 11 \times 21 = 2541$  grid points, which is equivalent to a grid spacing of 1.0 nm. This means that the device Hamiltonian is a matrix of size  $2541 \times 2541$ .
- The conduction band profile is constant and set to  $E_c = 0$  eV.

The following figure shows the calculated transmission coefficient as a function of energy between the leads 1 and 2.

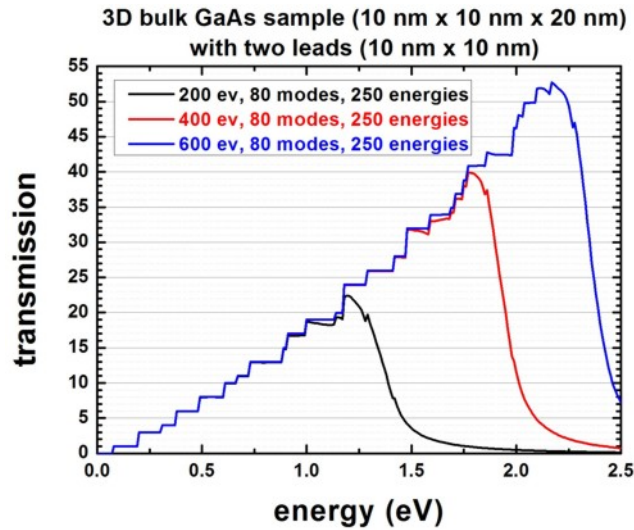
- For the blue lines 23.6 % (600 of 2541) of all eigenvectors were used whereas for the red lines only 15.7 % (400 of 2541) had to be calculated, i.e. one does not have to calculate all eigenvalues of the device Hamiltonian which grossly reduces CPU time.
- For the black lines 7.9 % (200 of 2541) of all eigenvectors were used.

A small percentage of eigenvalues suffices for  $T(E)$  in relevant energy range of interest. Note that the transmission drops significantly once the cutoff energy of the highest eigenvector taken into account is reached.

**The transmission coefficient can be found in this file:**

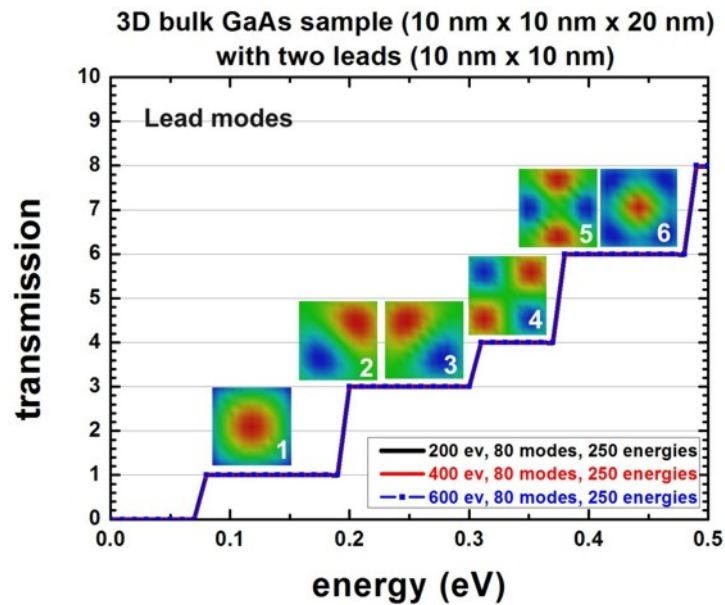
`CBR_data1/transmission3D_cb_sg001_CBR.dat`

In the following, the same data as above is shown again as a zoom into the energy range 0 eV - 0.5 eV. The colored figures show the wave function amplitude of the lowest energy lead modes.



They can be found in these files:

- CBR\_data1/2Dmodes\_lead001\_sg001\_psi\_ev001.dat
- ...
- CBR\_data1/2Dmodes\_lead001\_sg001\_psi\_ev080.dat
- CBR\_data1/2Dmodes\_lead002\_sg001\_psi\_ev001.dat
- ...
- CBR\_data1/2Dmodes\_lead002\_sg001\_psi\_ev080.dat



- Once the energy reaches 78 meV, the first lead mode energy is reached and then this mode transmits perfectly, giving a transmission of 1.
- The second and third lead mode states are degenerate due to the symmetry of the lead cross-section, thus they have the same energy (191 meV). Consequently, once the energy of 191 meV is reached, the transmission increases by 2.
- The total transmission is now equal to 3 as all lead modes transmit perfectly.
- The energy of the 4th lead mode is at 305 meV.

- The degeneracy of the 5th and 6th mode is accidental. They have the same energy.

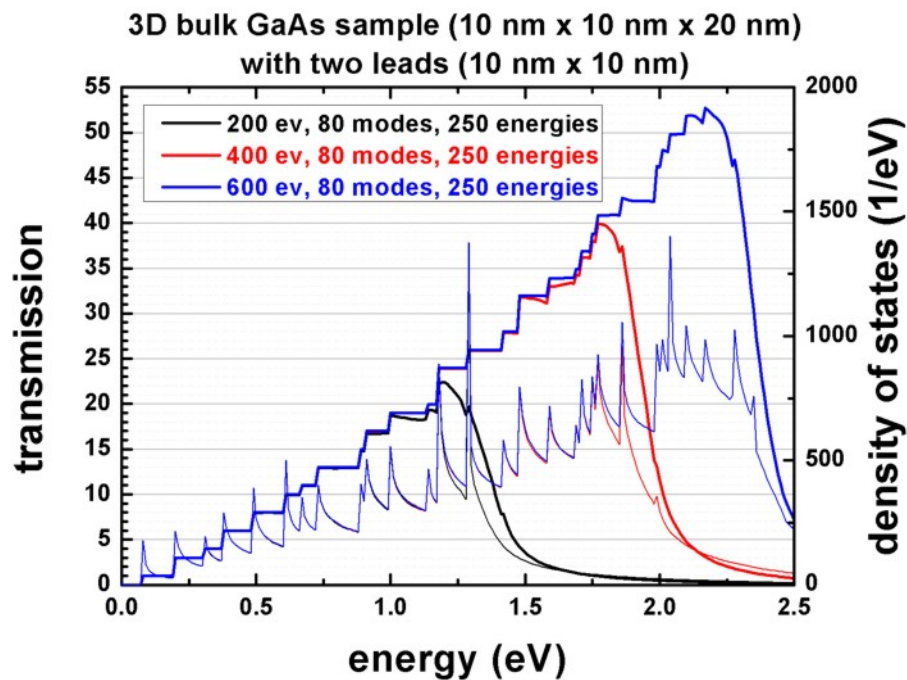
As one can clearly see, in this low energy limit, it is sufficient to calculate only a few percent of all eigenfunctions of the device Hamiltonian. For the leads, in all cases 80 eigenstates have been calculated.

## Density of states

Using

```
calculate-CBR-DOS = yes ! flag: yes / no
```

the CPU time increases but information like the density of states (DOS) can be obtained. This is shown in the next figure where the same transmission data as above is plotted but this time including the density of states. Again, the colors indicate taking into account 200, 400 or 600 eigenvectors of the decoupled system (closed system).



## Technical details

### Definition of contacts

For each contact (lead), a quantum cluster (“lead quantum cluster”) has to be defined because in each lead, a two-dimensional Schrödinger equation has to be solved which gives us the lead modes (i.e. energies and eigenvectors of the leads). In addition, a quantum cluster is required for the device itself (“main quantum cluster”).

```
!-----!
$quantum-regions                                !

region-number   = 1                               ! 'device'
base-geometry    = cuboid                          !
region-priority  = 1                               !
x-coordinates    = -5d0 5d0                        ! [nm] width of 'device' along x_
↪ = 10 nm
y-coordinates    = -5d0 5d0                        ! [nm] width of 'device' along y_
↪ = 10 nm
z-coordinates    = 0d0 20d0                        ! [nm] device length along z_
↪ = 20 nm
```

(continues on next page)

(continued from previous page)

```

region-number      = 2                      ! 'source' (lead 1)
base-geometry       = cuboid                  !
region-priority    = 2                      !
x-coordinates      = -5d0 5d0                ! [nm] width of 'lead 1' along x,
↳= 10 nm
y-coordinates      = -5d0 5d0                ! [nm] width of 'lead 1' along y,
↳= 10 nm
z-coordinates      = -2d0 -1d0                ! [nm] (including 2 grid points,
↳along this direction)

region-number      = 3                      ! 'gate' (lead 2)
base-geometry       = cuboid                  !
region-priority    = 2                      !
x-coordinates      = -5d0 5d0                ! [nm] width of 'lead 1' along x,
↳= 10 nm
y-coordinates      = -5d0 5d0                ! [nm] width of 'lead 1' along y,
↳= 10 nm
z-coordinates      = 21d0 22d0                ! [nm] (including 2 grid points,
↳along this direction)

!-----!
```

For each quantum cluster, the number of eigenstates to be calculated.

```

!-----!
$quantum-model-electrons      !
...
cluster-numbers               = 1
...
number-of-eigenvalues-per-band = 200        ! corresponds to 7.87 % of 2541
!number-of-eigenvalues-per-band = 400        ! corresponds to 15.74 % of 2541
!number-of-eigenvalues-per-band = 600        ! corresponds to 23.61 % of 2541

!-----!
! lead 1 = source: number of modes per lead = 80 maximum number of relevant quantum,
↳grid points in lead 1 = 121
!-----!
...
cluster-numbers               = 2            !
number-of-eigenvalues-per-band = 80          ! calculate 80 lead modes,
↳corresponds to 66.1 % of 121

!-----!
! lead 2 = source: number of modes per lead = 80 maximum number of relevant quantum,
↳grid points in lead 1 = 121
!-----!
...
cluster-numbers               = 3            !
number-of-eigenvalues-per-band = 80          ! calculate 80 lead modes,
↳corresponds to 66.1 % of 121

$end_quantum-model-electrons  !
!-----!
```

```

!-----!
$CBR-current
!
destination-directory = CBR_data1/      ! directory for output and data files
calculate-CBR         = yes             ! flag: "yes"/"no"
!
main-qr-num           = 1               ! number of main quantum cluster for
↪which transport is calculated: 'device'
num-leads              = 2              ! total number of leads attached to
↪main region
lead-qr-numbers        = 2 3            ! quantum cluster numbers
↪corresponding to each lead
propagation-direction = 3 3            ! '3'=z,      ! direction of propagation (1,2,3)
↪for each lead
num-modes-per-lead     = 80 80          ! number of modes per lead used for
↪CBR calculation
! must be <= number of eigenvalues
↪specified in corresponding quantum model
!num-eigenvectors-used = 200            ! corresponds to 7.87 % of 2541
!num-eigenvectors-used = 400            ! corresponds to 15.74 % of 2541
num-eigenvectors-used  = 600            ! corresponds to 23.61 % of 2541
!num-eigenvectors-used = 2541 ! 2541=11*11*21=N_x*N_y*N_z! number of eigenvectors in
↪main quantum cluster used for CBR calculation
E-min                  = 0.0d0          ! [eV] lower boundary for
↪transmission energy interval
E-max                  = 2.5d0          ! [eV] upper boundary for
↪transmission energy interval
num-energy-steps        = 250           ! number of energy steps in T(E)
!
$end_CBR-current
!-----!

```

### Loop over energy (parallelization)

For each energy  $E$  (num-energy-steps = 250) where the transmission coefficient  $T(E)$  has to be calculated, a matrix of size 160 x 160 has to be inverted.

The size of 160 is determined by the sum over the number of lead modes taken into account for each lead.

The upper limit would be the number of grid points in each lead that are in contact to the device, i.e. in this example where each lead has  $11 \times 11 = 121$  grid points, the maximum size of the matrix to be inverted could be  $242 = 121 + 121$ .

- Lead 1 (Source): 121 grid points (80 lead modes taken into account)
- Lead 2 (Drain): 121 grid points (80 lead modes taken into account)
  - in total: 242 grid points (but only 80 modes are taken into account for each lead:  $160 = 80 + 80$ )
  - The total CPU time for calculation of the transmission  $T(E)$  in this example was less than a minute for 200 eigenstates, 80 lead modes in each lead, and 250 energy steps.

This loop can be executed in parallel to improve computational performance.

```

$global-settings
...
!number-of-parallel-threads = 2 ! (for dual-core CPU)
number-of-parallel-threads  = 4 ! (for quad-core CPU)

```

For further information, please study this section: \$CBR-current .



### 3.4.3 1D - Quantum Tunneling: Comparison of CBR- and WKB approaches with the exact answer

#### Input Files:

*1D\_transmission\_GaAs\_analytic\_nn3.in*

#### Scope of the tutorial:

- General information about tunneling
- Comparison of the exact expression for the tunneling probability with that calculated by using CBR-based simulations and the semiclassical (WKB) method [Liang]

#### Main adjustable parameters:

- upper boundary for transmission energy - %E\_max
- analytical expression for the potential barrier - potential-from-function
- the effective mass of the electron - %effective\_mass

#### Relevant output Files:

- *Results\BandEdges.dat* (energy profile)
- *Results\Transmission\_cb\_sg1\_deg1.dat* (transmission)

#### Note

This tutorial is in progress and will be finalized soon.

Tunneling of particles through a potential barrier is one of the most well-known quantum phenomenon which does not exist in a classical world. Understanding the quantum tunneling and tunneling currents is very important for engineering and fabricating various elements of electronics.

As any quantum process, tunneling is described by the correspondent probability,  $\mathcal{T}$ , which depends on the energy of a propagating electron. There are only few potential shapes for which the analytical (at least approximate) expression for  $\mathcal{T}$  is known. Generically, one has to use either approximate or numerical methods to calculate  $\mathcal{T}$ .

In this tutorial, we discuss the tunneling probability through 1D potential defined as:

$$V(x) = V_0 / \cosh^2(\alpha(x - x_0)). \quad (3.4.3.1)$$

Here  $V_0$  is the height of the potential barrier, and  $\alpha^{-1}$  describes the width of the barrier, see the example in Figure 3.4.3.1. Eq. (3.4.3.1) belongs to these rare cases where the exact answer for  $\mathcal{T}$  is known [Landau Lifshitz], section §123 The general theory of scattering:

$$\begin{aligned} \mathcal{T}(E) &= \frac{\sinh^2(\pi k(E)/\alpha)}{\sinh^2(\pi k(E)/\alpha) + \cos^2(\pi/2\sqrt{1-\mathcal{U}})}, \quad \mathcal{U} < \infty; \\ \mathcal{T}(E) &= \frac{\sinh^2(\pi k(E)/\alpha)}{\sinh^2(\pi k(E)/\alpha) + \cosh^2(\pi/2\sqrt{\mathcal{U}-\infty})}, \quad \mathcal{U} > \infty; \\ \mathcal{U} &= \frac{8mV_0}{\hbar^2\alpha^2}, \quad k(E) = \sqrt{2mV_0/\hbar}; \end{aligned} \quad (3.4.3.2)$$

where  $E$ ,  $k$ , and  $m$  are the electron energy, wave vector, and effective mass, respectively;  $\hbar$  is the Planck constant.

The comparison of the transmission, which is obtained numerically (CBR), semi-analytically (WKB) and analytically, Eq. (3.4.3.2), is shown in Figure 3.4.3.2. The integration over the coordinate (between two turning points) in the WKB answer

$$\mathcal{T}_{\text{WKB}}(E) \simeq \exp\left(-2 \int_{x_1}^{x_2} k(x) dx\right) \quad (3.4.3.3)$$

has been performed numerically.

One can see that CBR is able to reproduce the exact answer with a high accuracy. Clearly, the WKB approach can be used to calculate transmission only in the semiclassical regime where transmission is small.



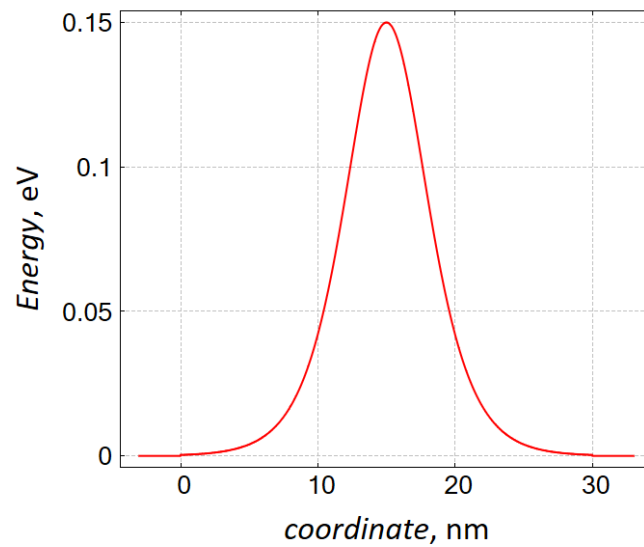
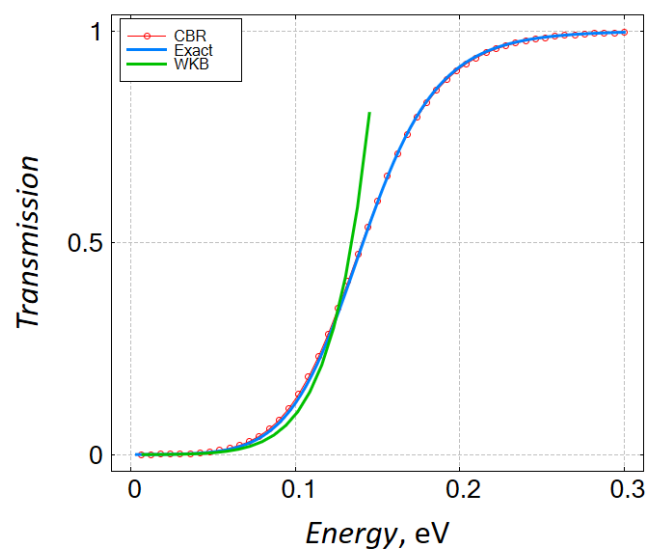


Figure 3.4.3.1: Potential  $V(x)$ , Eq. (3.4.3.1), for which the exact answer for the transmission probability is known, with  $V_0 = 0.15$  eV,  $x_0 = 15$  nm, and  $\alpha^{-1} = 4$  nm. For the consistency with the usual scattering approach [Landau Lifshitz], section §123 *The general theory of scattering*, the potential is finite only in the central region (device),  $0 \leq x \leq 30$  nm and is set to 0 outside of this region. The energy origin (i.e. the bottom of the conduction band) is fixed:  $E_c = 0$ .



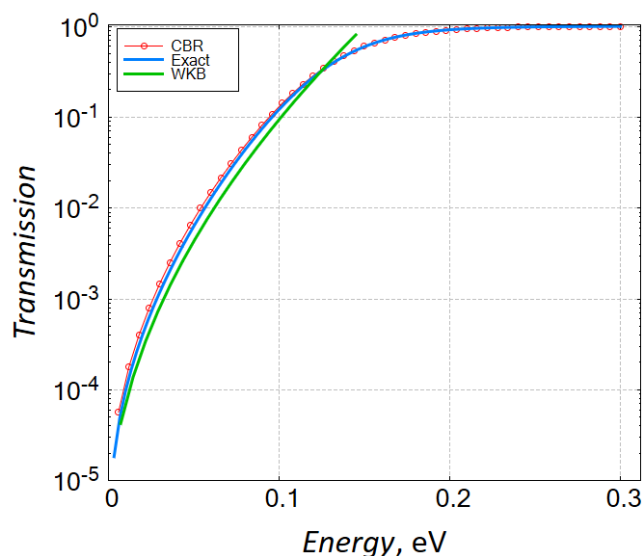


Figure 3.4.3.2: Energy dependence of transmission through the potential shown on Figure 3.4.3.1. The upper (linear scales) and lower (linear-log scales) panels shown answers which are obtained with the help of the CBR (red) and WKB (green) methods. The blue curve is the exact answer, Eq. (3.4.3.2). We have used in the input file the effective mass of the electron in GaAs. Note that finite temperature is irrelevant for these simulations.

#### Exercises:

- Check how the agreement between the exact answer, Eq. (3.4.3.2), and the numerical results of the CBR approach depends on the number of the (device) eigenstates which are taken into account.
- Using the sample file of this tutorial, calculate transmission of an electron through the parabolically shaped potential which exists only inside the device. Compare CBR results with approximate WKB answer. Explain your observations.

### 3.4.4 1D - Quantum Tunneling and Fowler-Nordheim theory

#### Input Files:

*1D\_quantum\_tunneling\_GaAs\_nn3.in*

#### Scope of the tutorial:

- General information about tunneling and tunneling current
- Comparison of the Fowler-Nordheim theory and the CBR-based simulations

#### Main adjustable parameters:

- upper boundary for transmission energy - %E\_max
- the barrier widths - %Delta\_x = %Barrier\_max - %Barrier\_min
- the barrier heights - %Barrier\_Height
- the electric field - %ELECTRIC\_FIELD
- the temperature - %Temperature
- Fermi levels of left ( $x_{\text{min\_contact}} < x < x_{\text{min}}$ ) and right ( $x_{\text{max}} < x < x_{\text{max\_contact}}$ ) regions (leads) - %Fermi\_left and %Fermi\_right
- the effective mass of the electron - %effective\_mass
- dimensionality of the interface which separates leads and connecting wires - %DIM\_LEAD

#### Relevant output Files:

- *Results\BandEdges.dat* (energy profile)
- *Results\Transmission\_cb\_sg1\_deg1.dat* (transmission)
- *Results\LocalDOS\_sg1\_deg1\_Lead1.fld* and *Results\LocalDOS\_sg1\_deg1\_Lead1.fld* (LDoS)
- *Results\IV\_characteristics.dat* (currents)

**Tunneling of particles through a potential barrier** is one of the most well-known quantum phenomenon which does not exist in a classical world. Understanding the quantum tunneling and tunneling currents is very important for engineering and fabricating various elements of electronics and nanodevices, including qubits. There are several seminal theories which describe these effects in various setups, for example, *Tsu-Esaki theory of tunneling in quantum superlattices* [TsuEsaki] and *Fowler-Nordheim theory for field electron emission* [FowlerNordheim] with applications to semiconductor structures [LenzlingerSnow]. This tutorial addresses the Fowler-Nordheim tunneling current in a semiconductor multilayer structure, see Figure 3.4.4.1.

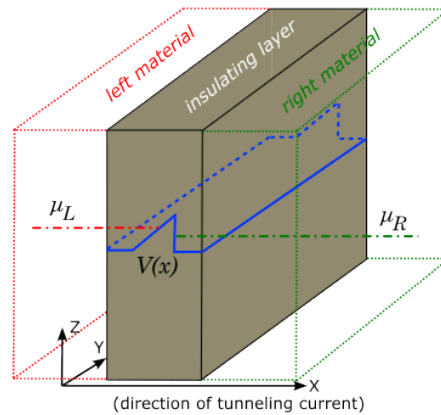


Figure 3.4.4.1: Sketch of a setup for the tunneling current in the Fowler-Nordheim theory. Left (red lines) and right (green lines) materials with different chemical potentials,  $\mu_{L,R}$ , are separated by an insulating layer (brown region) which makes a potential barrier (blue lines). Everything is homogeneous in  $\{y, z\}$ -directions. The tunneling through the potential barrier occurs in  $x$ -direction.

If one considers the tunneling current along one axis and the semiconductor structure is assumed to be homogeneous along the other two directions, simulations can be reduced to effectively one-dimensional Landauer-like setup, see the upper panel of Figure 3.4.4.2, where the device is connected to two leads with different chemical potentials. The current flows from the material with a larger chemical potential to that with a smaller one.

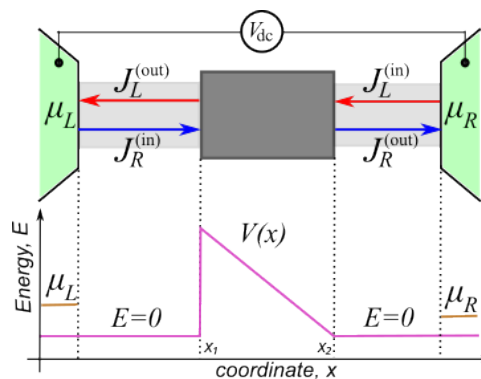


Figure 3.4.4.2: Upper panel: Landauer-like setup. Left and right leads (green regions) are connected to a semiconductor device (dark gray square) via connecting wires (light gray regions). The leads correspond to the left and right materials in Figure 3.4.4.1, hence, they are three-dimensional. Lower panel: chemical potentials of the leads (orange lines) and the energy of the potential barrier (magenta lines).

Following the standard approach, we have introduced wires which connect the device and the leads. These wires are

optional in the current tutorial. The leads correspond to the left and right materials in [Figure 3.4.4.1](#) and, therefore, are three-dimensional semiconductors. Their chemical potentials are different if the left and right materials are different (e.g., differently doped semiconductors). Alternatively, the chemical potentials are shifted by an applied DC voltage,  $\mu_L - \mu_R = eV_{dc}$ , with  $\mu_{L,R}$  being measured from the bottom of the conduction-band,  $E_c = 0$ .

### ! Attention

The value of chemical potentials is not calculated in this tutorial but is set a kind of “artificially”. Of course, this value must be in agreement with physics of a given material. For example, when the temperature (at  $k_B = 1$ ) is smaller than the energy gap separating the conduction and valence bands, the chemical potential of an intrinsic unbiased semiconductor is close to the center of that gap, see e.g. [section 3 The Fermi-Dirac Distribution](#) in [\[Grah\]](#).

The bias  $V_{dc}$  results in the electric current through the device. For simplicity, the connecting wires and the device are treated as purely one-dimensional. The wires are ideal conductors. The device contains a triangular potential barrier, the lower panel of [Figure 3.4.4.2](#), with  $V(x_1) = V_{max}$ ,  $V(x_2) = 0$ . Such a shape of the potential energy can be caused by an electric field,  $F$ :  $V(x) = V_{max} - eF(x - x_1)$ , with  $x_1 \leq x \leq x_2$  and  $F = V_{max}/e(x_2 - x_1)$ ;  $e$  denotes the electron charge. If  $\mu_{L,R} < V_{max}$ , classical transport of the electrons is impossible and one comes across purely quantum current supported by tunneling of the electrons.

The total current is the difference of currents flowing in opposite directions:  $J = J_R^{(in)} - J_L^{(out)} = J_R^{(out)} - J_L^{(in)}$ . Here, upper indices indicate whether a given current flows into or from the device. The Landauer formula allows one to express  $J$  via the transmission probability of tunneling through the barrier,  $\mathcal{T}$ :

$$J = 2e \int \frac{dk_x}{2\pi} v_x(k_x) \mathcal{T}(k_x) (n_L(k_x) - n_R(k_x)), \quad (3.4.4.1)$$

where  $n_{L/R}(k_x) = \int \frac{dk_{y,z}}{(2\pi)^2} f_{L/R}(k_{x,y,z})$  are the electron densities at interfaces between left/right material and the insulating layer;  $v_x$  is the electron velocity along x-axis. The electrons in the left/right materials (i.e., in the leads) are described by the Fermi-Dirac distribution functions,  $f_{L,R}$ . The transmission probability in our simplified model depends only on the longitudinal wave-vector of the electron,  $k_x$ , and is independent on the direction of tunneling (either from the left lead to the right one or vice versa). We assume further that there is no temperature gradient and the temperature,  $T$ , is the same in all parts of the circuit. We note that, in the setup shown in [Figure 3.4.4.1](#),  $J$  is the current per unit transverse cross-section.

$\mathcal{T}$  can be found analytically for the one-dimensional triangular barrier, see, e.g., [section in the book by Shi-Dong Liang \[Shi-Dong-Liang\]](#). In a general case, one can use either approximate analytical methods, like [WKB](#), or calculate the transmission probability numerically by solving the Schrödinger equation. We have used *Contact Block Reduction method [CBR]* for numerical simulations, see also [tutorial](#). Results of the CBR based simulations are shown in [Figure 3.4.4.3](#).

An additional information can be obtained from calculations of the local density of states, see [Figure 3.4.4.4](#) which illustrates, how lead modes enter the device when only one lead is connected. The modes with higher energies have a larger penetration depth.

After integration [\(3.4.4.1\)](#) over the electron momentum (or, equivalently, over the electron energy) one obtains the tunneling current. One example of its IV-characteristics at zero temperature is shown in [Figure 3.4.4.5](#). In the Fowler-Nordheim theory, one usually assumes that the drain (the right lead in the above setup) is the vacuum. It does not contain conduction electrons, i.e., this corresponds to the limit  $\mu_R \rightarrow 0$ . This theory predicts the following tunneling current in isotropic materials:  $J_{FN} \simeq \frac{A_{FN}}{\phi} \exp(-B_{FN}\phi^{3/2})$ . Here  $\phi = V_{max} - \mu_L$ ,  $A_{FN} = \frac{e^3}{16\pi^2\hbar} F^2$ ,  $B_{FN} = -4\sqrt{2m^*}/3\hbar eF$ ;  $\hbar$  and  $m^*$  are the Planck constant and the effective mass of the electron in a given material, respectively.

The disagreement between the constant  $B_{FN}$  and its numerical counterpart, which can be obtained from the fitting of the numerically obtained IV-characteristics, [Figure 3.4.4.5](#), by the Fowler-Nordheim-like equation, is very small; it is close to 2% with the parameters used in this tutorial. However, the constant  $A_{FN}$  substantially underestimates its numerically obtained value. This is related to the approximate calculations of the transmission in the Fowler-Nordheim theory and to the difference between the tunneling from a semiconductor to vacuum and between two different semiconductor materials, [Figure 3.4.4.1](#).

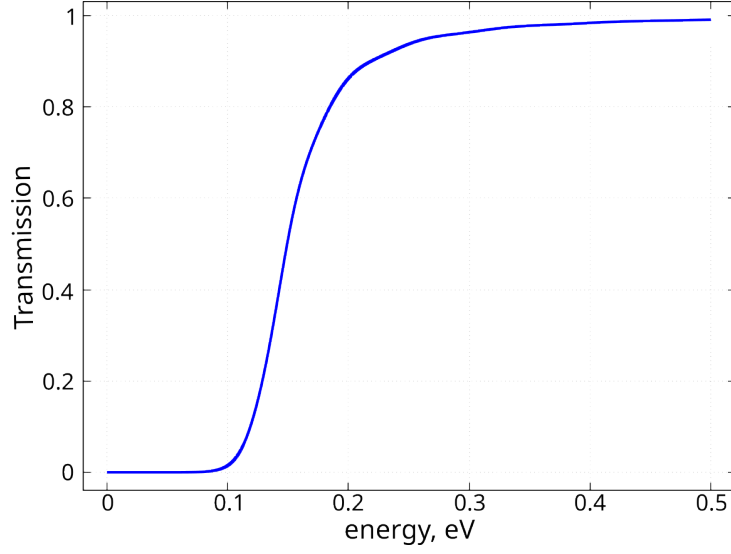


Figure 3.4.4.3: Energy dependence of the transmission probability through a triangular barrier, see Figure 3.4.4.2, with parameters  $V_{max} = 0.161$  eV,  $x_2 - x_1 = 30$  nm. Simulations have been done for the conductors made from GaAs and the potential barrier from  $\text{Al}_x\text{In}_{1-x}\text{As}$ .

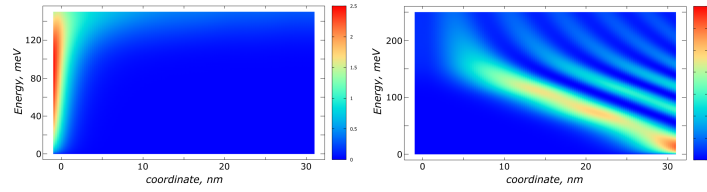


Figure 3.4.4.4: Local Density of states which illustrates how lead modes enter the device when only one lead (the left/right lead in the left/right panel) is connected. Mind the different scale of the color scheme in the left and right panels.

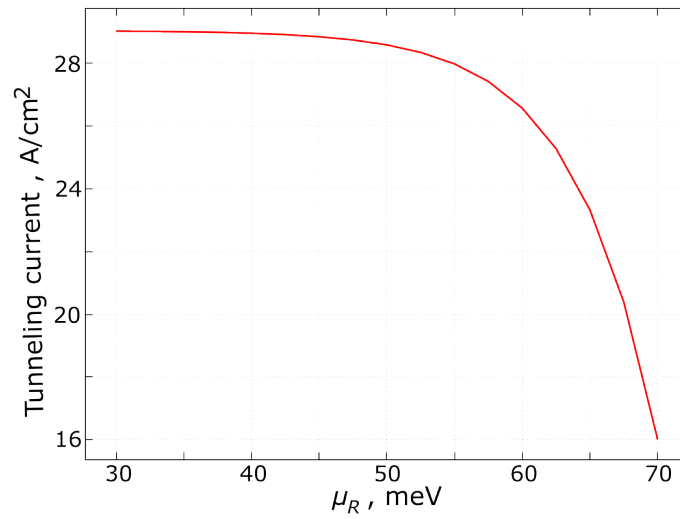


Figure 3.4.4.5: Dependence of the tunneling current on the chemical potential  $\mu_R$  at  $T = 0$  K at fixed value of  $\mu_L = 75$  meV. The DC bias is  $V_{dc} = \mu_L - \mu_R$ . The length of the connecting wires has been chosen as 1 nm. Other parameters are the same as in Figure 3.4.4.3.

The finite temperature changes the energy distribution of the electrons in the leads which results in a temperature-dependent contribution to the tunneling current, see the main panel of Figure 3.4.4.6. The Fowler-Nordheim theory predicts that  $J(T) \simeq J(T=0)(1 + (T/T_c)^2 + O[(T/T_c)^4])$  with  $T_c = \sqrt{3eF\hbar/2\pi k_B} \sqrt{m^*\phi}$ ,  $T/T_c \ll 1$ , and  $k_B$  being the Boltzmann constant.

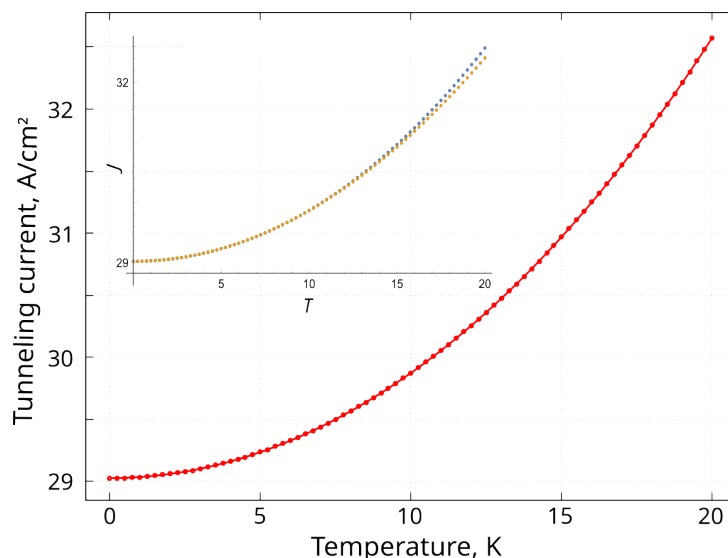


Figure 3.4.4.6: Main panel: Temperature dependence of the tunneling current at  $\mu_R = 30$  meV. Other parameters are the same as in previous Figures. Inset: Comparison of numerically obtained results (blue dots) and their fitting by the quadratic temperature dependence (orange dots) with  $T_c = 58.56$  K

Fitting the numerically obtained result by the quadratic temperature dependence yields  $T_c = 58.56$  K, see the inset of Figure 3.4.4.6 which reasonably agrees with the theoretical value of the Fowler-Nordheim theory  $T_c^{\text{FN}} \simeq 64.4$  K.

#### Exercise:

- Take all parameters from the example shown in Figure 3.4.4.5 but, instead of changing the value of  $\mu_R$ , fix the applied voltage  $V_{\text{dc}}$  and calculate numerically the dependence of the tunneling current on the mean value of the chemical potential,  $\bar{\mu} = (\mu_L + \mu_R)/2$ . Explain your results.

## 3.5 Electrolyte

### 3.5.1 Poisson–Boltzmann equation: The Gouy–Chapman solution

The following *nextnano*<sup>3</sup> input file were used:

- 1DGouyChapman\_template.in
- 2DGouyChapman\_template.in
- 3DGouyChapman\_template.in

We solve the Poisson–Boltzmann equation for a monovalent salt, i.e. NaCl ( $\text{Na}^+ \text{Cl}^-$ ). For this particular case, our numerical solution of the Poisson–Boltzmann equation can be compared to the analytical one-dimensional Gouy–Chapman solution for a monovalent and symmetric salt.

The temperature is set to 298.15 K = 25°C, the static dielectric constant of water is set to 78.

The electrolyte region (0 nm - 200 nm) contains the following ions (*\$electrolyte-ion-content*):

```
!-----
! The electrolyte (NaCl) contains two types of ions:
!   1) 100 mM singly charged cations (Na+)   <- 100 mM NaCl
!   2) 100 mM singly charged anions (Cl-)    <- 100 mM NaCl
```

(continues on next page)

(continued from previous page)

```

!-----
$electrolyte-ion-content

ion-number      = 1          ! singly charged cations
ion-valency      = 1.0        ! charge of the ion: Na+
ion-concentration = 100e-3     ! [M] = [mol/l] = 1d-3 [mol/cm^3]
ion-region       = 0.0 200.0 ! electrolyte region

ion-number      = 2          ! singly charged anions
ion-valency      = -1.0       ! charge of the ion: Cl-
ion-concentration = 100e-3     ! [M] = [mol/l] = 1d-3 [mol/cm^3]
ion-region       = 0.0 200.0 ! electrolyte region

```

We vary the NaCl concentration (ion-concentration) from 0.1 mM to 1 M.

- 0.1 mM
- 1 mM
- 10 mM
- 0.1 M
- 1 M

We assume an interface charge density (*\$interface-states*) between the oxide and the electrolyte of  $-0.2 \text{ C/m}^2 = -124.83 \cdot 10^{12} \text{ cm}^{-2}$ .

```

$interface-states
...
state-number      = 1          ! between contact / electrolyte at 0 nm
state-type        = fixed-charge ! sigma
interface-density = -124.830193e12 ! -0.2 [C/m^2] = -124.830193 * 10^12
-> [cm^-2]

```

The pH value (*\$electrolyte*) is 7, i.e. neutral.

```

$interface-states
...
state-number      = 2          ! between contact / electrolyte at 0 nm
state-type        = electrolyte

```

```

$electrolyte
...
pH-value          = 7.0       ! pH = -lg(concentration) = 7 -> concentration in
-> [M]=[mol/l]

```

The following figure shows the electrostatic potential for different salt concentrations (0.1 mM, 1 mM, 10 mM, 0.1 M and 1 M) at a fixed surface charge of  $-0.2 \text{ C/m}^2$ . The potential at the surface at 0 nm, that arises due to the fixed surface charge density that is in contact with the electrolyte, is screened by the ions in the solution and the resulting distribution of the ions depends on the value of the spatially varying electrostatic potential.

The *Debye screening lengths* (DebyeScreeningLength.txt) are indicated by the squares and the values are:

| NaCl salt concentration | Debye screening length (nm) |
|-------------------------|-----------------------------|
| 1 M                     | 0.303                       |
| 0.1 M                   | 0.959                       |
| 10 mM                   | 3.032                       |
| 1 mM                    | 9.589                       |
| 0.1 mM                  | 30.308                      |

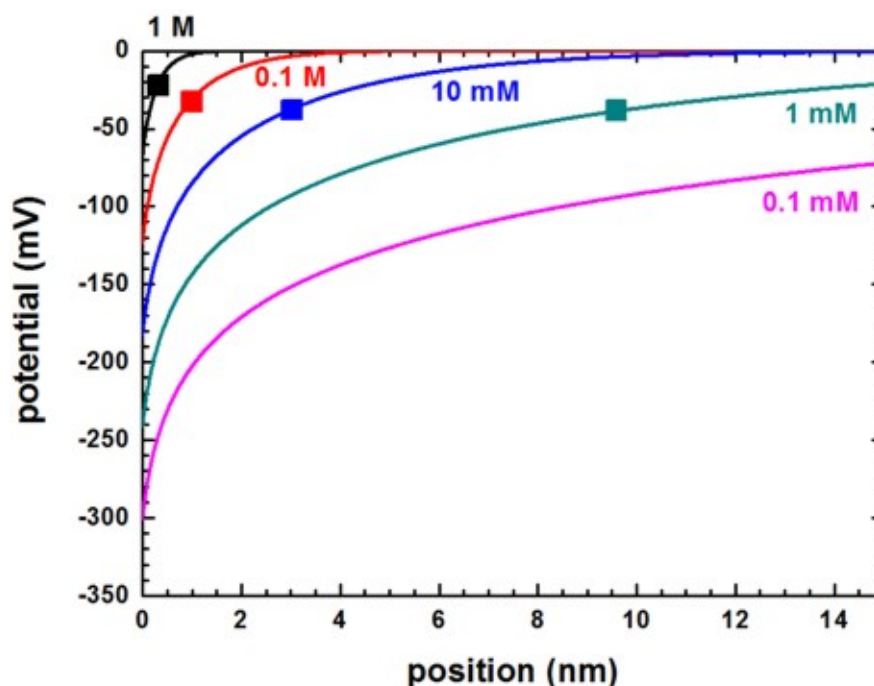


Figure 3.5.1.1: Electrostatic potential for different salt concentrations. The interface is at 0 nm.

**Note**

For 0.1 mM, the concentrations of the  $\text{H}_3\text{O}^+$  and  $\text{OH}^-$  ions slightly influence the Debye screening length (last two digits) because in the *nextnano*<sup>3</sup> simulations these ions are always present and their concentrations depend on the pH value.

For a definition of the Debye screening length, have a look here: [\\$electrolyte](#)

The following figure shows the Debye screening length for a monovalent salt such as NaCl as a function of the salt concentration.

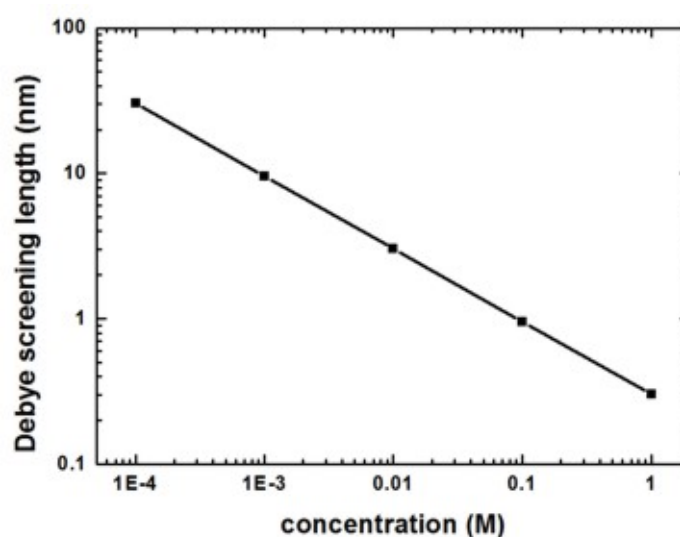


Figure 3.5.1.2: Debye screening length for NaCl as a function of salt concentration

For a monovalent salt the nominal value of the salt concentration is equal to the *ionic strength* which is a measure



for the screening of charges in a solution.

The surface potential can be found in this file: `InterfacePotentialDensity_vs_pH1D.dat`

It reads for a salt concentration of 0.1 M NaCl:

| pH value  | interface potential [V] | interface density ( $1 \cdot 10^{12}$ [e/cm <sup>2</sup> ]) |
|-----------|-------------------------|-------------------------------------------------------------|
| 7.0000000 | -0.123478240580906      | -124.8301930000000                                          |

The following figure shows the ion distribution for a 0.1 M NaCl electrolyte. The multiples of the Debye screening lengths ( $\kappa^{-1} = 0.959$  nm) are indicated by the vertical lines. The negative surface charge is screened by the positive Na<sup>+</sup> ions whereas the negatively charged Cl<sup>-</sup> ions are repelled from the surface. At about 5 nm both ions reach their equilibrium concentration of 0.1 M.

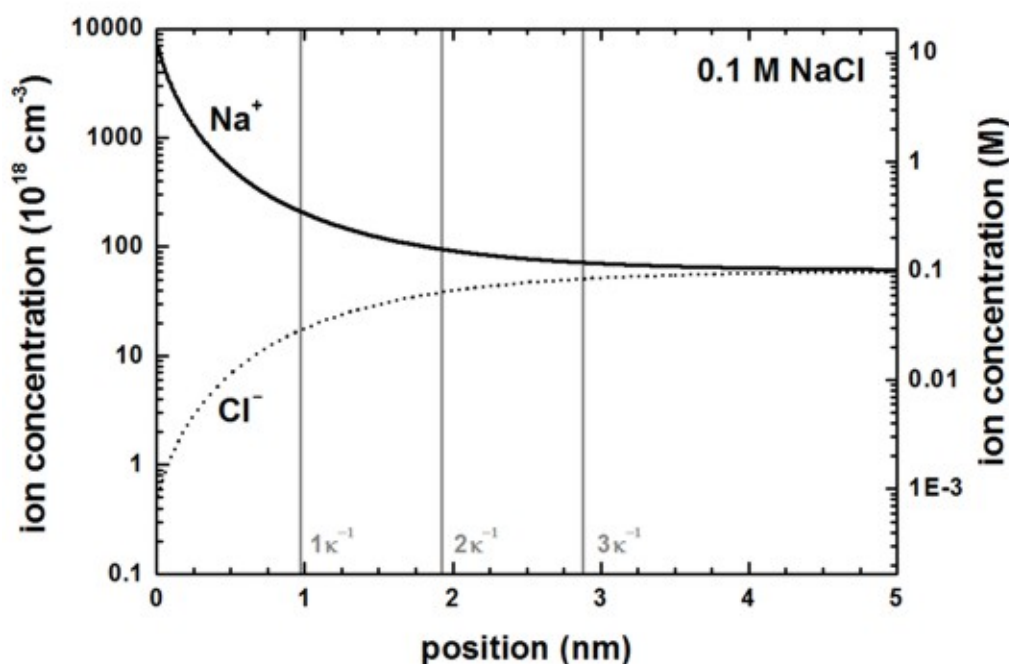


Figure 3.5.1.3: Ion concentrations as a function of distance from the surface at 0 nm

### Linearization of the Gouy–Chapman model

In this approximation which is only valid for high salt concentrations or small surface charges, the surface charge and the surface potential can be related through the basic capacitor equation

$$\sigma_s = \phi_s C_{DL}$$

where  $C_{DL}$  is the capacitance per unit area of the electric double layer.

The following figure shows the surface potential at the solid/electrolyte interface as a function of interface charge for a monovalent salt such as NaCl at different salt concentrations calculated with the Poisson–Boltzmann equation (symbols). The solid lines are the solutions of the analytical Grahame equation for a monovalent salt which relates surface potential to surface charge. The dotted lines are the solutions of the Debye–Hückel approximation for a monovalent salt which relates surface potential to surface charge.

It can be clearly seen that only for high salt concentrations or small surface charges the linearization is valid.

The linearization of the Poisson–Boltzmann equation is called *Debye–Hückel approximation*. It can be switched on using:

```
! electrolyte-equation = Poisson-Boltzmann      ! Poisson-Boltzmann
↪equation (default)
```

(continues on next page)

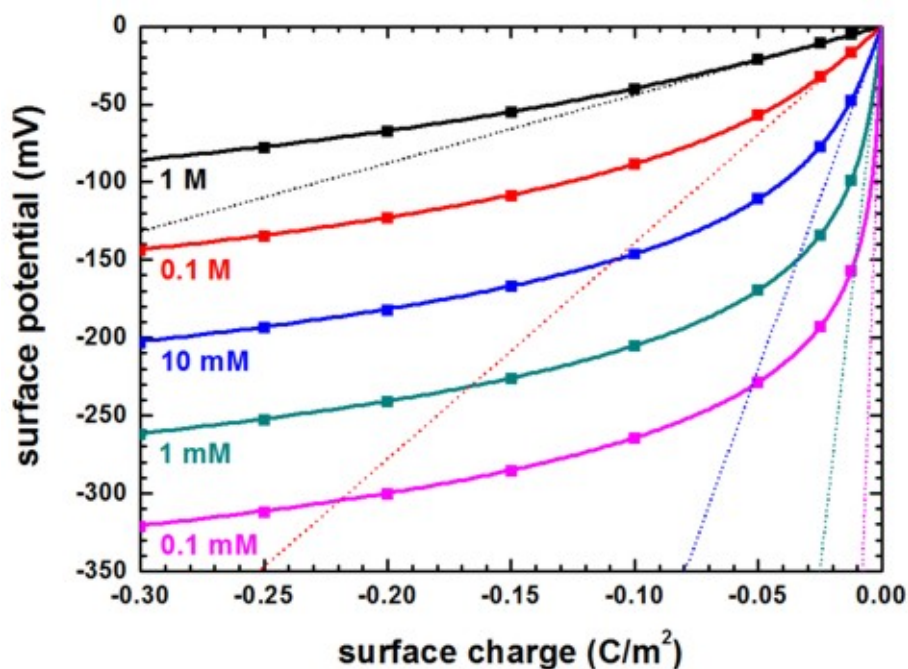


Figure 3.5.1.4: Surface potential vs. surface charge for different salt concentrations

(continued from previous page)

electrolyte-equation = Debye-Hueckel-approximation ! Debye-Hückel-  
 ↳ approximation

## Capacitance

The numerical Poisson–Boltzmann calculations for the capacitance (using the same data as in the previous figure) are shown next. The capacitance increases rapidly for higher potentials but at very small surface potentials, the capacitance is equal to the approximation of parallel plate capacitor model of the electric double layer. This is expected because in the limit of low potentials, the solution of the Poisson–Boltzmann equation must converge to the solution of the Debye–Hückel equation.

## 3.5.2 — DEV — Charge accumulation at the diamond–electrolyte interface

### ⚠ Attention

This tutorial is under construction

#### Input Files:

- 2DHG\_Diamond\_Electrolyte\_Birner\_Stefan\_2011\_1D\_sPM\_nn3
- 2DHG\_Diamond\_Electrolyte\_Birner\_Stefan\_2011\_1D\_ePM\_nn3

#### Relevant output files:

- band\_structure\BandEdges.dat
- band\_structure\potential.dat
- density\density\_hl.dat
- density\density\_space\_charge.dat
- density\density\_ion\_total.dat

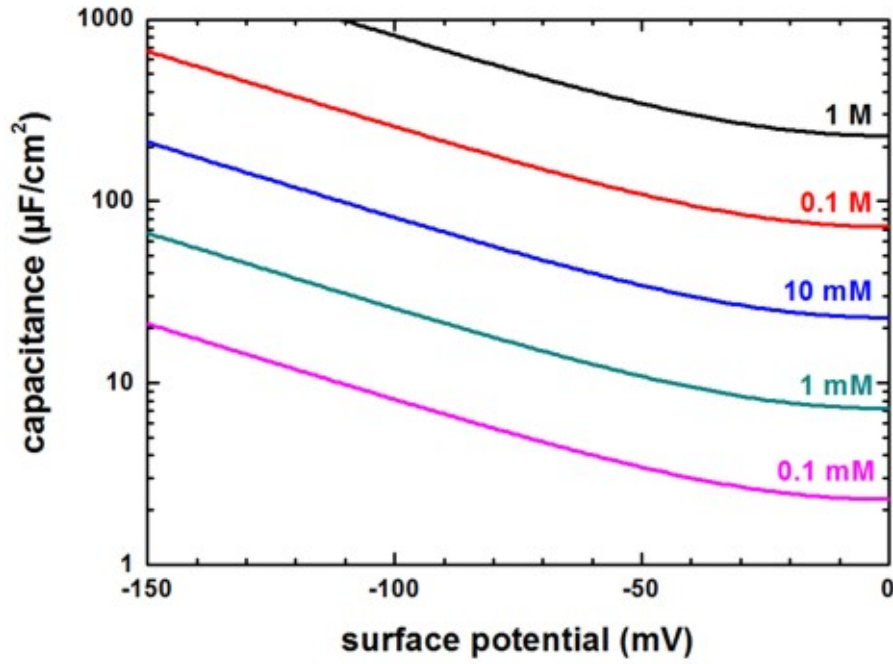


Figure 3.5.1.5: Capacitance vs. surface potential

- `density\density_hl.dat`
- `density\density_ion1ion10_M.dat`
- `Shroedinger_kp\kp6_psi_squared_hl_kpar001_1D_shift.dat`

#### Scope of the tutorial:

- Poisson-Boltzmann equation
- Introduce the PMF factor

This tutorial aims to briefly summarize the results obtained in [\[BirnerPhD2011\]](#) with *nextnano*<sup>3</sup>.

## Introduction

### The standard Poisson-Boltzmann equation

The standard Poisson-Boltzmann (PB) equation, which is described below, enables us to evaluate charge distributions for ions around charged surfaces.

$$\nabla (\epsilon_r \nabla \varphi(x)) = - \sum_i z_i e C_{0,i} \exp \left( - \frac{z_i e (\varphi(x) - U_G)}{k_B T} \right)$$

Here, each variable is shown in the table.

Table 3.5.2.1: variables in the standard Poisson-Boltzmann equation

|              |                                                     |
|--------------|-----------------------------------------------------|
| $\epsilon_r$ | The dielectric constant                             |
| $\varphi(x)$ | The electrostatic potential at a point $x$          |
| $\rho(x)$    | The carrier density at a point $x$                  |
| $z_i$        | The valency of an ion $i$                           |
| $e$          | The elementary electric charge                      |
| $C_{0,i}$    | The bulk concentration of an ion $i$ at a point $x$ |
| $U_G$        | The potential of the reference gate electrode.      |
| $k_B$        | Boltzmann constant                                  |
| $T$          | The temperature                                     |

## The extended Poisson-Boltzmann equation

Refer to these tutorials, the paper, and the thesis for detailed information of this model. [\\$electrolyte](#), [PMF](#), [\[Schwierz2010\]](#), and [\[BirnerPhD2011\]](#).

The standard Poisson-Boltzmann equation in *nextnano*<sup>3</sup> does not consider a spatially varying dielectric constant ( $\epsilon_r(x)$ ), which means that it is equivalent to a homogeneous dielectric constant of water ( $\epsilon_r = 78$ ). On the other hand, the extended Poisson-Boltzmann equation takes into account the spatially varying dielectric constant, and the potential of mean force (PMF) for the electrolyte ions. The equation is given by

$$\nabla(\epsilon_r(x) \nabla \varphi(x)) = - \sum_i z_i e C_{0,i} \exp\left(-\frac{z_i e (\varphi(x) - U_G) + V_{PMF,i}(x)}{k_B T}\right)$$

, where the potential energy term  $V_{PMF,i}(x)$  is the spatially varying potential of mean force of ion species  $i$ .

Instead of assuming a constant value of  $\epsilon_r = 78$  for the dielectric constant of water, a local dielectric constant, depending on  $x$ , for the electrolyte is employed for all PMF calculations.

The local dielectric constant  $\epsilon_r(x)$  to vary in the electrolyte as a function of distance  $x$  from the solid-liquid interface at  $x_0 = 0$  nm is assumed as

$$\epsilon_r(x) = \epsilon_{r,s} + (\epsilon_r^{H_2O} - \epsilon_{r,s}) \frac{\rho^{H_2O}(x)}{\rho_0^{H_2O}}$$

As you can see, the dielectric constant is assumed to be proportional to the water density profile  $\rho^{H_2O}(x)$ .  $\rho_0^{H_2O}$  is the bulk density of water, and  $\epsilon_r^{H_2O}$  is the dielectric constant of water. The density profile is obtained by molecular dynamics simulations approximated by a fitting function.  $\epsilon_{r,s}$  is the relative dielectric constant at the surface of the electrolyte. Therefore, it can become the constant of vacuum ( $\epsilon_{r,s} = 1$ ) if there is a distance of a few Angstrom where there are no ions or water molecules away from the solid-liquid interface (equivalent to the hydrophobic interface).

In this tutorial, we use these Poisson-Boltzmann equations to calculate charge distributions at diamond-electrolyte interfaces and reveal the influence of the hydrophobic interface on the distributions.

## Hydrophobic interaction and charge accumulation at the diamond-electrolyte interface

Refer to the detail of the system used in the simulations in the thesis [\[BirnerPhD2011\]](#). The figure below ([Figure 3.5.2.1](#)) shows the calculated valence band edge energy (black lines) of the hydrogen-terminated diamond-electrolyte interface for an applied gate voltage  $U_G = -0.2$  V.

By adjusting the gate potential applied between the reference electrode in the electrolyte and a contact on the diamond, you can shift the valence band edge with respect to the Fermi level ( $E_F = 0$  eV). This modifies the confinement potential of the resulting triangular-well, and thus also the positive charge density (light blue lines) of the 2DHG in the diamond, leading to a band bending at the surface.

The hole density is mirrored by the corresponding total ion charge density (net negative charge) in the electrolyte indicated in the violet color. The arrow indicates the region of low water density (hydrophobic region) which has a width of approximately 0.3 nm.

The same situation is shown in [Figure 3.5.2.2](#).

In the left part, the three highest eigenstates (shown in the form of their probabilities) at  $k_{||} = 0$  are shown for both the hydrophobic (light green solid lines) and the standard PB approach (purple dotted lines). In fact, the states at  $k_{||} = 0$  are twofold degenerate due to spin, so essentially six states are shown. In the right part, the distribution of the  $Cl^-$  and the  $Na^+$  ions is shown for both models. In the bulk electrolyte, both ions reach their equilibrium concentration of 50 mM. The potential of mean force (PMF) for the  $Na^+$  ions causes the local maximum in the  $Na^+$  ion density profile. For the standard PB model, both the 2DHG and the  $Cl^-$  ions are closer to the interface. The second and the third eigenstate have almost the same energy (also separated by 4 meV for both the hydrophobic and the standard PB model) and the same shape. This is however difficult to see in this figure. The 2DHG sheet charge density for the hydrophobic model is  $\sigma = 3.8 \times 10^{12} \text{ cm}^{-2}$  and for the standard PB model  $\sigma = 10 \times 10^{12} \text{ cm}^{-2}$ .

[Figure 3.5.2.3](#) shows the calculated electrostatic potential and valence band energy (black lines) under the same situation as in the previous figures.

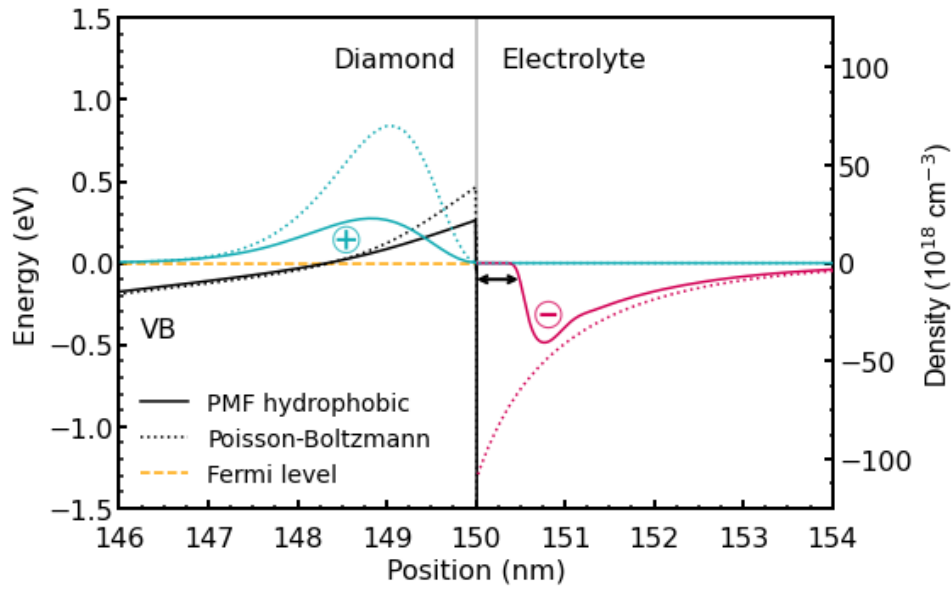


Figure 3.5.2.1: Calculated valence band edge energy (black lines) of the hydrogen-terminated diamond-electrolyte interface for an applied gate voltage  $U_G = -0.2$  V. The density of the accumulated holes (light blue lines) is mirrored by the corresponding total ion charge density (net negative charge) in the electrolyte (violet lines). The results of the standard PB calculation are shown in dotted lines whereas the results obtained with the extended PB model are shown in solid lines. The arrow indicates the region of low water density (hydrophobic region).

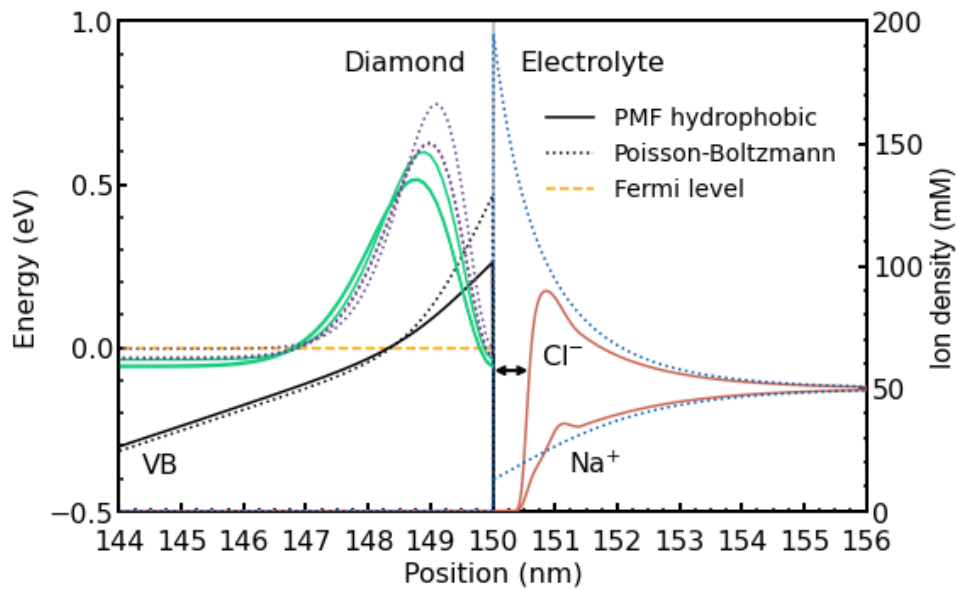


Figure 3.5.2.2: Calculated valence band edge energy (black lines) for an applied gate voltage  $U_G = -0.2$  V. The arrow indicates the hydrophobic region. The probabilities of the three highest eigenstates at  $k_{||} = 0$  are shown in the left part for both models (light green lines and purple dotted lines). In the right part, the distribution of the  $Na^+$  and  $Cl^-$  ions is shown as well. The results of the standard PB calculation are shown in dotted lines whereas the results obtained with the extended PB model are shown in solid lines.

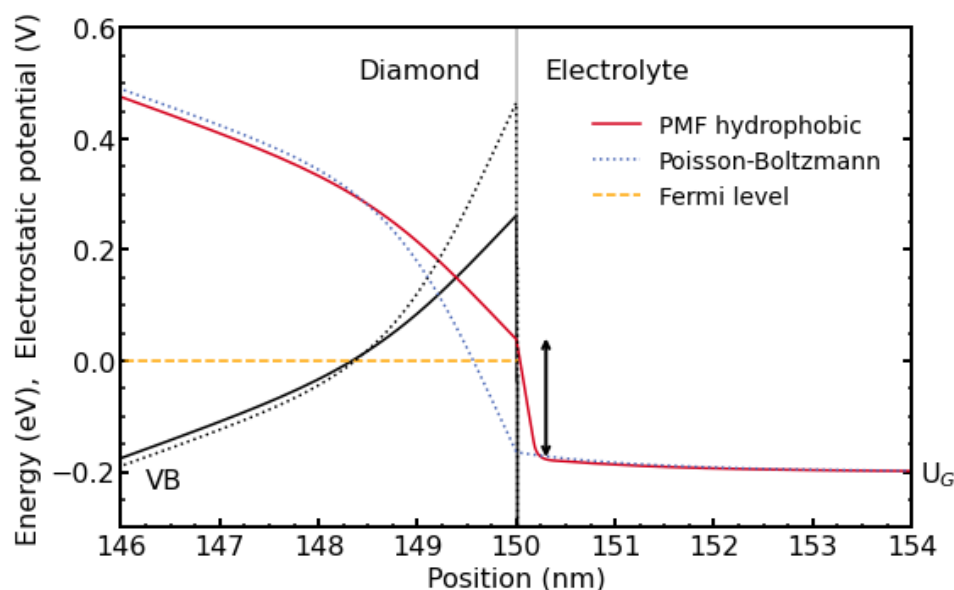


Figure 3.5.2.3: Calculated electrostatic potential and valence band edge (black lines) of the hydrogen-terminated diamond-electrolyte interface under the same situation as in the previous figures. The arrow indicates the large potential drop in the electrolyte region for the hydrophobic model (red solid line). The potential drop in the electrolyte for the standard PB model (blue dotted line) is much smaller. The results of the standard Poisson-Boltzmann calculation are shown in dotted lines whereas the results obtained with the new extended Poisson-Boltzmann model are shown in solid lines.

The applied gate voltage  $U_G$  in the electrolyte cannot be directly related to the electrostatic potential in the diamond because a part of the applied voltage drops in the electrolyte region close to the interface. The electrostatic potential distribution in Figure 3.5.2.3 reveals the potential drop across the diamond-electrolyte interface. There is a striking difference between the two models. The arrow indicates the large potential drop in the electrolyte region for the hydrophobic model (red solid line). The potential drop in the electrolyte for the standard PB model (blue dotted line) is much smaller because here the ions are allowed to approach the interface infinitely close and additionally, the dielectric constant of water is very high even close to the surface. Both effects minimize the potential drop in the electrolyte. Recall that the dielectric constant for the standard PB model has a value of  $\epsilon_r = 78$  up to the interface, whereas the extended PB model assumes a dielectric constant of  $\epsilon_r = 1$  in the hydrophobic ‘gap’ where no ions are present. As a result, the valence band edge for the hydrophobic model is closer to the Fermi level resulting in a lower 2DHG density.

The above results show that the hydrophobicity (or hydrophilicity) of the surface of most biosensor devices has a significant impact on the behavior of the devices, since the sensing signals are generated by the potential change across the interface.

Unfortunately, many *nextnano*<sup>3</sup> tutorials have not been moved yet to the new documentation.

Here is a list of these old tutorials.

- Tutorials indicated by (++) are also available for *nextnano*++.
- Tutorials indicated by (Ma) already moved to the new documentation.

## 3.6 Quantum Mechanics

- (++) (Ma) [Triangular well](#)
- (++) (Ma) [Parabolic quantum well](#)
- (++) (Ma) [Double quantum well](#)
- (++) (Ma) [Dispersion in infinite superlattices: Minibands \(Kronig-Penney model\)](#)
- (++) (Ma) [Quantum corral \(2D\)](#)

- (++) (Ma) Quantum wire (2D)
- (++) (Ma) Artificial atom (3D)
- (++) (Ma) Efficient method for the calculation of ballistic quantum transport - The CBR method (2D example)
- (++) (Ma) Transmission function - NEGF
- (++) (Ma) Transmission through a 3D nanowire (3D)
- (++) (Ma) Exciton correction in 1D quantum wells

## 3.7 Semiconductor Physics

- (++) (Ma) p-n junction
- Electron density in doped semiconductors (Si, Ge, GaAs) - Compensated semiconductors
- Density in n-doped GaAs - Comparison of classical, quantum, k.p and full-band density k.p approach
- (++) (Ma) Poisson equation (for different charge density profiles)

## 3.8 Strain and Piezoelectricity

- (++) Strain: Band shifts and splittings due to conduction and valence band deformation potentials
- (++) (Ma) Band gap of strained AlGaInP on GaAs substrate
- (++) Strain and displacement tensors along different growth directions
- Growth of layers on strained (or stressed) substrates (biaxial and uniaxial)
- strained silicon
- (++) Piezoelectric fields due to strain (Quantum well)
- Piezoelectric field in InAs/GaAs QWs grown along the [111] orientation
- (++) GaN/AlN wurtzite structure: Strain, piezo and pyroelectric charges in wurtzite
- (++) (Ma) Strain effects in freestanding three-dimensional nitride nanostructures (3D)

## 3.9 Magnetic field

- (++) (Ma) Landau levels in bulk GaAs (magnetic field) (2D)
- (Ma) Fock-Darwin states of a 2D parabolic potential (magnetic field) (2D)
- (Ma) Fock-Darwin states of a 2D parabolic, anisotropic (elliptical) potential in a magnetic field (2D)
- Vertically coupled quantum wires in a longitudinal magnetic field (2D)

## 3.10 Heterostructures

- Strained AlAs QW - Crossover of ground states
- (++) (Ma) Schrödinger equation of a two-dimensional core-shell structure Hexagonal 2DEG - Two-dimensional electron gas in a delta-doped hexagonal shaped GaAs/AlGaAs nanowire heterostructure (2D)
- (++) (Ma) Electron and hole wave functions in a T-shaped quantum wire grown by CEO (cleaved edge overgrowth) (2D)
- (++) (Ma) Electron and hole wave functions in a strained T-shaped quantum wire grown by CEO (cleaved edge overgrowth) (2D)
- (++) (Ma) Cubic and cuboidal shaped quantum dots (3D)



- Nanocrystals (3D)
- (++) (Ma) Artificial quantum dot crystal - Superlattice dispersion (minibands) (3D)
- Intermediate-band solar cell (artificial quantum dot crystal) (3D)
- (++) (Ma) Energy levels in a pyramidal shaped InAs/GaAs quantum dot including strain and piezoelectric fields (3D)
- Exciton correction in 2D quantum wires (2D)
- Exciton and biexciton correction in idealistic 3D cubic quantum dots (3D)
- (++) (Ma) QD molecule (3D)
- Cleaved edge overgrowth quantum dots (CEO QDs) (3D)
- (++) (Ma) Hexagonal shaped GaN quantum dot embedded in AlN (wurtzite) (3D)

### 3.11 k.p

- (++) (Ma) k.p dispersion in bulk GaAs (strained / unstrained)
- (++) (Ma) k.p dispersion in bulk GaN (strained / unstrained) (wurtzite)
- (++) (Ma) k.p dispersion in bulk unstrained ZnS, CdS, CdSe and ZnO (wurtzite)
- (++) (Ma) k|| energy dispersion of holes in a GaAs/AlAs quantum well - How to modify database parameters in the input file.
- k|| energy dispersion of holes in unstrained and strained silicon inversion layers
- Electron density of states (DOS) of a GaAs quantum well and Hole density of states (DOS) of a Si hole channel
- Self-consistent 6-band k.p calculations of holes in strained Si/SiGe MOSFETs
- (++) (Ma) Energy dispersion of a cylindrically shaped GaN nanowire (2D)
- (++) (Ma) Artificial atom (Si QD) - 6-band k.p (3D)
- Single-band is a special case of 8-band k.p if the electrons are decoupled from the holes
  - Self-consistent calculation of the quantum mechanical density within the single-band approximation and 8-band k.p
  - Calculation of the quantum mechanical density from the k.p dispersion (no self-consistency)
- 2D Tutorial of the 1D tutorial: Single-band ('effective-mass') is a special case of 8-band k.p ('8x8kp') if the electrons are decoupled from the holes (2D)
  - Calculating the quantum mechanical density within the single-band approximation and 8-band k.p self-consistently
  - Calculation of the quantum mechanical density from the k.p dispersion (no self-consistency)

### 3.12 T2SL

- (++) (Ma) InAs / In<sub>0.4</sub>Ga<sub>0.6</sub>Sb superlattice dispersion with 8-band k.p (type-II band alignment)
- (++) (Ma) InAs / GaSb broken gap quantum well (BGQW) (type-II band alignment)
- HgTe/CdTe quantum well (available on request;  $\mathbf{k} \cdot \mathbf{p}$  and tight-binding)



## 3.13 2DEGs

### 3.13.1 Mobility in two-dimensional electron gases (2DEGs)

Author: Stefan Birner

The input files used in this tutorial are:

- *1DInSb\_mobility\_ShaoFig3.in*
- *1DGaAs\_mobility\_WalukiewiczFig2.in*
- *1DGaAs\_mobility\_WalukiewiczFig3.in*
- *1DInGaAs\_mobility\_WalukiewiczFig8.in*
- *1DGaN\_mobility\_WalukiewiczFig4.in*
- *1DGaN\_mobility\_WalukiewiczFig5.in*

#### Table of Contents

#### Mobility in delta-doped InSb quantum wells

This tutorial is based on the following paper:

- [Carrier Mobilities in delta-doped Heterostructures](#), Y. Shao, S.A. Solin, L.R. Ram-Mohan, arXiv: Materials Science (2006)

Our implementation is based on the equations that are given in this paper (with the exception of Eq. (3.5) where we added a factor of  $\frac{1}{4\pi}$  because of SI units).

We calculate the mobility in a 40 nm InSb quantum well that is surrounded by and strained with respect to  $\text{Al}_{0.15}\text{In}_{0.85}\text{Sb}$  barriers.

At  $z = -20$  nm, there is a delta-doping layer with a sheet doping density of  $1 \times 10^{12} \text{cm}^{-2}$ . The delta-doping layer is separated from the InSb QW by a 40 nm  $\text{Al}_{0.15}\text{In}_{0.85}\text{Sb}$  spacer layer.

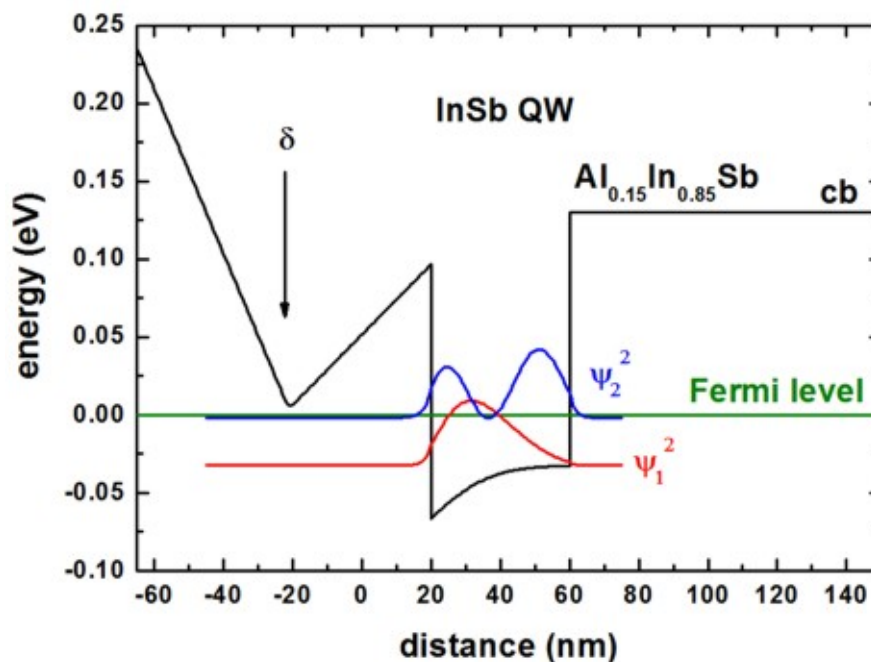
```
quantum-well-width      = 40.0 ! [nm]      [Shao] Fig. 3
spacer-width            = 40.0 ! [nm]      [Shao] Fig. 3
remote-doping-sheet-density = 1e12 ! [cm^-2] [Shao] Fig. 3
```

We calculate all properties for different temperatures. This can be done as follows:

```
$global-parameters
lattice-temperature      = 1.0 ! start value T = 1 [K]
temperature-sweep-active = yes ! 'yes' / 'no'
temperature-sweep-step-size = 10.0 ! increase temperature each time by T = 10.
↪ [K]
temperature-sweep-number-of-steps = 31 ! increase temperature 31 times
data-out-every-nth-step = 1 ! output all data for every temperature.
↪ sweep
$end_global-parameters
```

All output files are labelled with an index, starting from zero, that refers to each individual temperature sweep: `.../..._ind000....dat` Here, the index runs from 0 (000) to 30 (030), i.e. 31 output files for each property in total.

We note that band gaps and lattice constants depend on temperature. This is taken into account automatically for each temperature sweep. The following figure shows the conduction band edge, the Fermi level and the square of the lowest two electron wave functions at  $T = 1$  K.



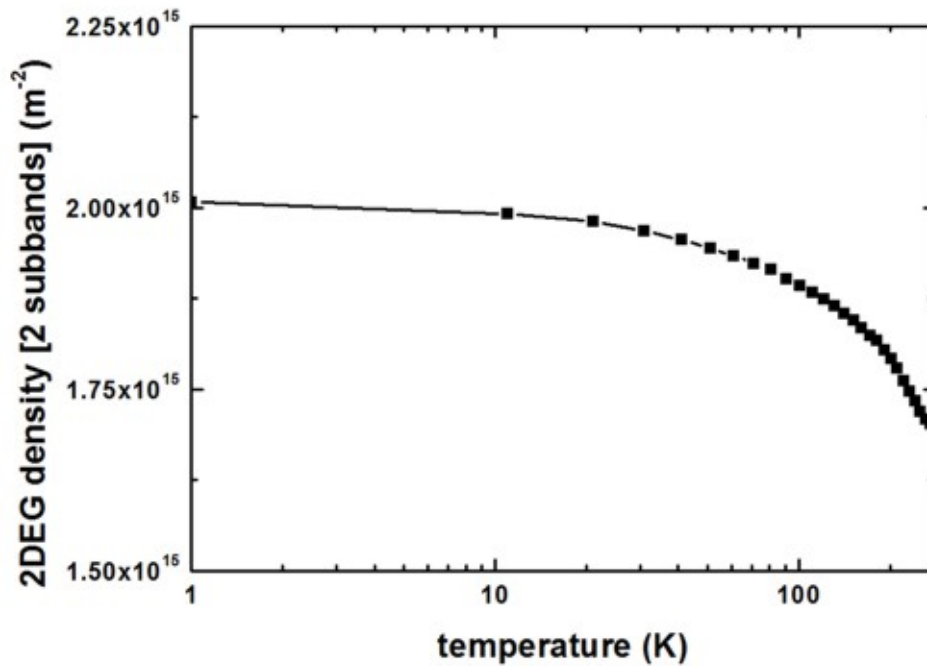
To plot such a figure, the following output files are needed:

- `band_structure/cb1D_001_ind000.dat`
  - 1st column: distance [nm]
  - 2nd column: conduction band edge at the Gamma point [eV]
  - Note: The index ‘ind000’ refers to the temperature sweep. Here, index 0 means  $T = 1$  K.
- `current/fermi1Del_ind000.dat`
  - 1st column: distance [nm]
  - 2nd column: Fermi level of the electrons [eV]
  - In this tutorial, the Fermi level is always equal to 0 eV.
- `Schroedinger_1band/cb001_ind000_sg1_deg1.dat`
  - 1st column: distance [nm]
  - n columns: n energies of the eigenstates [eV]
  - n columns: n squares of the wave functions ( $\psi^2$ ) [eV]
  - In the figure, we plotted the columns for  $\psi^2$  of the two lowest states:  $\psi_1^2$ ,  $\psi_2^2$

The following figure shows the sheet electron density as a function of temperature. We considered the two lowest subbands for calculating the 2DEG density (the spin degeneracy of the subbands is included):

`2DEG-sheet-density-number-of-subbands = 2`

Our results differ from the results of Fig. 3(b) of the Shao’s paper. (Some obvious discrepancies are the conduction band offset (We used  $\sim 0.15$  eV whereas Shao used  $\sim 0.25$  eV.) and the Schottky barrier height.)

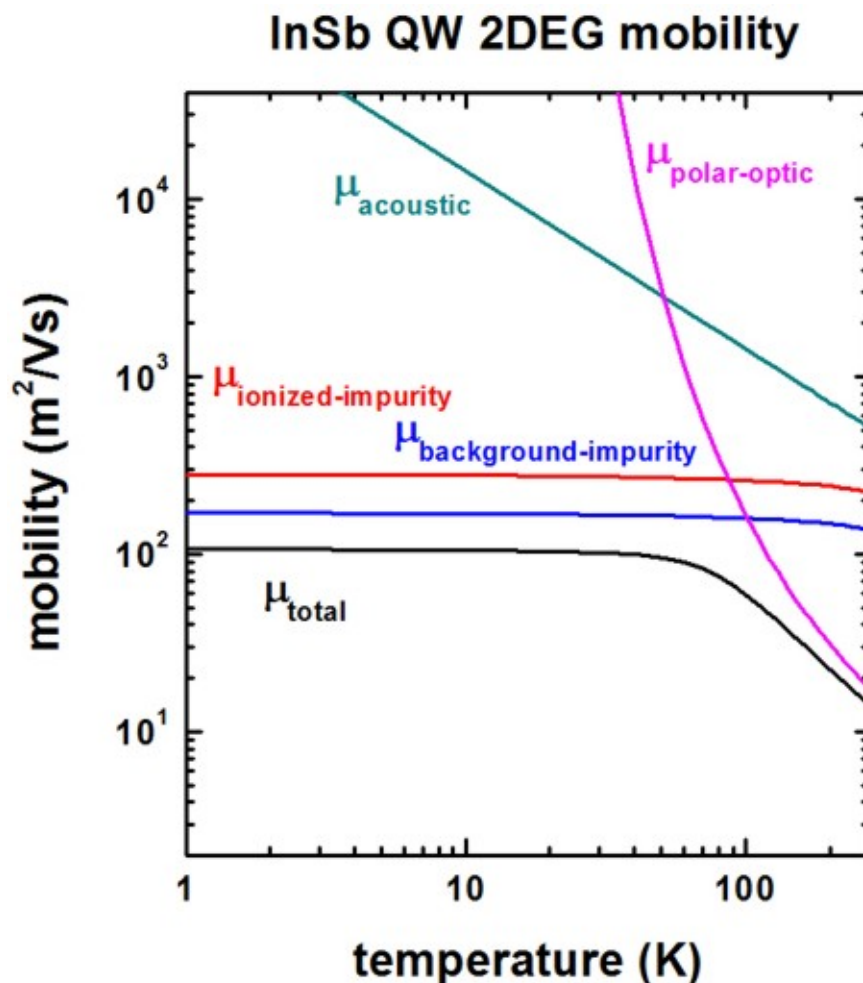


To plot such a figure, the following output file was used:

- Monte\_Carlo/mobility\_TemperatureSweep.dat
  - 1st column: temperature [K]
  - last column: electron sheet density of the lowest subband(s) [ $\text{m}^{-2}$ ]

The following figure shows the calculated 2DEG mobility as a function of temperature. The relevant data can be found in this file:

- Monte\_Carlo/mobility\_TemperatureSweep.dat
  - 1st column: temperature [K]
  - 2nd column: total mobility [ $\text{m}^2/\text{Vs}$ ]
  - 3rd column: mobility due to ionized impurity scattering [ $\text{m}^2/\text{Vs}$ ]
  - 4th column: mobility due to background impurity scattering [ $\text{m}^2/\text{Vs}$ ]
  - 5th column: mobility due to deformation potential acoustic phonon scattering [ $\text{m}^2/\text{Vs}$ ]
  - 6th column: mobility due to polar optic (LO) phonon scattering [ $\text{m}^2/\text{Vs}$ ]



We included the following scattering mechanisms:

```
ionized-impurity-scattering = yes ! [Shao] (including remote and background_
↳ionized impurity scattering)
acoustic-phonon-scattering  = yes ! [Shao]
polar-optical-phonon-scattering = yes ! [Shao]
alloy-scattering            = no  ! [Shao]
```

We now discuss the agreement/disagreement compared to Fig. 3(a) of the [Shao] paper.

- The mobility due to acoustic phonon scattering is in excellent agreement.
- The mobility due to polar optic LO phonon scattering is in excellent agreement if one takes into account that [Shao] forgot to include the factor of  $1/(4\pi)$  due to SI units.
- The mobility due to ionized and background impurity scattering differs significantly. It seems that the disagreement is not only due to the different sheet density that has been used. We used the following value

```
impurity-background-doping-concentration = 5e15 ! [cm⁻³] [Shao] Fig. 3
```

The following InSb material parameters have been used:

```
conduction-band-masses = 0.0135  0.0135  0.0135  ! [m₀] [Shao]
...                      !
static-dielectric-constants = 16.82  16.82  16.82  ! [Shao] epsilon(0)
optical-dielectric-constants = 15.7      ! [Shao] epsilon(infinity)
LO-phonon-energy          = 0.025      ! [eV] [Shao] (optical_
↳phonon energy)
```

(continues on next page)

(continued from previous page)

|                                |          |                        |        |
|--------------------------------|----------|------------------------|--------|
| mass-density                   | = 5.79e3 | ! [kg/m <sup>3</sup> ] | [Shao] |
| sound-velocity                 | = 3.7e3  | ! [m/s]                | [Shao] |
| acoustic-deformation-potential | = 7.2    | ! [eV]                 | [Shao] |

### Mobility in doped GaAs quantum wells

The following input files were used:

- *1DGaAs\_mobility\_WalukiewiczFig2.in* (Experiment of Hiyamizu et al.)
- *1DGaAs\_mobility\_WalukiewiczFig3.in* (Experiment of DiLorenzo et al.)

Here, we test our algorithm to results on GaAs 2DEGs of another publication. We note that our algorithm is suitable for delta-doped 2DEGs but the GaAs examples are not delta-doped.

This section is based on the following paper:

[Walukiewicz]

Electron mobility in modulation-doped heterostructures

W. Walukiewicz, H.E. Ruda, J. Lagowski, H.C. Gatos

Physical Review B 30, 4571 (1984)

The experimental data is based on:

[Walukiewicz, Fig. 2]: [Hiyamizu]

Improved Electron Mobility Higher than 10<sup>6</sup> cm<sup>2</sup>/Vs in Selectively Doped GaAs/N-AlGaAs Heterostructures grown by MBE

S. Hiyamizu, J. Saito, K. Nanbu

Japan. J. Appl. Phys. 22, L609 (1983)

[Walukiewicz, Fig. 3]: [DiLorenzo]

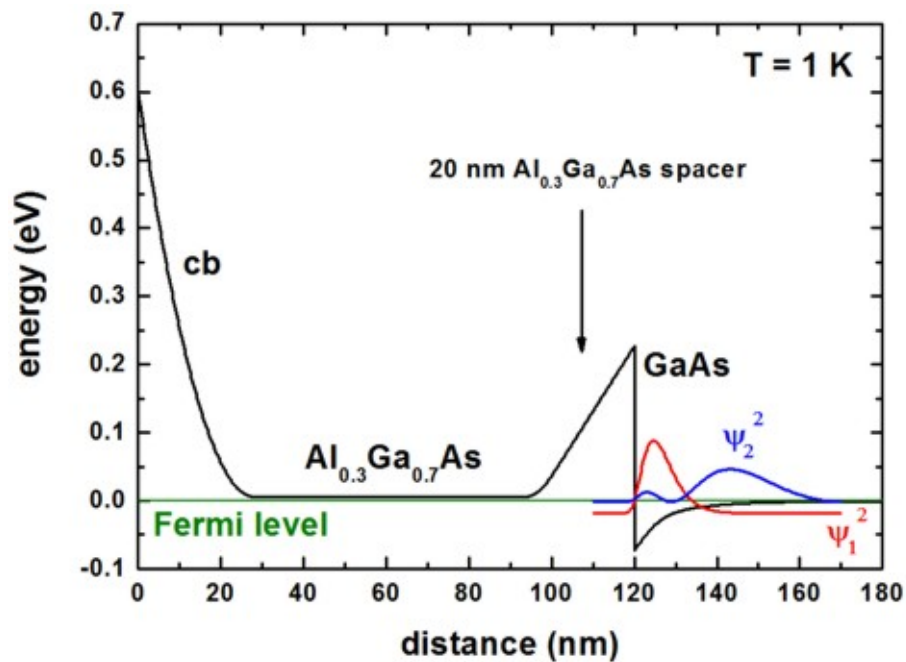
Material and device considerations for selectively doped heterojunction transistors

J.V. DiLorenzo, R. Dingle, M. Feuer, A.C. Gossard, R. Hendel, J.C.M. Hwang, A. Kastalsky, V.G. Keramidas, R.A. Kiehl, P. O'Connor

IEEE-IEDM (International Electron Devices Meeting) 28, 578 (1982)

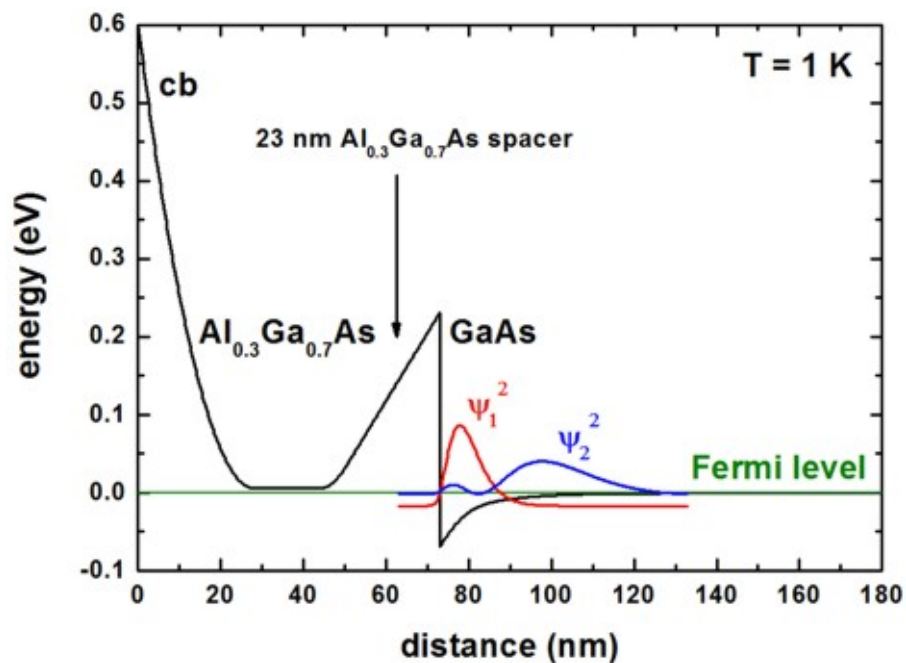
## Conduction band profile and wave functions

[Walukiewicz, Fig. 2]: [Hiyamizu]



```
! 20 nm Al0.3Ga0.7As spacer
spacer-width = 20.0    ! 20 [nm]
```

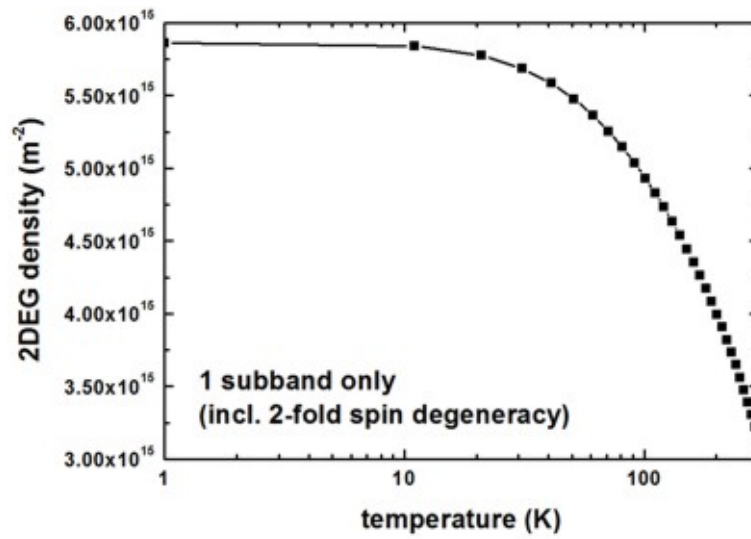
[Walukiewicz, Fig. 3]: [DiLorenzo]



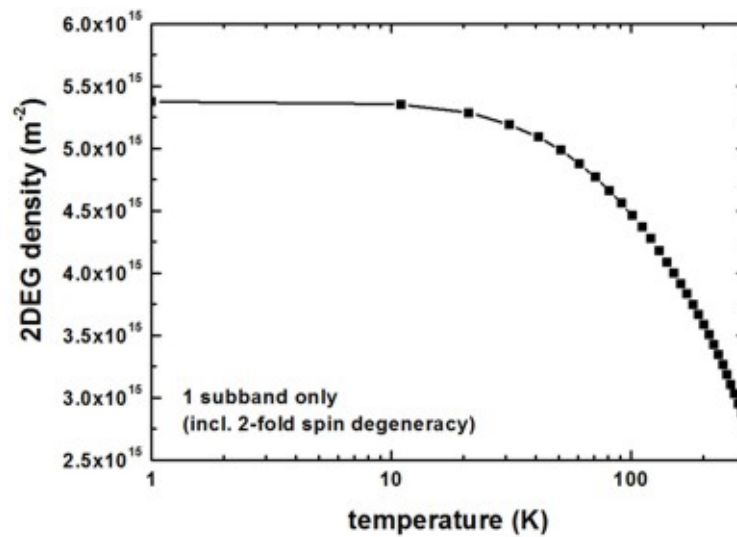
```
! 23 nm Al0.3Ga0.7As spacer
spacer-width = 23.0    ! 23 [nm]
```

## 2DEG sheet density

[Walukiewicz, Fig. 2]: [Hiyamizu]

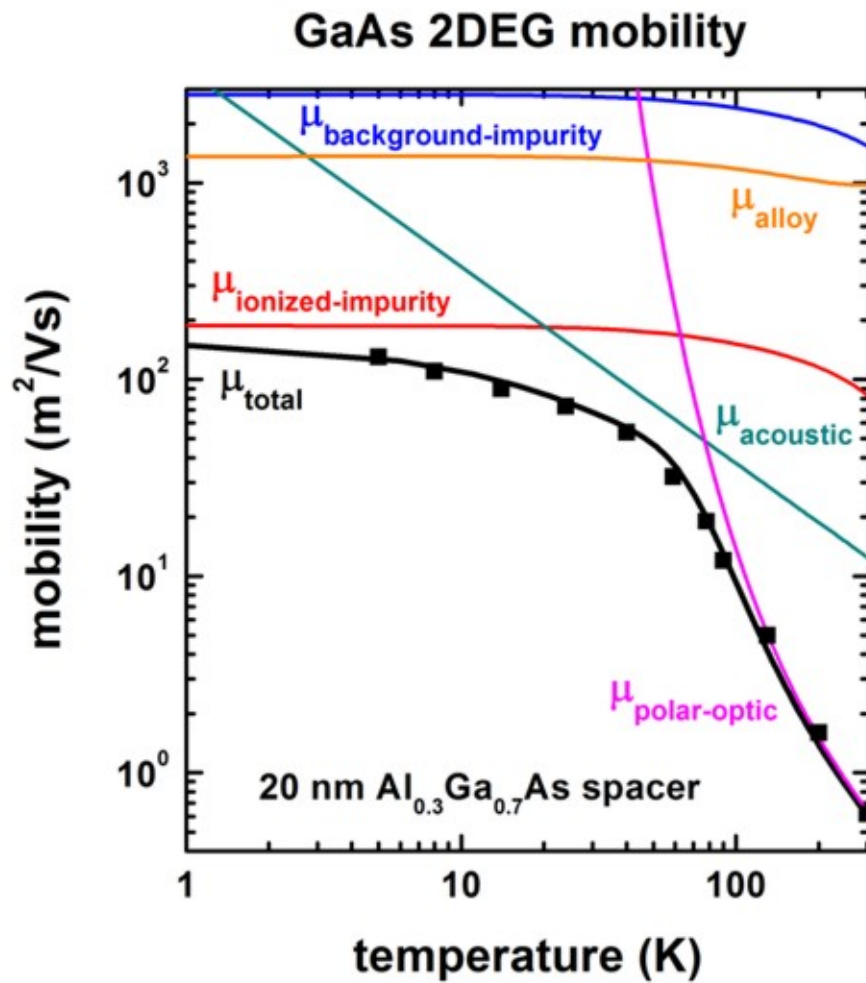


[Walukiewicz, Fig. 3]: [DiLorenzo]



## Mobility

[Walukiewicz, Fig. 2]: [Hiyamizu]



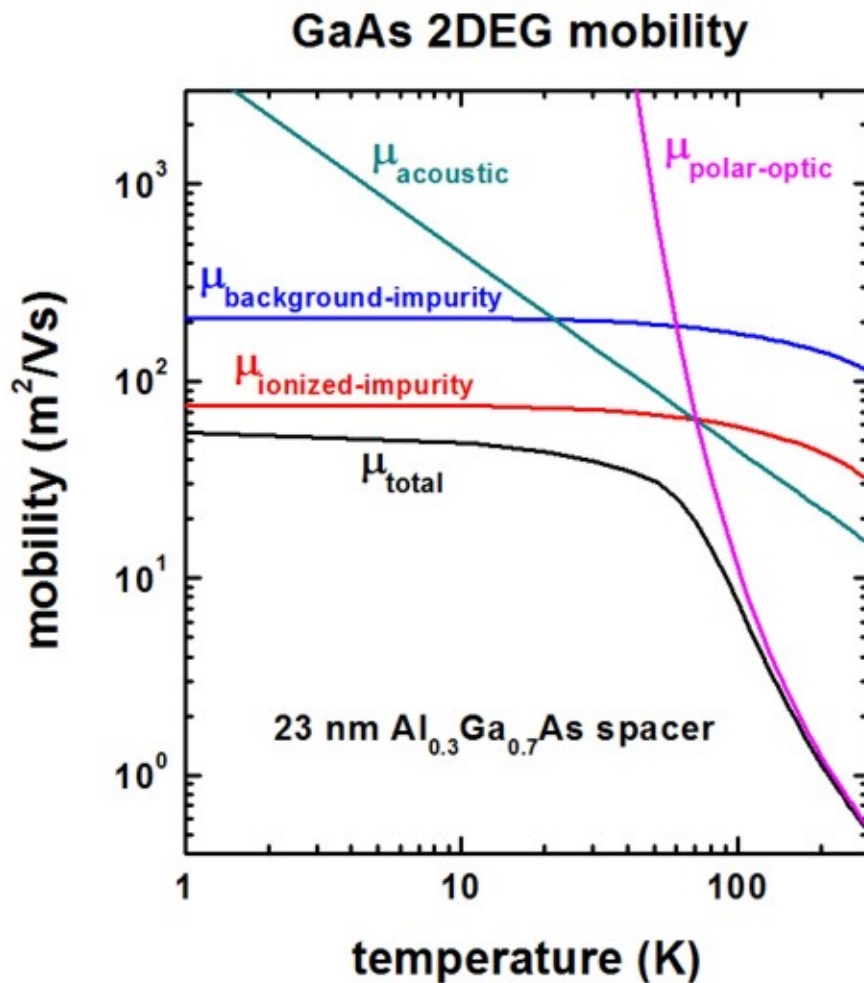
```

impurity-background-doping-concentration = 9e13 ! [cm-3]
remote-doping-sheet-density              = 3.5e11 ! [cm-2] (to fit experiment)
(remote-doping-sheet-density             = 1.948344e11 ! [cm-2])
! [Walukiewicz] Fig. 2: 8.6 * 1016 [cm-3] ==> 8.6 * 1016 [cm-3]2/3 = 1.948344e11 (?)

```

[Walukiewicz, Fig. 3]: [DiLorenzo]





```
impurity-background-doping-concentration = 1e15 ! [cm-3]
remote-doping-sheet-density              = 1e12 ! [cm-2]
```

#### Differences with respect to Walukiewicz [Hiyamizu] paper

Walukiewicz used a 2DEG density of  $3 \times 10^{11} \text{ cm}^{-2}$ .

```
conduction-band-masses = 0.067  0.067  0.067  ! [m0]
...                                     ! a higher value than for bulk_
↪ because of nonparabolicity
```

Here we used 0.067 as this gives better agreement to the mobility at higher temperatures and this is the usually accepted material parameter for GaAs.

```
quantum-well-width = 13.0 ! [nm] (13 nm seems to be a better value than 20 nm.)
alloy-scattering   = yes
```

Alloy scattering is relevant for the part of the wave function that penetrates into the AlGaAs barrier.

The squares are experimental values of Fig. 5 in:

Improved Electron Mobility Higher than  $106 \text{ cm}^2/\text{Vs}$  in Selectively Doped GaAs/N-AlGaAs Heterostructures Grown by MBE  
S. Hiyamizu et al.

Jpn. J. Appl. Phys. 22, L609 (1983)

The following GaAs material parameters have been used:

```
static-dielectric-constants = 12.9    12.9    12.9    ! [Walukiewicz] epsilon(0)
optical-dielectric-constants = 10.9                                ! [Walukiewicz]
↪epsilon(infinity)
LO-phonon-energy             = 0.036                                ! [eV] [Walukiewicz]
↪(optical phonon energy)
mass-density                 = 5.318e3                             ! [kg/m^3] [Davies] p. 411
sound-velocity               = 5.29e3                             ! [m/s] [X.L. Lei, J.
↪Phys. C 18, L593 (1985)]
acoustic-deformation-potential = 7.0                               ! [eV] [Walukiewicz]
```

Into the equation for the **deformation potential acoustic phonon scattering**, the quantum well width is an input parameter. We used a value of 13 nm which corresponds roughly to the extension of the ground state wave function inside the “triangular” QW.

Differences with respect to Walukiewicz [DiLorenzo] paper

In Walukiewicz’s paper, the background impurity scattering is dominating the ionized impurity scattering. We found the opposite.

Walukiewicz used a 2DEG density of  $2.2 - 3.8 \times 10^{11} \text{ cm}^{-2}$

```
conduction-band-masses = 0.076    0.076    0.076    ! [m0] [Walukiewicz]
...                                     ! a higher value than for bulk
↪because of nonparabolicity. This is also the value used by Walukiewicz.

quantum-well-width = 20.0 ! [nm] (This value might be too large. See left where 13 nm
↪was used.)

alloy-scattering      = no
```

## Mobility in doped InGaAs quantum wells

The input file used is:

- *1DInGaAs\_mobility\_WalukiewiczFig8.in* (Experiment of Kastalsky et al.)

Here, we test our algorithm to results on InGaAs 2DEGs of another publication. We note that our algorithm is suitable for delta-doped 2DEGs but the InGaAs examples are not delta-doped.

This tutorial is based on the following paper:

[Walukiewicz]

Electron mobility in modulation-doped heterostructures

W. Walukiewicz, H.E. Ruda, J. Lagowski, H.C. Gatos

Physical Review B 30, 4571 (1984)

The experimental data is based on:

[Walukiewicz, Fig. 8]: [Kastalsky]

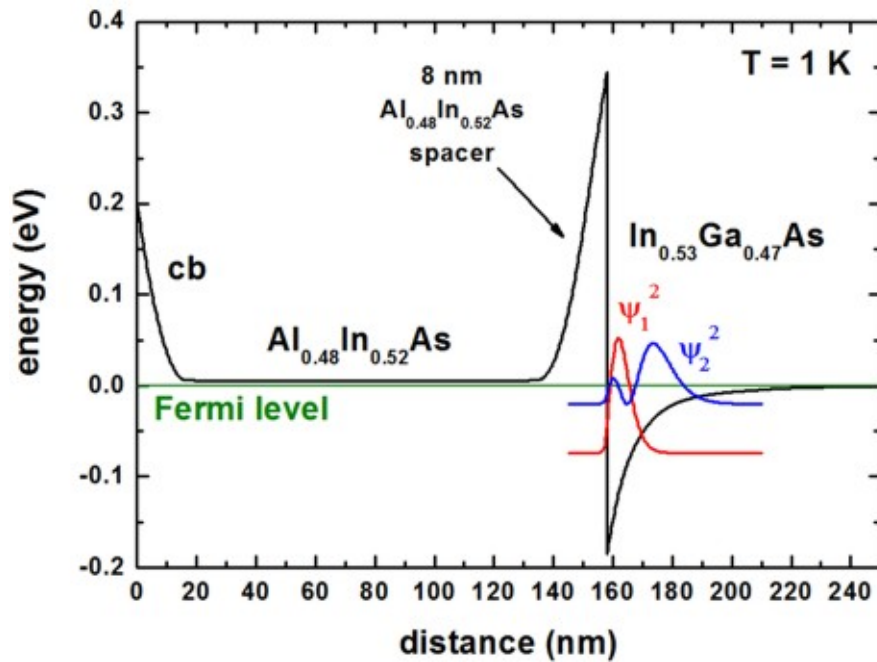
Two-dimensional electron gas at a molecular beam epitaxial-grown, selectively doped,  
In<sub>0.53</sub>Ga<sub>0.47</sub>As-In<sub>0.48</sub>Al<sub>0.52</sub>As interface

A. Kastalsky, R. Dingle, K.Y. Cheng, A.Y. Cho

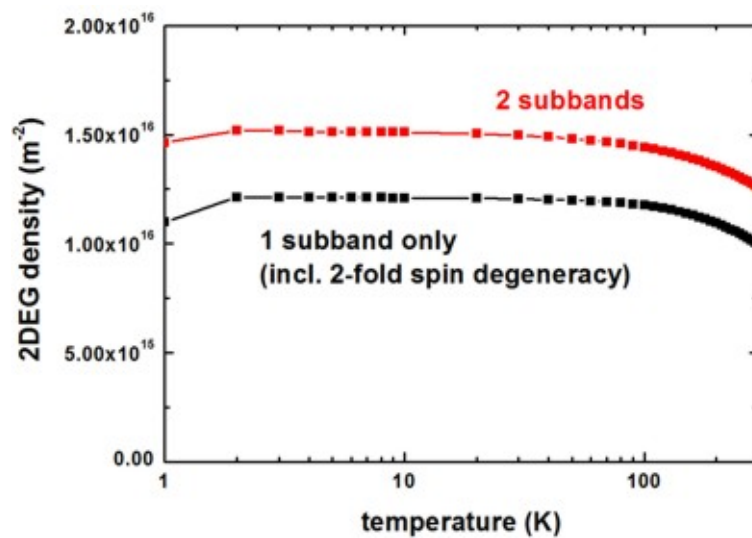
Applied Physics Letters 41, 274 (1982)

[Kastalsky] and [Walukiewicz] have used In<sub>0.52</sub>Al<sub>0.48</sub>As whereas we used In<sub>0.52</sub>Al<sub>0.48</sub>As which is lattice matched to InP and In<sub>0.53</sub>Al<sub>0.47</sub>As.

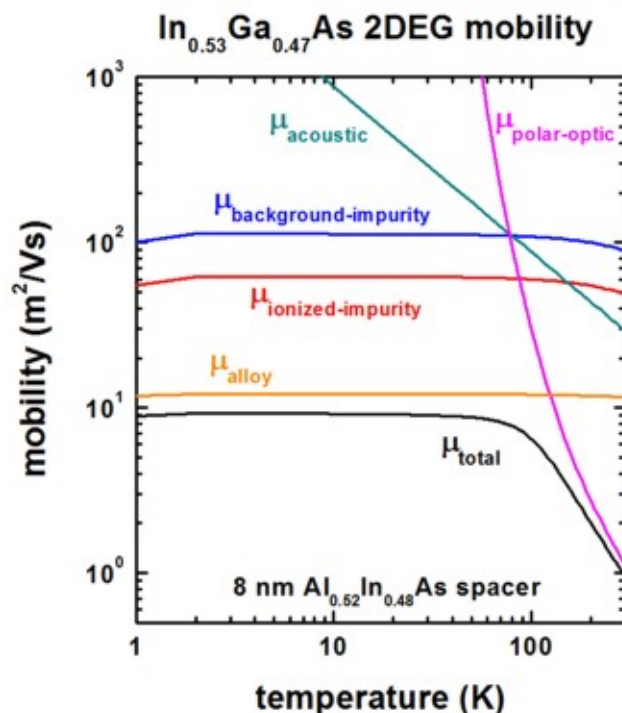
The conduction band profile is shown in the following figure. Here, two subbands ( $\psi_1^2$ ,  $\psi_2^2$ ) are occupied although our implementation of calculating the mobility is only applicable to one occupied subband. Note that the 2DEG is located in an alloy, i.e. InGaAs. Thus we expect that alloy scattering has a significant effect on the total mobility.



The following figure shows the subband density of the **first subband** and of the first two subbands as a function of temperature. Inside the mobility algorithm, only the density of the first subband has been considered.



The following figure shows the mobility as a function of temperature. At temperatures below 100 K, the total mobility is dominated by alloy scattering.



Our results for the mobility are in reasonable agreement with Fig. 8 of the paper of [Walukiewicz].

The following parameters were used:

```

ionized-impurity-scattering = yes      ! (including remote and background_
↳ionized impurity scattering)
acoustic-phonon-scattering  = yes      !
polar-optical-phonon-scattering = yes  !
alloy-scattering            = yes      !
quantum-well-width          = 15.0     ! [nm] 15 nm seems to be a_
↳reasonable approximation for the triangular well
spacer-width                = 8.0      ! [nm]
impurity-background-doping-concentration = 1.0e16 ! [cm-3]
remote-doping-sheet-density  = 1.0e12  ! [cm-2]
2DEG-sheet-density-number-of-subbands = 1
alloy-disorder-scattering-potential = 0.60 ! [eV] InGaAs bulk value [J.R._
↳Hayes et al. (1982)]

!-----
! In0.53Ga0.47As material parameters
!-----
mass-density                 = 5.5025e3 ! [kg/m^3] InGaAs [Wen et al.,_
↳JAP 100, 103516 (2006)]
sound-velocity               = 4.753e3  ! [m/s] In0.53Ga0.47As[111] [Wen et al.,
↳ JAP 100, 103516 (2006)]
acoustic-deformation-potential = 7.0    ! [eV] InGaAs [Walukiewicz]

```

### Mobility in doped GaN quantum wells

The input files used are:

- 1DGaN\_mobility\_WalukiewiczFig4.in
- 1DGaN\_mobility\_WalukiewiczFig5.in

Here, we test our algorithm to results on GaN 2DEGs of another publication. We note that our algorithm is suitable for delta-doped 2DEGs but the GaN examples are not delta-doped.

This tutorial is based on the following paper:

[WalukiewiczGaN]

Electron mobility in  $\text{Al}_x\text{Ga}_{1-x}\text{N}/\text{GaN}$  heterostructures

L. Hsu, W. Walukiewicz

Physical Review B 56, 1520 (1997)

Note: To be consistent with the paper of Walukiewicz, no (!) piezo- and pyroelectricity is included.

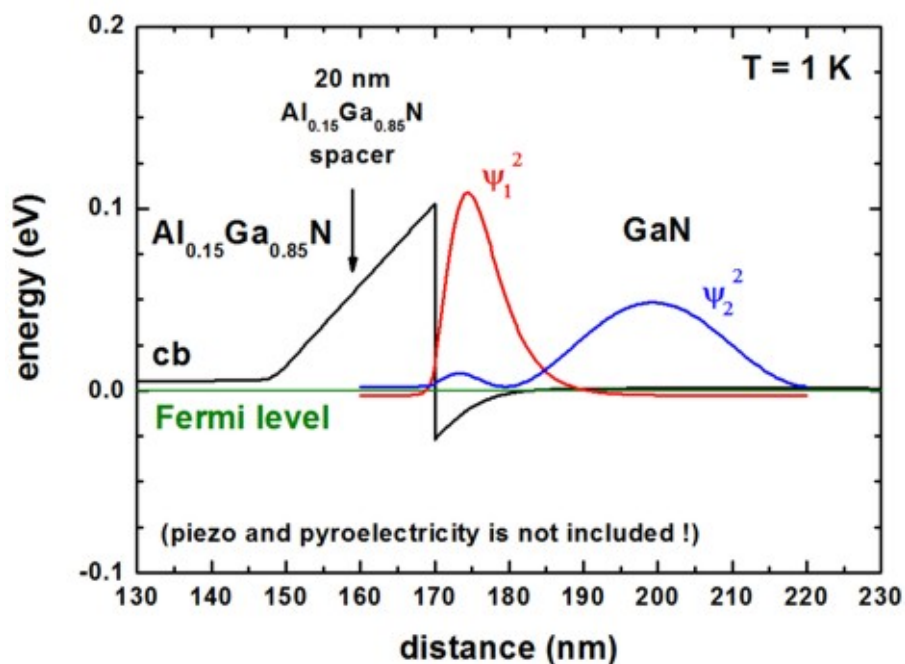
The following GaN material parameters have been used:

|                                     |          |        |                                |                         |
|-------------------------------------|----------|--------|--------------------------------|-------------------------|
| conduction-band-masses              | = 0.21   | 0.21   | 0.21                           | ! [m0] [WalukiewiczGaN] |
| static-dielectric-constants         | = 9.5    | 9.5    | 9.5                            | ! [WalukiewiczGaN]      |
| ↪epsilon(0)                         |          |        |                                |                         |
| optical-dielectric-constants        | = 5.35   | 5.35   | 5.35                           | ! [WalukiewiczGaN]      |
| ↪epsilon(infinity)                  |          |        |                                |                         |
| LO-phonon-energy                    | = 0.0905 | 0.0905 | 0.0905                         | ! [eV] [WalukiewiczGaN] |
| ↪(optical phonon energy)            |          |        |                                |                         |
| mass-density                        | = 6.1e3  |        |                                | ! [kg/m^3]              |
| ↪[WalukiewiczGaN]                   |          |        |                                |                         |
| sound-velocity                      | = 6.6e3  |        |                                | ! [m/s]                 |
| ↪[WalukiewiczGaN]                   |          |        |                                |                         |
| acoustic-deformation-potential      | = 8.5    |        |                                | ! [eV]                  |
| ↪[WalukiewiczGaN]                   |          |        |                                |                         |
| alloy-disorder-scattering-potential | = 2.3    | ! [eV] | conduction band offset GaN/AlN |                         |

Here, alloy scattering is only relevant for the part of the wave function that penetrates into the AlGaN barrier.

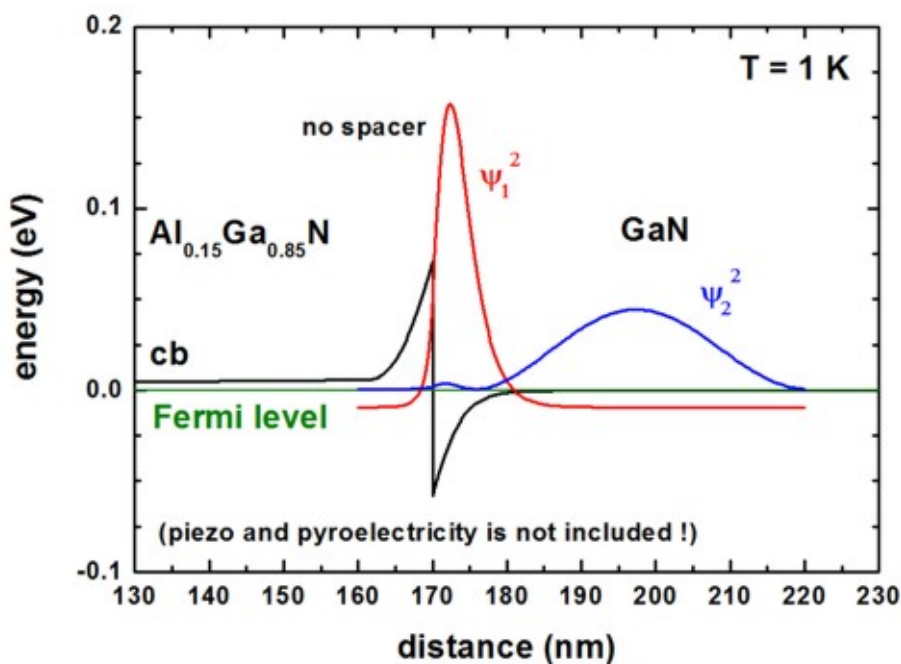
### Conduction band profile and wave functions [WalukiewiczGaN]

[WalukiewiczGaN] Fig. 4



```
! 20 nm Al0.15Ga0.85N spacer
spacer-width = 20.0 ! 20 [nm]
```

[WalukiewiczGaN] Fig. 5

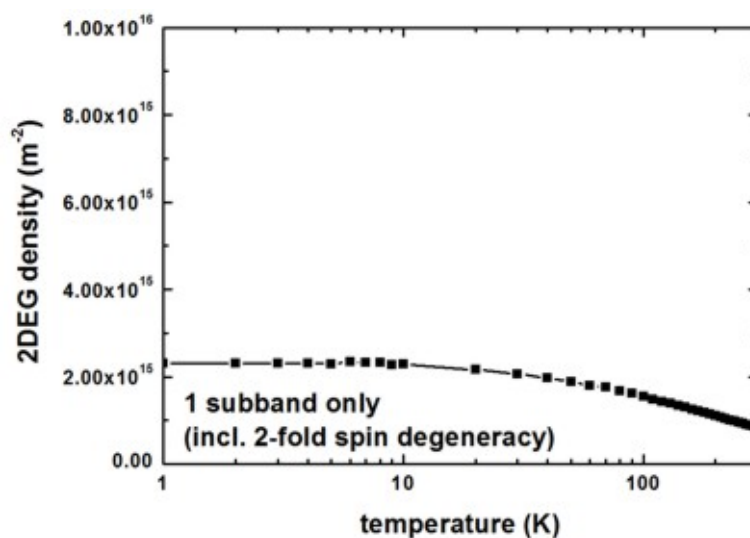


```
! no spacer
spacer-width = 0.0 ! 0 [nm]
```

## 2DEG sheet density

[WalukiewiczGaN] Fig. 4

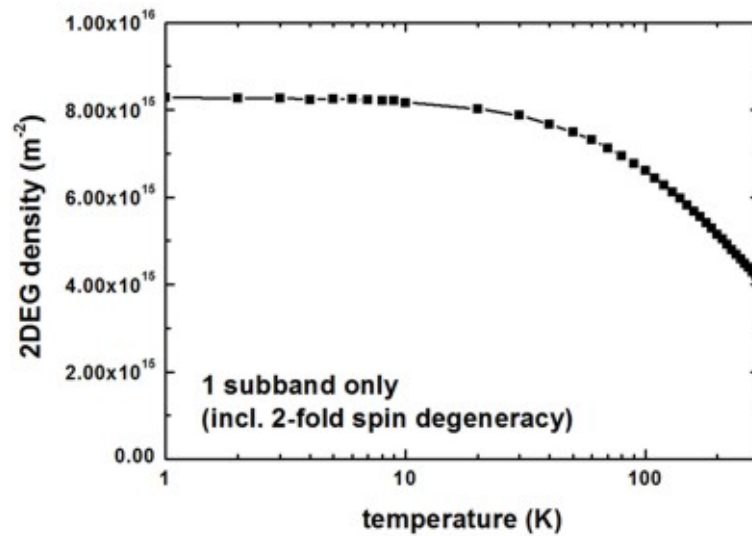
Into the equation for the **deformation potential acoustic phonon scattering**, the quantum well width is an input parameter. We used values which correspond roughly to the extension of the ground state wave function inside the “triangular” QW.



[WalukiewiczGaN] used a 2DEG density of  $6.2 \cdot 10^{15} \text{ m}^{-2}$ .

```
quantum-well-width = 15.0 ! 15 [nm]
```

[WalukiewiczGaN] Fig. 5

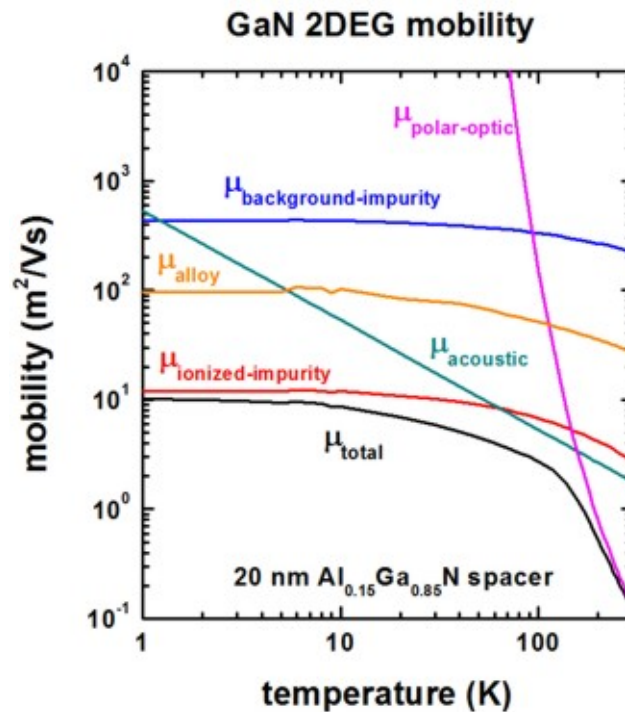


[WalukiewiczGaN] used a 2DEG density of  $1.59 \times 10^{16} \text{ m}^{-2}$ .

```
quantum-well-width = 10.0 ! 10 [nm]
```

## Mobility

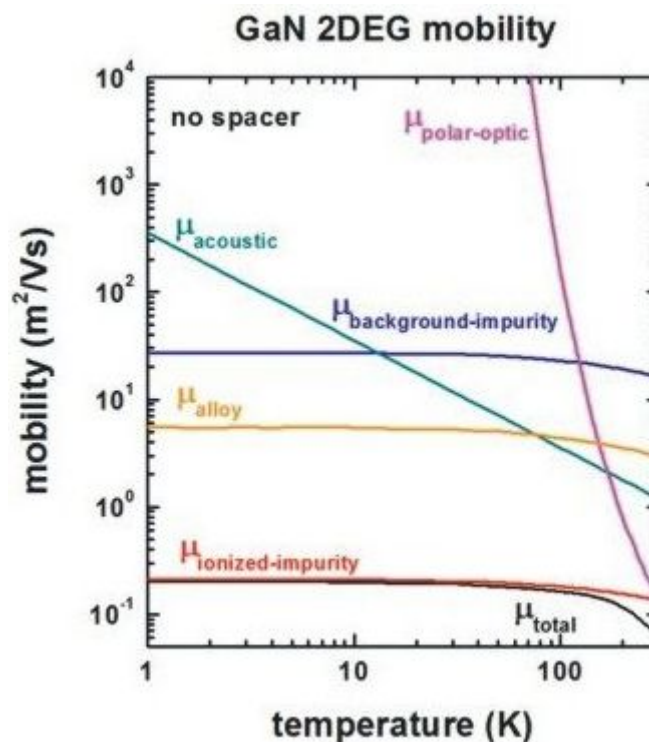
[WalukiewiczGaN] Fig. 4



```
impurity-background-doping-concentration = 1e14 ! [cm-3]
remote-doping-sheet-density = 7.883735e11 ! [cm-2]
! [WalukiewiczGaN] Fig. 4:  $7 \times 10^{17} \text{ [cm-3]} \Rightarrow 7 \times 10^{17} \text{ [cm-3]}^{2/3}$ 
```

[WalukiewiczGaN] Fig. 5





```

impurity-background-doping-concentration = 4e15 ! [cm-3]
remote-doping-sheet-density              = 1e12 ! [cm-2]

```

Final remark: In principle, the results of these GaN 2DEGs are not reliable as piezo- and pyroelectricity have to be included. Our results disagree quantitatively with the results of [WalukiewiczGaN]. However, it is not clear, which material parameters he used for the conduction band offset and the alloy scattering.

### Further hints

If two remote doping regions should be taken into account, one can input an array of values.

```

spacer-width                = 20.0  10.0  ! [nm]  spacer width of 1st_
↪ and 2nd doping region
remote-doping-sheet-density  = 1e12  1e11  ! [cm-2] remote doping density_
↪ of 1st and 2nd doping region

```

- (++) (Ma) Schrödinger-Poisson
- (++) (Ma) Si/SiGe MODQW (Modulation Doped Quantum Well)
- (++) (Ma) HEMT structure (High Electron Mobility Transistor)
- inverted High Electron Mobility Transistor (HEMT)
- (++) (Ma) Two-dimensional electron gas in an AlGaIn/GaN field effect transistor
- (++) (Ma) Single-electron transistor (3D)

## 3.14 Optoelectronics

- (++) (Ma) Cascade solar cell (AlGaAs/InGaAs Tandem solar cell)
- (++) (Ma) Quantum Cascade Laser (simple structure)
- (++) (Ma) Quantum Cascade Laser
- (++) (Ma) Intraband transitions



- (++) (Ma) Interband transitions
- (++) (Ma) Intersubband transitions in InGaAs/AlInAs multiple quantum well systems
- (++) (Ma) Scattering times for electrons in unbiased and biased single and multiple quantum wells
- Optical absorption of an InGaAs quantum well

### 3.15 Electronics

- (++) (Ma) Capacitance-Voltage (C-V) curve of a “metal”-insulator-semiconductor (MIS) structure
- p-Si / SiO<sub>2</sub> / poly-Si structure (MOSFET with inversion channel due to applied gate voltage)
- (++) (Ma) Resistance in bulk n-type doped silicon
- (++) (Ma) n-i-n Si resistor (classical and quantum mechanical current calculation)
- I-V characteristics of an n-doped Si structure
- (++) (Ma) I-V characteristics of an n-i-n-doped Si structure
- (++) (Ma) Ultrathin-body Double Gate FET - Double Gate MOSFET (Metal Oxide Semiconductor Field Effect Transistor) (2D)
- (++) (Ma) Double Gate MOSFET (5 nm) (quantum mechanical calculation) (2D)
- (Ma) Electron wave functions of a 2D slice of a Triple Gate MOSFET (2D)

### 3.16 NEGF

- Ballistic current in a resonant tunneling diode (RTD)

### 3.17 Electrolyte

for e.g. Ion-Sensitive Field Effect Transistors (ISFETs)

- Extended Poisson-Boltzmann equation: Potentials of mean force (PMF)
- Detection of proteins with a SOI (silicon-on-insulator) electrolyte sensor
- Si-SiO<sub>2</sub> electrolyte sensor
- Electrolyte Gate AlGaIn/GaN Field Effect Transistor as pH Sensor
- Diamond biosensors (available on request)

### 3.18 Graphene

- Carrier statistics of graphene sheets

### 3.19 Hello World

- Simple SiGe structure (This is a very ancient tutorial and was written 20 years ago. Meanwhile the software has evolved.)
  - Step 1: Simple SiGe structure (input\_file1.in)
  - Step 2: Include current (input\_file2.in)
  - Step 3: Include quantum models (1-band Schrödinger equation) (input\_file3.in)
  - Step 4: Read in data and solve k.p (input\_file4.in)

Four example input files are provided in this tutorial which represent typical simulation settings that are connected to each other. These files contain extensive comments to guide you through your first experience with the input system.

## 3.20 Archived *nextnano++* tutorials related to *nextnano*<sup>3</sup>

### 3.20.1 Landauer conductance and conductance quantization: from quantum wires to quantum point contacts

- *Header*
- *Introduction*
- *Simulations of the current in 1D wires*
- *Transmission and conductance of QPC, conductance quantization*

#### Header

##### Files for the tutorial:

- *1D\_GaAs\_conductance\_nnp.in* - simulations of 1D quantum wire in *nextnano++*
- *1D\_GaAs\_conductance\_nn3.in* - simulations of 1D quantum wire in *nextnano*<sup>3</sup>
- *2D\_transmission\_QPC\_nnp.in* - simulations of QPC in 2DEG
- *2D\_transmission\_QPC\_potential\_of\_2DEG\_1.fld* - numerically obtained energy profile of QPC
- *2D\_transmission\_QPC\_potential\_of\_2DEG\_2.fld* - numerically obtained energy profile of differently shaped QPC

##### Main adjustable parameters for 1D simulations (quantum wire):

- upper boundary for transmission energy - %E\_max
- the barrier widths - %Delta\_x = %Barrier\_max - %Barrier\_min
- the barrier heights - %Barrier\_Height
- the temperature - %Temperature
- Fermi levels of left ( $x_{\min\_contact} < x < x_{\min}$ ) and right ( $x_{\max} < x < x_{\max\_contact}$ ) regions (leads) - %Fermi\_left and %Fermi\_right
- the effective mass of the electron - %effective\_mass

##### Relevant output files of 1D simulations (quantum wire):

- *Results\BandEdges.dat* (energy profile)
- *Results\Transmission\_cb\_sg1\_deg1.dat* (transmission)
- *Results\LocalDOS\_sg1\_deg1\_Lead1.fld* and *Results\LocalDOS\_sg1\_deg1\_Lead1.fld* (LDoS)
- *Results\IV\_characteristics.dat* (currents)

##### Main adjustable parameters for 2D simulations (QPC):

- dimensions of the device - \$x\_length and \$y\_length
- grid spacing in x and y direction, \$grid\_spacing
- number of eigenvalues in the device and the leads - \$num\_eigenstates\_device and \$num\_eigenvalues\_leads
- the temperature - \$Temperature

- energy range and resolution that the transmission will be computed -  $E_{\min}$ ,  $E_{\max}$  and  $\Delta E$
- path of the file to be imported - `$pathPotentialFile`

#### Relevant output files of 2D simulations (QPC):

- `bias_00000\bandedges.fld` (energy profile)
- `Structure\contact.fld` (contacts)
- `bias_00000\CBR\transmission_sums_device_Gamma.dat` (transmission)

## Introduction

**Conductance**,  $G$ , is the quantity which describes the relation between an electric current,  $J$ , and an applied voltage,  $V_{\text{dc}}$ , which causes this current. In this tutorial, we briefly review the analytical theory, which allows one to calculate the conductance, and compare it with the numerical approach implemented into the *nextnano* software. We discuss only the dc case with the main focus on the linear response regime where  $J = GV_{\text{dc}}$ .

Unlike conductivity, which characterizes properties of a material, conductance describes a given sample. Therefore, geometry and size of the sample matter. We start below from an example of a *quantum wire* where the electric current is carried either by one (one-dimensional, 1D) or several (quasi-1D) propagating modes. Conductance of the quantum wire is described by the seminal *Landauer theory*. A simple introduction to the Landauer theory can be found in the book by S. Datta [Datta], section 2 “Conductance from Transmission”.

The setup of the Landauer theory is shown in the upper panel of Figure 3.20.1.1. The device is connected via left and right ideal wires (ballistic conductors) to two leads with different chemical potentials. The current flows from the material with a larger chemical potential to that with a smaller one.

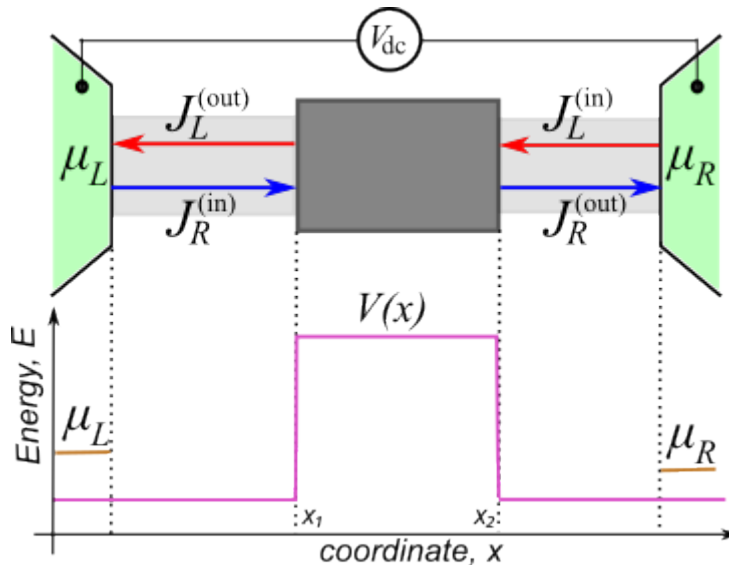


Figure 3.20.1.1: Upper panel: Landauer setup. Left and right leads (green regions) are connected to a semiconductor device (dark gray square) via connecting wires (light gray regions). Lower panel: chemical potentials of the leads (orange lines) and the energy of the potential barrier (magenta lines).

In the standard approach, the leads are two- or three-dimensional large conductors and contacts between the leads and the wires are reflectionless. This ensures that electrons supporting the current  $J_R^{(\text{in})}$  are in equilibrium with the left lead and have the chemical potential  $\mu_L$ . Similarly, the electrons supporting the current  $J_L^{(\text{in})}$  are in equilibrium with the right lead and have the chemical potential  $\mu_R$ .

## Simulations of the current in 1D wires

Let us assume that all elements of the electric circuit are purely 1D, there is no temperature gradient, and the chemical potentials of the leads are shifted by the applied external voltage,  $\mu_R - \mu_L = eV_{dc}$ .

### ⚠ Attention

The value of chemical potentials is not calculated in this tutorial but is set a kind of “artificially”. Of course, this value must be in agreement with physics of a given material. For example, when the temperature (at  $k_B = 1$ ) is smaller than the energy gap separating the conduction and valence bands, the chemical potential of an intrinsic unbiased semiconductor is close to the center of that gap, see e.g. section 3 *The Fermi-Dirac Distribution* in [Grah].

Since the connecting wires are ballistic and the contacts are reflectionless, the backscattering of the electrons can occur only inside the semiconductor device. We model this by including a potential scatterer (a square barrier) into the simulations. Hence, the scattering inside the device is elastic, the energy of the scattered electron is unchanged, and the electrons supporting the currents  $J_{R,L}^{(out)}$  are a mixture of the electrons with the chemical potentials  $\mu_{R,L}$ . The energy landscape of the device containing a square potential,  $V(x < x_1) = V(x > x_2) = 0$ ,  $V(x_1 < x < x_2) = V_0$ , is shown in the lower panel of Figure 3.20.1.1. The electrons whose energy is small,  $E < V_0$ , can tunnel through the potential barrier. The electrons with large energies,  $E > V_0$ , can be reflected due to quantum effects. For the simple case of the rectangular barrier, the transmission in both cases is known:

$$\mathcal{T}(0 < E < V_0) = 1 / \left( 1 + [\kappa \sinh(\tilde{k}a)]^2 \right), \quad (3.20.1.1)$$

$$\mathcal{T}(E > V_0) = 1 / \left( 1 + [\kappa \sin(\tilde{k}a)]^2 \right); \quad (3.20.1.2)$$

Here  $\tilde{k} = \sqrt{2m|V_0 - E|}/\hbar$ ,  $\kappa = \sqrt{V_0^2/4E|V_0 - E|}$ ,  $m$  is the (effective) mass of the electron and  $\hbar$  is the Planck constant. Transmission of the device is needed to calculate the current: The total current is the difference of currents flowing in opposite directions:  $J = J_R^{(in)} - J_L^{(out)} = J_R^{(out)} - J_L^{(in)}$ . Here, upper indices indicate whether a given current flows into or from the device. The Landauer formula allows one to express  $J$  via  $\mathcal{T}$ . In the purely 1D setup, the current reads:

$$J = 2e \int \frac{dk}{2\pi} v(k) \mathcal{T}(k) (f_L(k) - f_R(k)) = \frac{2e}{h} \int dE \mathcal{T}(E) (f_L(E) - f_R(E)); \quad (3.20.1.3)$$

where  $e$  and  $k$  are the electron charge and its wave-vector, respectively. The electrons in the left/right leads are described by the Fermi-Dirac distribution functions,  $f_{L,R}$ . The second equality in (3.20.1.3) has been obtained after changing the integration variable from the electron wave-vector to its energy.

If  $V_0 = 0$ , i.e.  $\mathcal{T} = 1$ , a simple calculation yields  $J = G_0 V_{dc}$  where  $G_0 = 2e^2/h$  is the quantum of the conductance. The nextnano software reproduces this result with a very high accuracy, see Figure 3.20.1.2. The numerical simulations presented in this tutorial were done by using *Contact Block Reduction method* [CBR], see also *tutorial on the CBR method*.

The users of the nextnano software should pay attention that regions, which are called “leads” in the CBR-based sample input files, are actually interfaces between the devices and the connecting wires. These interfaces have minimal width of the space discretization. In the toy model which we discuss the chemical potential of each interface is equal to that of the corresponding lead. Such a simplification of the Landauer setup is natural in the CBR method. One may refer to the interfaces between the device and the connecting wires as “CBR-leads”. An example of the CBR-leads is shown below for the case of the two-dimensional (2D) device.

Figure 3.20.1.3 and Figure 3.20.1.4 shows the transmission and the IV characteristics of the device which contains the square scattering potential of width 30 nm with  $V_0 = 100\text{meV}$ .

Since transmission of the device is exponentially small at energies below 0.1 eV, the current become nonzero only at  $\mu_L > 0.1\text{eV}$  and, after some transient, the IV characteristics becomes again linear with the slope being close to  $G_0$  with accuracy of several percents.

### Exercise

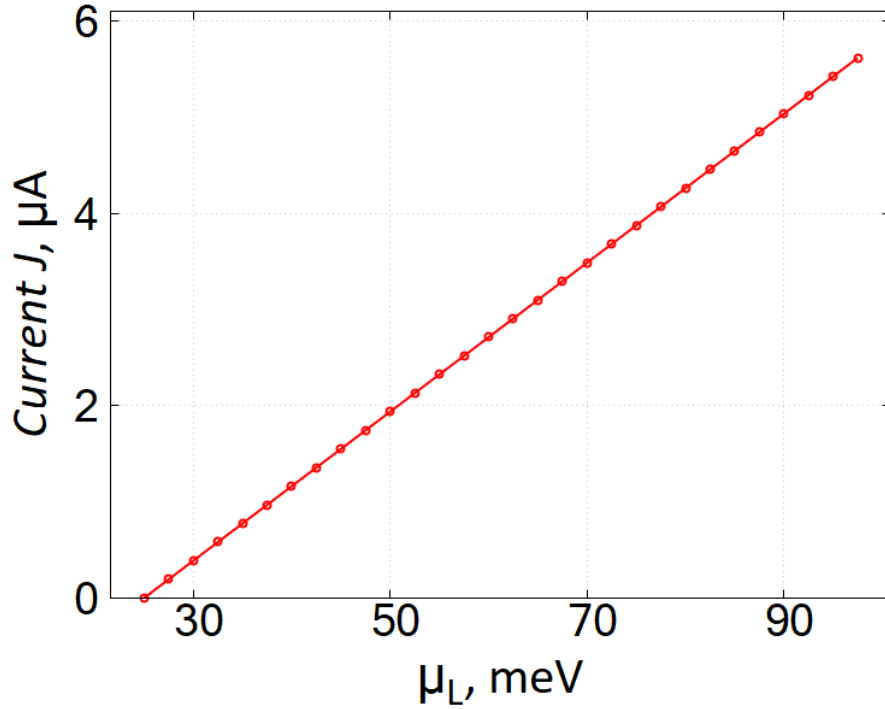


Figure 3.20.1.2: Numerically calculated IV-characteristics of a ballistic 1D conductor,  $V_0 = 0$ . We have chosen GaAs as the material of the conductor with the total length 32 nm at  $\mu_R = 25\text{meV}$ ; the temperature was set to  $T = 50\text{mK}$ . Note that these parameters has no influence on the universal slope of the IV straight line which is equal to  $G_0$ . For chosen parameters of the numerical solver and the numerical integration procedure (cf. the sample input file), the difference between the numerically calculated slope and  $G_0$  is  $\simeq 4\%$ .

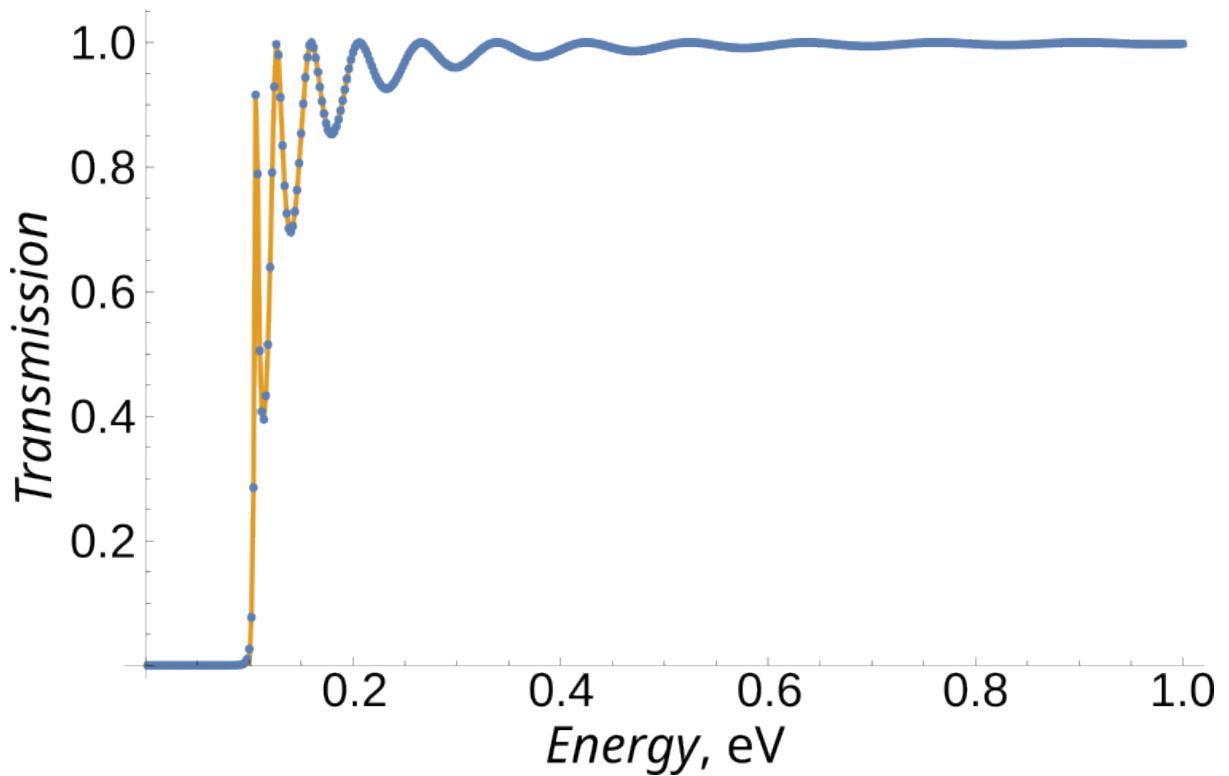


Figure 3.20.1.3: Transmission of a 1D conductor with  $V_0 = 100\text{meV}$  and width 30nm. Orange line and blue dots shows the exact analytical answer, Eqs. (3.20.1.1) and (3.20.1.2), and CBR calculations, respectively.

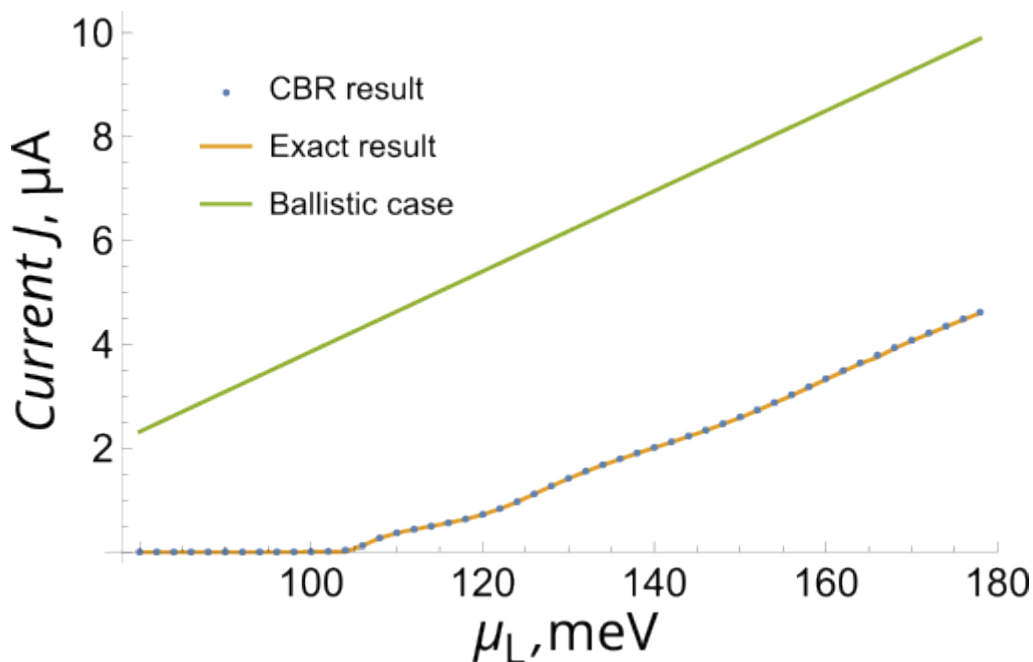


Figure 3.20.1.4: IV-characteristics of a 1D conductor with  $V_0 = 100\text{meV}$  and width 30 nm at  $\mu_R = 50\text{meV}$ . Other parameters are the same as in Figure 3.20.1.2. Orange line and blue dots shows the exact analytical answer [obtained by using Eqs. (3.20.1.1) and (3.20.1.2)], and CBR calculations. Green line exemplifies the ballistic law  $J = G_0 V_{\text{dc}}$ .

- **Calculate numerically transmission and current through a biased potential which linearly** increases from the value  $V(x_1) = V_1$  to  $V(x_2) = V_2$  with  $V_1 < V_2$ . Compare the result of simulations with that for the unbiased barrier.
- **Repeat the simulations for the inverted biased barrier:**  $V(x_1) = V_2$  to  $V(x_2) = V_1$  keeping all other parameters the same as in the previous task. Do transmission and current change under spatial inversion of the barrier? Explain your answer.

### Transmission and conductance of QPC, conductance quantization

The CBR method implemented in [nextnano software](#) allows one also to calculate conductance of more complicated semiconductor devices, for example, of a [quantum point contact](#) (QPC). QPC in a 2D electron gas (2DEG) can be created in a semiconductor heterostructure by a voltage applied to a top gate. In this case, the potential energy in the plane of the 2DEG can be obtained from the numerical solution of the Poisson equation. An example of such a profile of the potential energy is shown in [Figure 3.20.1.5](#).

The energy profile can be imported into the nextnano procedure which calculates transmission, e.g., from left to right boarder of the sample. The left CBR-lead used in this tutorial is illustrated in [Figure 3.20.1.6](#). The right CBR-lead is attached at  $x = 400\text{nm}$ .

Numerically calculated energy dependence of the QPC transmission is shown in [Figure 3.20.1.7](#). Temperature corrections to the transmission (due to the temperature-dependent gap) and to the conductance (due to the thermal broadening of the distribution functions) are negligibly small in the sub-Kelvin range ( $\ll 1\text{K}$ ) and we neglect them in this tutorial.

The lowest modes with the energy  $< -35\text{meV}$  are localized near the CBR-leads and do not contribute to transmission. A small plateau of  $\mathcal{T} \simeq 1$  at  $-34.5\text{meV} < E < -34\text{meV}$  corresponds to the energies where the first delocalized mode of the device yields its maximum contribution to the transmission. The second (slightly smeared) plateau,  $\mathcal{T} \simeq 2$ , signals that the second delocalized mode yields its maximum contribution to the transmission, etc.

The example of the gate-induced QPC is 2D and requires 2D simulations. However, the second equation in (3.20.1.3) still can be used. It suggests that, if temperature and  $V_{\text{dc}}$  are extremely small, then linear conductance is proportional to transmission:  $G_{\text{QPC}} = G_0 \mathcal{T}(\mu)$ . Negative values of the chemical potential,  $\mu$ , of the gated

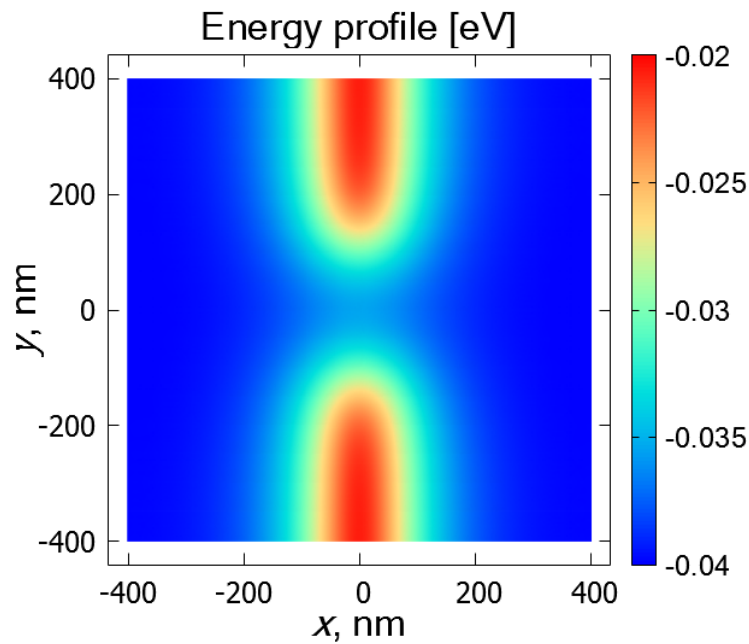


Figure 3.20.1.5: An example of the numerically obtained energy profile for a QPC in the plane of the 2D electron gas. The simulations were done for the 2D electron gas in GaAs at temperature 100mK.

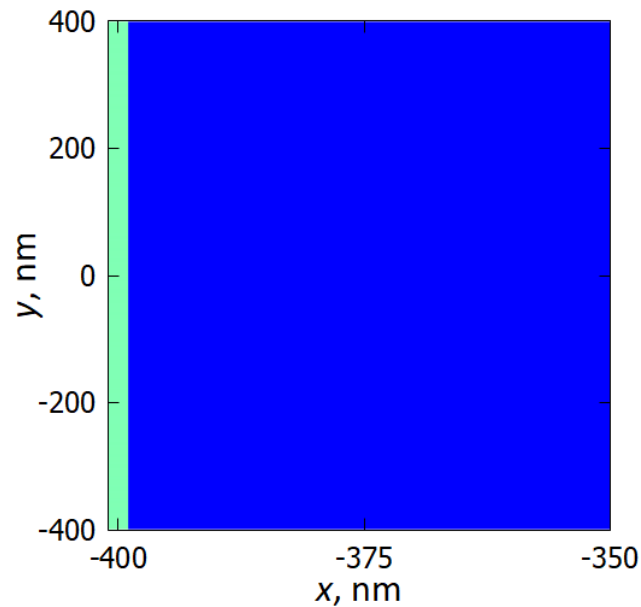


Figure 3.20.1.6: Illustration of how the left CBR-lead (light green region) is attached to the device (blue region). The width of the lead along x-axis is equal to the step of the space discretization. The width of the lead along y-axis has been chosen to be equal to the width of the device.

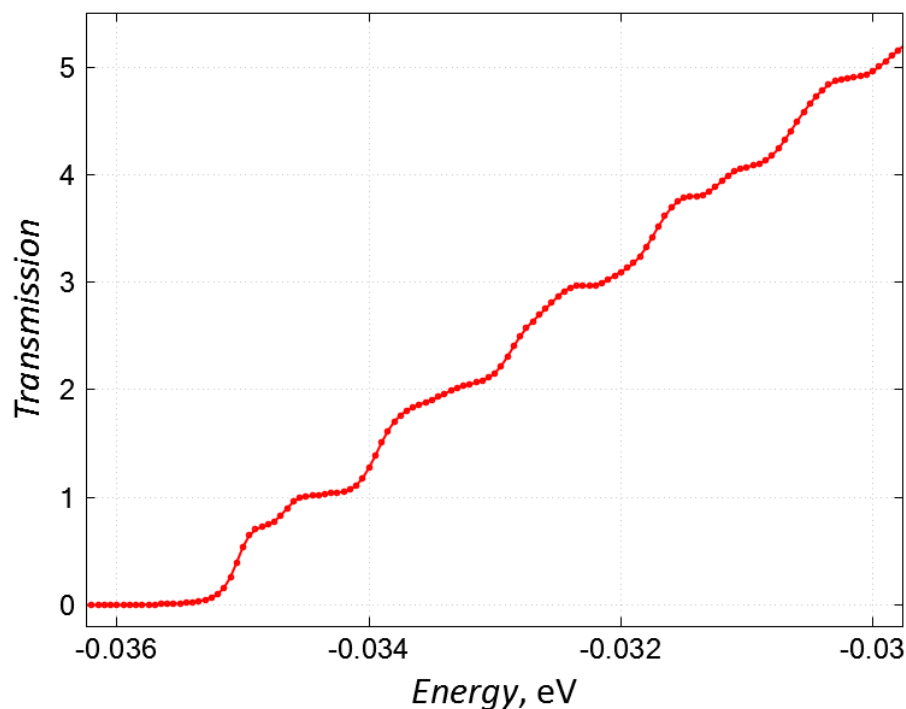


Figure 3.20.1.7: Numerically calculated energy dependence of the transmission via the QPC which is presented in Figure 3.20.1.5. The bottom of the conduction band,  $E_0$ , of the gated 2DEG is located at  $\simeq -40\text{meV}$ . Hence,  $E_0$  is the origin of the energy for this example.

semiconductor structure are related to the choice of the origin, which is explained above. To conclude, we note that plateaux in the energy dependent transmission correspond to those in the conductance which are called in the literature “conductance quantization”.

### Exercises

- The above example was based on the QPC geometry taken from the file `2D_transmission_QPC_2D_potential-v1_of_2DEG.fld`. File `2D_transmission_QPC_2D_potential-v2_of_2DEG.fld` contains another QPC geometry which results from a different shape of the top gate electrode. Use this file with the alternated QPC geometry, process it with the help of the nextnano input file, and calculate the QPC transmission.

#### Attention

The minimal energy, above which transmission is finite (not zero), depends on the QPC geometry and on the applied gate voltage. Hence, one has to find an appropriate energy range where the plateaux of the quantized conductance are well visible.

- Compare the energy profile and the energy dependent transmission for the both shapes of the QPC.
- Note that the second QPC shape does not possess “left  $\leftrightarrow$  right” inversion symmetry (inversion with respect to the line  $x = 0$ ). Compare transmissions from the left to right CBR leads with that from the right to left leads. Are they equal? Explain your observation.



### 3.20.2 InAs / GaSb broken gap quantum well (BGQW) (type-II band alignment)

Author: Stefan Birner

Input files required:

- `1DInAs_GaSb_BGQW_k_zero_nnp.in`
- `1DInAs_GaSb_BGQW_k_parallel_nnp.in`
- `1DInAs_GaSb_BGQW_k_parallel_nnp_01.in`
- `1DInAs_GaSb_BGQW_k_parallel_nnp_11.in`

This tutorial aims to reproduce Figs. 1, 2(a), 2(b) and 3 of *Hybridization of electron, light-hole, and heavy-hole states in InAs/GaSb quantum wells*

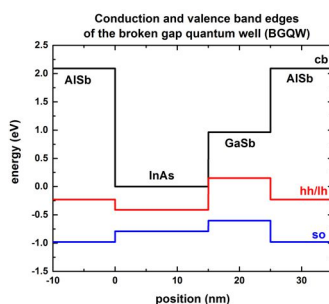
Material parameters used are taken from *Optical transitions in broken gap heterostructures*.

The heterostructure is a **broken gap quantum well (BGQW)** with 15 nm **InAs** and 10 nm **GaSb**, sandwiched between two 10 nm **AlSb** layers. Note that this heterostructure is asymmetric.

To be consistent with the above cited papers, strain is not included into the calculations although this would be possible. The structure has a **type-II** band alignment, i.e. the electrons are confined in the InAs layer, whereas the holes are confined in the GaSb layer. Depending on the width of the InAs and/or GaSb layers, things can be even more complicated because the hole states can hybridize with the electron states, making it difficult to distinguish between electron-like and hole-like states. Another difficulty arises because the lowest electron states might be located below the highest hole states. This requires a new algorithm to occupy the states according to a suitable Fermi level.

The following figure shows the electron and hole band edges of the BGQW structure.

- `band_structure/cb1D_001.dat` (Gamma conduction band edge) in units of [eV]
- `band_structure/vb1D_001.dat` (heavy hole valence band edge) in units of [eV]
- `band_structure/vb1D_002.dat` (light hole valence band edge) in units of [eV]
- `band_structure/vb1D_003.dat` (split-off hole valence band edge) in units of [eV]



The origin of the energy scale is set to the InAs conduction band edge energy. The heavy hole and light hole band edges are degenerate because we neglect the effects of strain to be consistent with the above cited papers.

#### Results

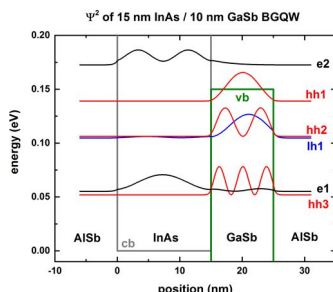
The input file used here is `1DInAs_GaSb_BGQW_k_zero_nnp.in`. The following figure shows the conduction band edge and the heavy/light hole valence band edges in this BGQW structure together with the electron (**e1**, **e2**), heavy hole (hh1, hh2, hh3) and light hole (lh1) energies and wave functions ( $\psi^2$ ), calculated within 8-band  $\mathbf{k} \cdot \mathbf{p}$  theory at the zone center, i.e. at  $\mathbf{k}_{||} = 0$ .

One can clearly see that the electron state (**e1**, **e2**) are confined in the InAs layer (left part of the figure), whereas the heavy (hh1, hh2, hh3) and light hole (lh1) states are confined in the GaSb layer (right part of the figure). One can see a slight hybridization of the **e1** and lh1 states, i.e. these states are mixed states whereas the heavy hole states (hh1, h2, hh3) are not mixed and thus confined in the GaSb layer.

We use the data files

- `Schroedinger_kp/kp_8x8psi_squared_qc001_el_kpar0001_1D_dir.dat`, which contains  $\psi^2$
- `Schroedinger_kp/kp_8x8psi_squared_qc001_el_kpar0001_1D_dir_shift.dat`, which contains  $\psi^2 + E_i$

The latter file contains the square of the wave functions (for `par0001`, i.e.  $k_{||} = 0$ , i.e.  $k_x = k_y = 0$ ), shifted by their energies, so that one can nicely plot the conduction and valence band edges together with the square of the wave functions.



The energies of the eigenstates are in units of [eV] and are contained in the file `Schroedinger_kp/kp_8x8eigenvalues_qc001_el_kpar0001_1D_dir.dat`

The input file `1DInAs_GaSb_BGQW_k_parallel.in` was used for the following results. The following figure shows the  $E(k_{||})$  dispersion of the electron and hole states along the [10] direction and along the [11] direction in  $(k_x, k_y)$  space. The [01] direction has the same dispersion due to symmetry arguments.

In this input file, the energy levels and wave functions for 24  $k_{||}$  points along a line from  $(k_x, k_y) = (0,0)$  to  $(k_x, k_y) = (0, k_y)$  have been calculated.

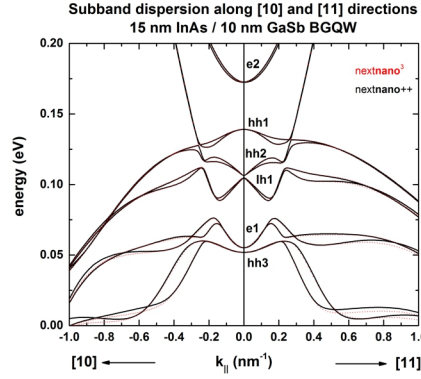
`Schroedinger_kp/kpar1D_disp_01_00el_8x8kp_ev_min001_ev_max020.dat` contains the  $k_{||}$  dispersion from [00] to [01] because in the input file, it is specified that

```
dispersion{
  path{
    name = "kpar_01_00_10"
    point{ k = [0.0, 0.0, 1.0] }
    point{ k = [0.0, 0.0, 0.0] }
    point{ k = [0.0, 1.0, 0.0] }
    spacing = 1 / $number_k_parallel
  }
  path{
    name = "kpar_10_00_11"
    point{ k = [0.0, 1.0, 0.0] }
    point{ k = [0.0, 0.0, 0.0] }
    point{ k = [0.0, 1.0, 1.0] }
    spacing = 1 / $number_k_parallel
  }
  output_dispersions{}
  output_masses{}
}
```

The first column contains the  $k_{||}$  value, the other columns contain the eigenvalues for each  $k_{||}$  value:  $E_n(k_{||}) = E_n(k_x, k_y) = E_n(0, k_y)$ . Here,  $n = 1, \dots, 20$ . (...ev\_min 001\*\*\_ev\_max \*\*020...) Note that for this particular example, the eigenvalues have to be sorted manually if you want to connect the energy values, i.e. to include lines ("lines are a guide to the eye").

The **black lines** are the results of `nextnano++`, the red dots are the results of `nextnano3`.

At an in-plane wave vector of  $0.014 \text{ 1/\AA}$ , strong intermixing between the e1 and the lh1 states occurs. In contrast to the wave functions at  $k_{||} = 0$ , where the **e1** and **lh1** wave functions are nearly purely electron- or hole like, the



wave functions at  $k_{||} = (0, 0.014) = (0.014, 0)$  are a mixture of electron and light hole wave functions. Compare with Fig. 4 of the A. Zakharova et al.

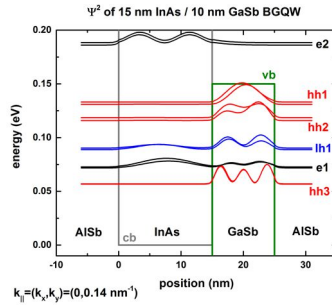
In asymmetric quantum wells, the double spin degeneracy is lifted at finite values of  $k_{||}$  because of spin-orbit interaction. This is the reason why we have two different dispersions  $E(k_{||})$  for “spin up” and “spin down” states. This also means that the wave functions at finite  $k_{||}$  are different for “spin up” and “spin down” states.

The file `Schroedinger_kp/kp_8x8k_parallel_qc001_el1D_dir.dat` tells us which number of  $k_{||}$  vector corresponds to  $(k_x, k_y)$ .

| k_par_number | k_x [1/nm]    | k_y [1/nm]    |                                               |
|--------------|---------------|---------------|-----------------------------------------------|
| 1            | 0.000000E+000 | 0.000000E+000 | ==> $k_{  } = (k_x, k_y) = (0, 0)$ [1/nm]     |
| ...          |               |               |                                               |
| 29           | 0.000000E+000 | 1.400000E+000 | ==> $k_{  } = (k_x, k_y) = (0, 0.14)$ [1/nm]  |
| 1326         | 1.000000E+000 | 1.000000E+000 | ==> $k_{  } = (k_x, k_y) = (1.0, 1.0)$ [1/nm] |

In the following figure, we plot the square of the wave functions for  $k_{||} = (0, 0.14) \text{ nm}^{-1}$ . The corresponding label of our  $k_{||}$  numbering is **29**. Note that this labeling depends on the  $k_{||}$  space resolution, i.e. the number of  $k_{||}$  points that have been specified in the input file: `num-kp-parallel = 10000`

The wave functions  $(\psi^2 + E_i)$  are contained in the file `Schroedinger_kp/kp_8x8psi_squared_qc001_hl_kpar00029_1D_dir_shift.dat`



The electron states (**e1**) couple strongly with the light hole states (**lh1**). This is expected from the energy dispersion plot because at  $0.14 \text{ nm}^{-1}$  a strong anticrossing is present for these states. One can also clearly see that for spin up and spin down states, different energy levels and different probability densities exist. This is in contrast to the states at  $k_{||} = 0$  which are two-fold spin degenerate as shown in the figure further above. Our results are similar to Fig. 4 of Zakharova’s paper.

Last update: nnnn/nn/nn

### 3.20.3 Cascade solar cell (Tandem solar cell)

#### Input Files:

- `1DCascadeSolarCell_nnp.in` / `*nn3.in`

In this tutorial, we solve the Poisson equation in an AlGaAs/InGaAs monolithic cascade solar cell (tandem solar cell).

The layout is based on [US patent 4179702 \(1979\): Cascade solar cells](#) by Michael F. Lamorte.

See also the following publication for more details

*Computer Modeling of a Two-Junction, Monolithic Cascade Solar Cell*

M.F. Lamorte, D.H. Abbott

IEEE Transactions on Electron Devices 27 (1), 231 (1980)

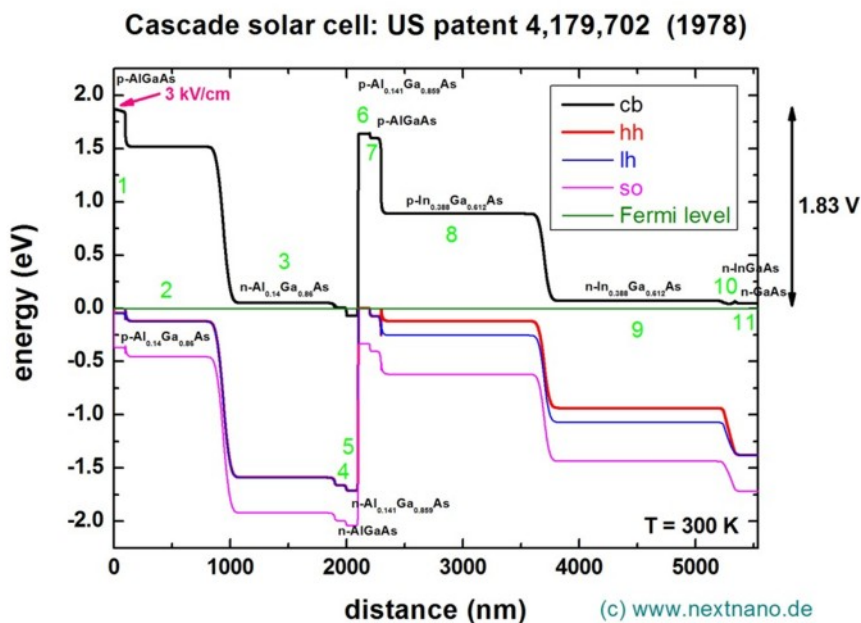
Input files used in this tutorial are followings:

#### Outputs

##### Band profile

The following figure shows the conduction band edge and the valence band edges (heavy hole, light hole and split-off hole) of this solar cell at zero bias. The built-in potential has been calculated to be 1.83 V.

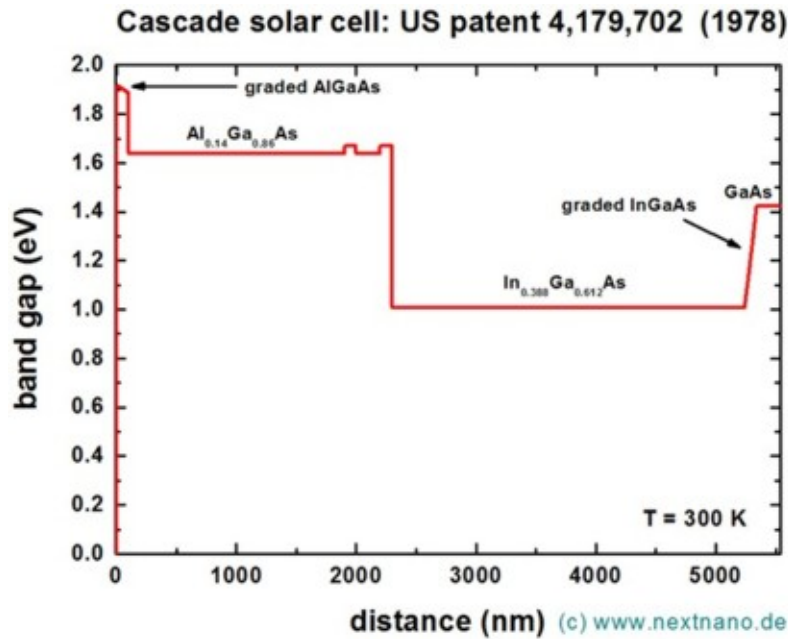
On the left side (region 1), a graded p-type AlGaAs layer has been used to generate an electric field of 3 kV / cm (= 30 meV / 100 nm). We assumed that all materials are strained with respect to the GaAs substrate, thus the degeneracy of heavy and light hole valence band edges is lifted, especially inside the InGaAs regions.



##### Band gap

The band gap as a function of distance is shown in the following figure. This data can be found in these files.

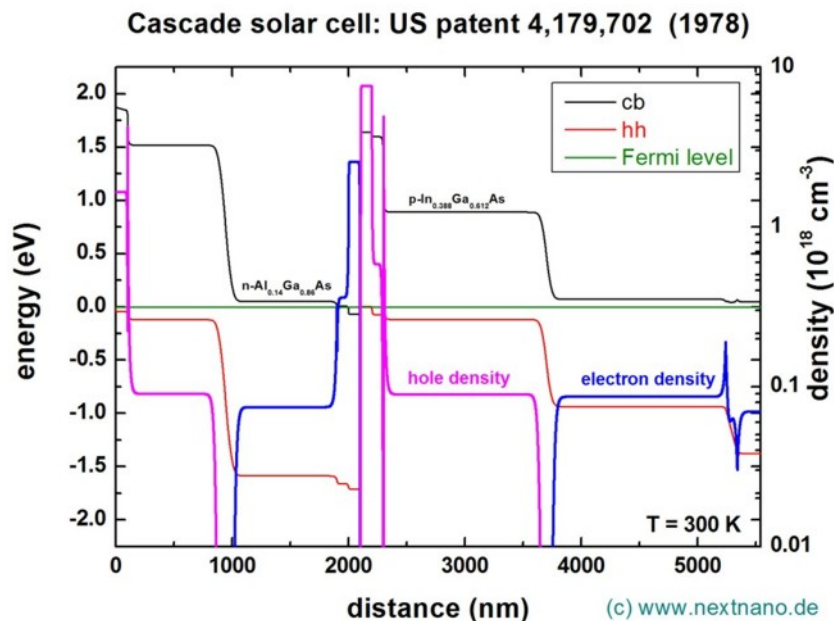
- `bias_00000/bandgap.dat` (nextnano++)
- `band_structure/BandGap1D.dat` (nextnano<sup>3</sup>)



### Electron and hole densities

Here, the electron and hole densities are plotted. This data can be found in these files.

- *bias\_00000/density\_electron.dat*, *bias\_00000/density\_hole.dat* (nextnano++)
- *densities/density\_el.dat*, *densities/density\_hl.dat* (nextnano<sup>3</sup>)

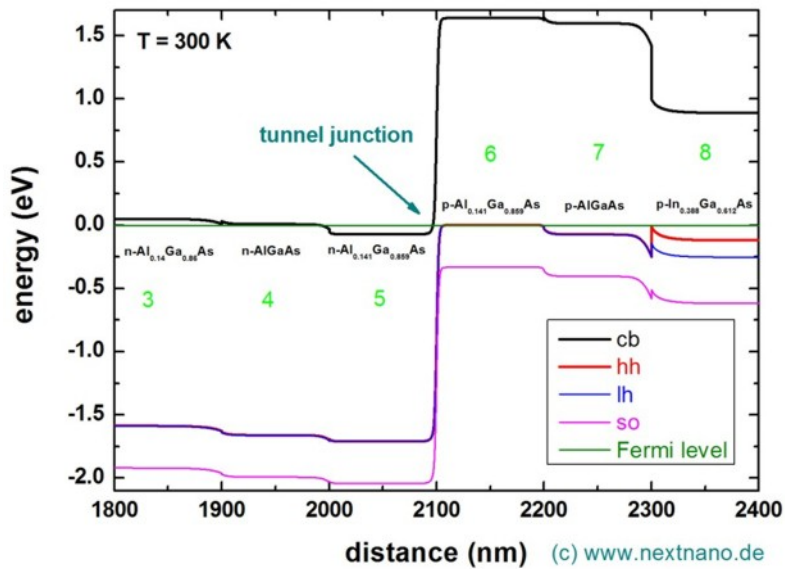


### Tunnel junction

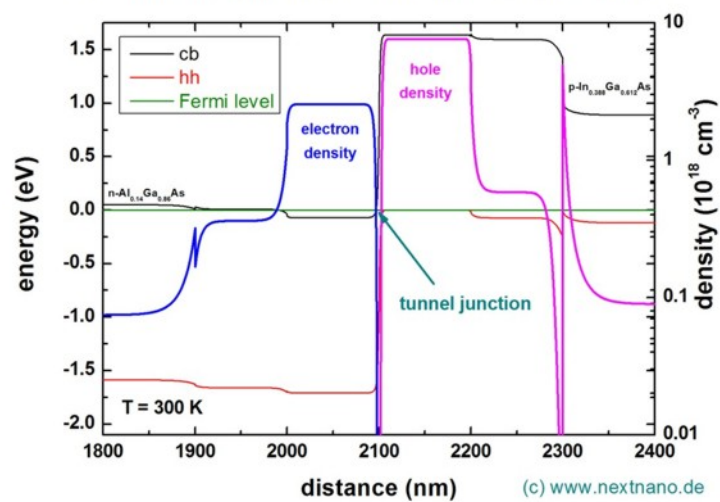
The area around the tunnel junction which is in the middle of the device at  $\sim 2100 \text{ nm}$  is shown in this plot:

The electron and hole densities in the vicinity of the tunnel junction are shown in this graph. Note that the density has been calculated classically (without solving the Schrödinger equation, i.e. without quantum mechanics).

Tunnel junction of the cascade solar cell



Electron and hole densities of the cascade solar cell



### What we can do on a solar cell using nextnano

We have the demonstration of the simulation for GaAs solar cell using nextnano here: [GaAs Solar Cell](#).

As we can see in this demonstration, we can calculate the following characteristics by solving the Poisson equation and current equation self-consistently.

- **Current-Voltage characteristics**
  - The dark current can also be calculated.
- **Solar efficiency**
  - We can also see the effect of optical concentration quantitatively.

The data we need to prepare independently for this calculation is:

1. spectral irradiance (solar spectrum)
2. reflectivity at the front surface
3. absorption spectrum

Both *nextnano++* and *nextnano<sup>3</sup>* can calculate the generation rate  $G(x)$  now. We can also import the data of  $G(x)$  directly instead of 2 and 3 above.

The links for all the used data is also specified in this tutorial: [GaAs Solar Cell](#).

---

*Last update: 2024/06/10*

### 3.20.4 — FREE — Double Quantum Well

#### Input files:

- *DoubleQuantumWell\_6\_nm\_nnpp.in*
- *DoubleQuantumWell\_6\_nm\_nn3.in*

This tutorial calculates the energy eigenstates of a double quantum well. This aims to reproduce two figures (Figs. 3.16, 3.17, p. 92) of Paul Harrison's excellent book "Quantum Wells, Wires and Dots" (Section 3.9 "The Double Quantum Well"), thus the following description is based on the explanations made therein. *We are grateful that the book comes along with a CD so that we were able to look up the relevant material parameters and to check the results for consistency.*

Input files for both the *nextnano++* and *nextnano<sup>3</sup>* tools are available.

To generate the input files for various thicknesses and some of the plots the following *nextnanomat* features are used:

- Template tab
- Postprocessing feature

It is recommended to read the documentation about these features of the graphical user interface *nextnanomat* **before** starting this tutorial.

#### Structure: **AlGaAs / 6 nm GaAs / AlGaAs / 6 nm AlGaAs / AlGaAs**

Our symmetric double quantum well consists of two 6 nm GaAs quantum wells, separated by a  $\text{Al}_{0.2}\text{Ga}_{0.8}\text{As}$  barrier and surrounded by 20 nm  $\text{Al}_{0.2}\text{Ga}_{0.8}\text{As}$  barriers on each side. We thus have the following layer sequence: **20 nm  $\text{Al}_{0.2}\text{Ga}_{0.8}\text{As}$  / 6 nm GaAs /  $\text{Al}_{0.2}\text{Ga}_{0.8}\text{As}$  / 6 nm GaAs / 20 nm  $\text{Al}_{0.2}\text{Ga}_{0.8}\text{As}$** . (The barriers are printed in bold.)

In this tutorial, we demonstrate the following two examples:

1. we set the thickness of the  $\text{Al}_{0.2}\text{Ga}_{0.8}\text{As}$  barrier that separates the two quantum wells **4 nm** and calculate **the lowest two eigenstates**.



2. we vary the thickness of the barrier layer **from 1 nm to 14 nm** fixing the width of the quantum well (6 nm). Then we calculate **the lowest two eigenstates** for each case and see the barrier-width dependency of their eigenenergies.

We also explain where the relevant output files are in.

## Material Parameters

The material parameters are given in `database_nn*.in` but we can also redefine them manually in input files. In this tutorial, we redefine parameters so that they are the same as the section 3.9 of Paul Harrison's book "Quantum Wells, Wires and Dots".

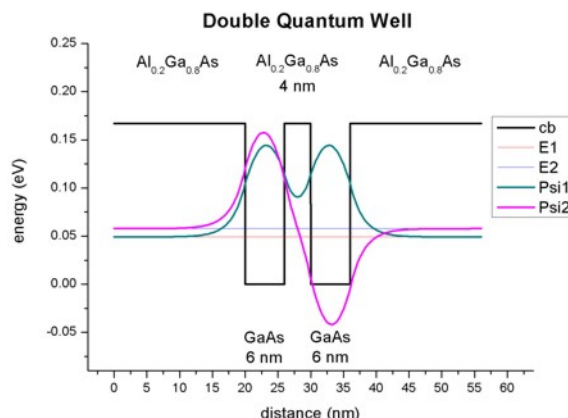
|                                |                                                          |             |
|--------------------------------|----------------------------------------------------------|-------------|
| conduction band offset         | $\text{Al}_{0.2}\text{Ga}_{0.52}\text{As} / \text{GaAs}$ | 0.167 eV    |
| conduction band effective mass | $\text{Al}_{0.2}\text{Ga}_{0.52}\text{As}$               | 0.084 $m_0$ |
| conduction band effective mass | GaAs                                                     | 0.067 $m_0$ |

## Results

### 1. barrier width = 4 nm

- The following figure shows the **conduction band edge** and **wave functions** that are confined inside the wells with barrier width = 4 nm.

(Note that the energies were shifted so that the conduction band edge of GaAs equals 0 eV.)



- The wave functions form a symmetric and an anti-symmetric pair. The symmetric one is lower in energy than the anti-symmetric one. The plot is in excellent agreement with Fig. 3.17 (page 92) of Paul Harrison's book "Quantum Wells, Wires and Dots".
- For comparison, the following figure shows for the same structure as above, the square of the wave function rather than  $\psi$  only.

## Output

- a. The conduction band edge of the Gamma conduction band can be found here:

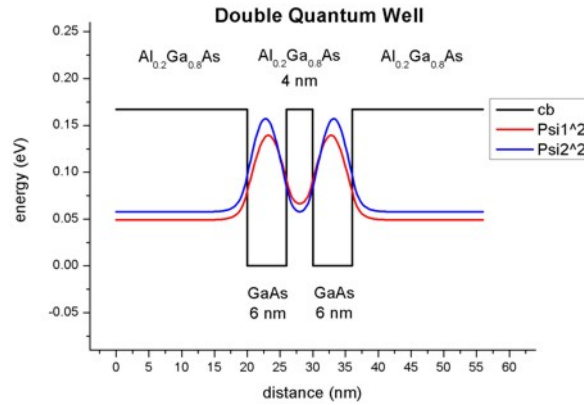
`bias_00000/bandedge_Gamma.dat (nextnano++)`  
`band_structure/cb_Gamma.dat (nextnano3)`

- b. This file contains the eigenenergies of the two lowest eigenstates. The units are [eV].

`bias_00000/Quantum/wf_energy_spectrum_quantum_region_Gamma_0000.dat (nextnano++)`  
`Schroedinger_1band/ev_cb1_sg1_deg1.dat (nextnano3)`

These are the comparison of eigenvalues:





|                                 | <i>nextnano++</i> | <i>nextnano</i> <sup>3</sup> | Harrison's book |
|---------------------------------|-------------------|------------------------------|-----------------|
| ground state energy [eV]        | 0.04920           | 0.04920                      | 0.04912         |
| first excited state energy [eV] | 0.05779           | 0.05779                      | 0.05770         |

c. This file contains the eigenenergies and the wave functions ( $\psi$ ):

*bias\_00000/Quantum/wf\_amplitudes\_shift\_quantum\_region\_Gamma\_0000.dat* (*nextnano++*)

*Schroedinger\_1band/cb1\_sg1\_deg1\_psi\_shift.dat* (*nextnano*<sup>3</sup>)

This file contains the eigenenergies and the squared wave functions ( $\psi^2$ ):

*bias\_00000/Quantum/wf\_probabilities\_shift\_quantum\_region\_Gamma\_0000.dat*  
(*nextnano++*)

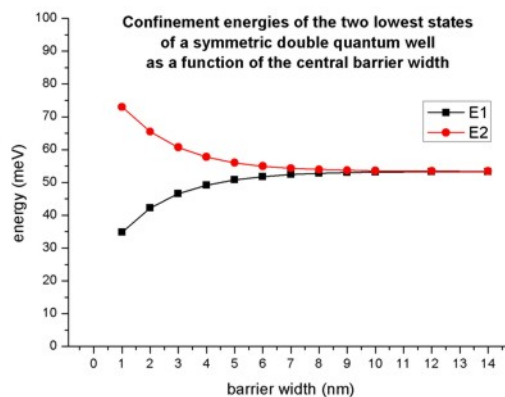
*Schroedinger\_1band/cb1\_sg1\_deg1\_psi\_squared\_shift.dat* (*nextnano*<sup>3</sup>)

The subscript *\_shift* indicates that  $\psi^2$  and  $\psi$  are shifted by the corresponding energy levels.

a. and c. can be used to plot the data as shown in the figures above.

## 2. barrier width = 1 ~ 14 nm

- Here, we varied the thickness of the  $\text{Al}_{0.2}\text{Ga}_{0.8}\text{As}$  barrier layer from 1 nm to 14 nm fixing the width of the quantum well (6 nm). We calculated the lowest two eigenstates and show their eigenvalues for each barrier width in the following figure (generated with the Postprocessing feature of *nextnanomat*).



- If the separation between the two quantum wells is large, the wells behave as two independent single quantum wells having the identical ground state energies. The interaction between the energy levels localized within each well increases once the distance between the two wells decreases below 10 nm. One state is forced to higher energies and the other to lower energies. (Here, the electron spins align in an “anti-parallel” arrangement in order to satisfy the Pauli exclusion principle.)

- This is analogous to the hydrogen molecule where the formation of a pair of bonding and anti-bonding orbitals occurs once the two hydrogen atoms A and B are brought together.

$$\psi_{\text{bonding}} = \frac{1}{\sqrt{2}}\psi_A + \psi_B \quad (\text{lower energy})$$

$$\psi_{\text{antibonding}} = \frac{1}{\sqrt{2}}\psi_A - \psi_B \quad (\text{higher energy})$$

- Again, the plot is in excellent agreement with Fig. 3.16 (page 92) of Paul Harrison’s book “Quantum Wells, Wires and Dots”.

## Output

The energy values were taken from the same file as before:

*bias\_00000/Quantum/wf\_energy\_spectrum\_quantum\_region\_Gamma\_0000.dat* (nextnano++)  
*Schredinger\_1band/ev\_cb1\_sg1\_deg1.dat* (nextnano<sup>3</sup>)

For example, the values for the 1 nm barrier read:

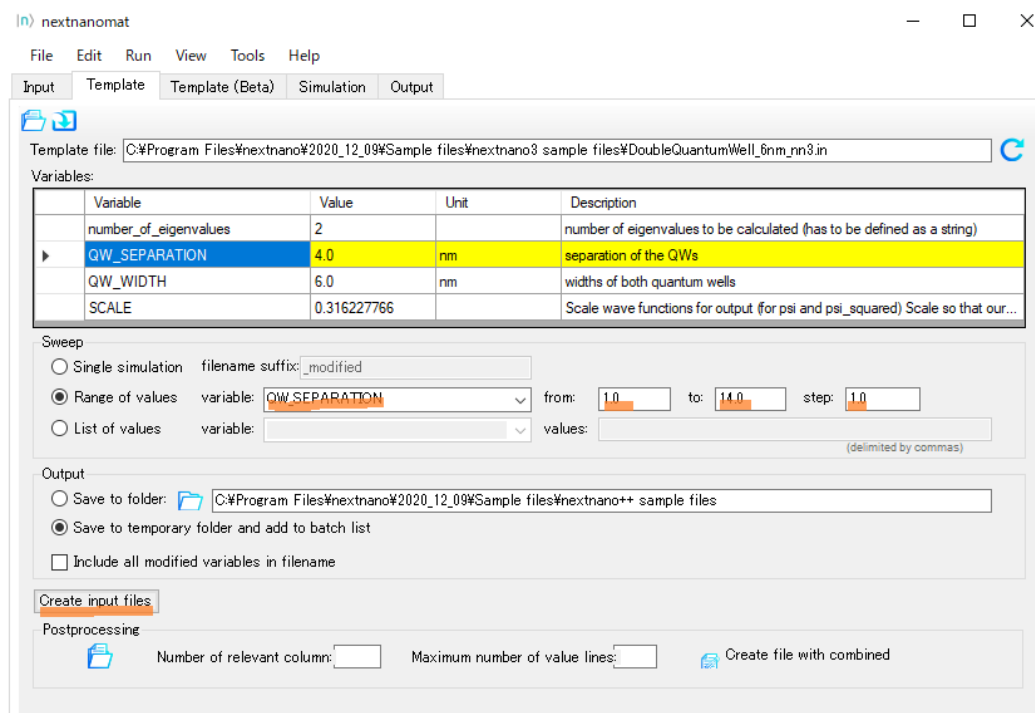
|                                 | nextnano++ | nextnano <sup>3</sup> | Harrison’s book |
|---------------------------------|------------|-----------------------|-----------------|
| ground state energy [eV]        | 0.03476    | 0.03476               | 0.03470         |
| first excited state energy [eV] | 0.07298    | 0.07298               | 0.07290         |

The values for the 14 nm barrier read:

|                                 | nextnano++ | nextnano <sup>3</sup> | Harrison’s book |
|---------------------------------|------------|-----------------------|-----------------|
| ground state energy [eV]        | 0.05332    | 0.05332               | 0.05323         |
| first excited state energy [eV] | 0.05338    | 0.05338               | 0.05329         |

## Tip: Sweeping

A sweep over the **thickness** of the Al<sub>0.2</sub>Ga<sub>0.8</sub>As barrier layer, i.e. the variable %QW\_SEPARATION, can easily be done by using nextnanomat’s Template feature. The following screenshot shows how this can be done. Go to “Template”, open input file, select “Range of values”, select “QW\_SEPARATION”, click on “Create input files”, go to “Run and start your simulations.



Another tutorial on coupled quantum wells can be found [here](#) .

Last update: nnnn/nn/nn

### 3.20.5 Scattering times for electrons in unbiased and biased single and multiple quantum wells

#### Input files:

- *1DGaAs\_AlGaAs\_10nmQW\_Lifetime.in*
- *1DGaAs\_AlGaAs\_12nmQW\_LifetimeFig5\_field.in*
- *1DGaAs\_AlGaAs\_SingleQW\_7nm.in*
- *1DGaAs\_AlGaAs\_DoubleQW\_7nm\_nonsymmetric.in*
- *1DGaAs\_AlGaAs\_DoubleQW\_LifetimeFig12\_field.in*

#### Note

If you want to obtain the input files that are used within this tutorial, please check if you can find them in the installation directory. If you cannot find them, please submit a Support Ticket.

#### Scope:

This tutorial tries to reproduce the results of [\[FerreiraBastard1989\]](#).

#### Scattering time as a function of quantum well width

Input file: *1DGaAs\_AlGaAs\_10nmQW\_Lifetime.in*

First, we want to study the electron lifetimes (scattering rates) of a single quantum well as a function of quantum well width `$QW_width`. (Note: Use *nextnanomat*'s Template feature to automatically sweep over the quantum well width.)

Our quantum well consists of *GaAs* that is sandwiched between two *Al<sub>0.3</sub>Ga<sub>0.7</sub>As* barriers. The material parameters that we are using for this tutorial are identical to the ones used in [\[FerreiraBastard1989\]](#), namely:

- electron mass:  $m_e = 0.07 m_0$
- conduction band offset:  $CBO = 0.2138$  eV
- static dielectric constant:  $\epsilon = 12.5$
- LO phonon energy:  $\hbar\omega_0 = 0.036$  eV (longitudinal optical phonon)

For the calculations, a grid resolution of 0.1 nm has been used.

```
quantum{
  region{
    ...
    momentum_matrix_elements{ # calculate dipole moment elements <i|p|j> for
↪intraband transitions
      Gamma{ }
    }
    dipole_moment_matrix_elements{ # calculate dipole moment elements <i|x|j> for
↪intraband transitions
      Gamma{ }
    }
  }
}
```

(continues on next page)

(continued from previous page)

```

transition_energies{ # calculate transition energies
  Gamma{}
}
lifetimes{ # calculate lifetimes
  Gamma{}
  phonon_energy = 0.036 # [eV]
}
}

```

The following two figures (Figure 3.20.5.1, Figure 3.20.5.2) show the conduction band edges and the lowest confined eigenstates (including the square of the wave functions) for a 6 nm and an 18 nm *AlGaAs/GaAs* quantum well.

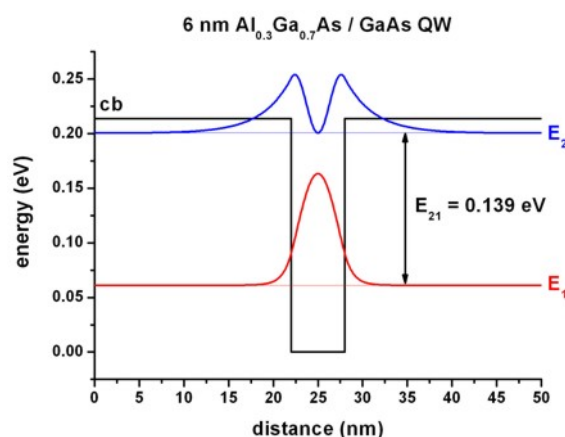


Figure 3.20.5.1: Calculated conduction band edge profile (black) and wave functions of confined electron states ( $E_1$  and  $E_2$ )

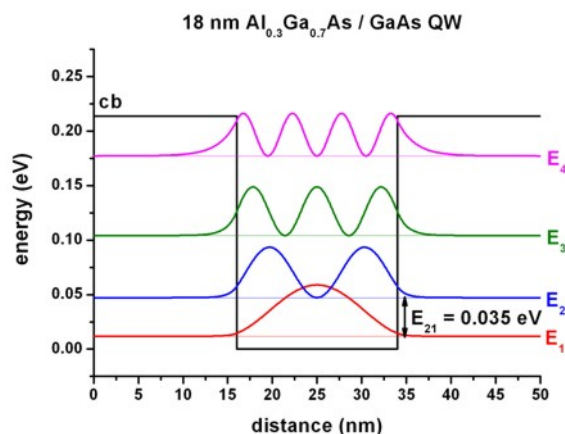


Figure 3.20.5.2: Calculated conduction band edge profile (black) and wave functions of confined electron states ( $E_1$ ,  $E_2$ ,  $E_3$  and  $E_4$ ).

The quantum well width can be varied easily by making use of the variable

```

$QW_width = 10 # (DisplayUnit:nm) (ListOfValues:5.2,5.4,5.6,5.8,6,7,8,10,12,14,15,16,
  ↪ 17,18,19,20)

```

which can be swept automatically using the *nextnanomat*'s Template feature. Open the input file and select "List of values" and variable "QW\_width".

Figure 3.20.5.3 shows the electron lifetime of the second eigenstate ( $E_2$  = initial state) to the ground state ( $E_1$  = final state), i.e. the intersubband transition with energy  $E_{21}$  for different quantum well widths. The temperature is set to 0 K.

For quantum well widths smaller than 5.4 nm ([*FerreiraBastard1989*]: 5.5 nm), only the ground state is confined and  $E_2$  is unbound. For quantum well widths larger than 18 nm ([*FerreiraBastard1989*]: 17.8 nm), the transition energy  $E_{21}$  is smaller than the LO phonon energy of 36 meV, thus scattering through the emission of an LO phonon is not possible anymore. The *nextnano*<sup>3</sup> calculations are in good agreement with Fig. 3 of [*FerreiraBastard1989*].

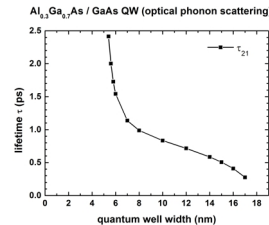


Figure 3.20.5.3: Calculated lifetimes  $\tau$  as a function of quantum well width

The output of the electron lifetime can be found in this file: *bias\_00000\Quantum\lifetimes\_quantum\_region\_Gamma.dat*.

```

...
                Intersubband dipole moment | < psi_f* | pz | psi_i > | [h_bar / e
↪[nm]
-----|-----
↪---

                Oscillator strength []
-----|-----|-----
↪---

                Energy of transition [eV]
-----|-----|-----|-----
↪---

                m* [m_0]      lifetime_
↪[ps]
-----|-----|-----|-----|-----
↪---

...
<psi001*|pz|psi002> 0.19717291    0.985747159    0.085864536    0.070000000    0.
↪833765805
...

```

Here, the shown values for the intersubband transitions correspond to a 10 nm QW.

### Scattering times as a function of electric field magnitude

Input file: *1DGaAs\_AlGaAs\_12nmQW\_LifetimeFig5\_field.in*

This input file will perform a sweep over the electric field strength. Figure 3.20.5.4 shows the lowest eigenstates of a 12 nm *AlGaAs/GaAs* QW at an applied electric field of -50 kV/cm. This time the conduction band edge is not flat anymore. It is tilted because of the electric field.

The sweep over the electric field magnitude can be done automatically. For these calculations, a grid resolution of 0.10 nm had been used. The *nextnano*<sup>3</sup> calculations presented in Figure 3.20.5.5 are in reasonable agreement with Fig. 5 in [*FerreiraBastard1989*].

### Single quantum wells

Input file: *1DGaAs\_AlGaAs\_SingleQW\_7nm.in*

The two confined energy levels and wave functions of the 7 nm single quantum well are shown in Figure 3.20.5.6. The energy of the ground state is 50.4 meV.

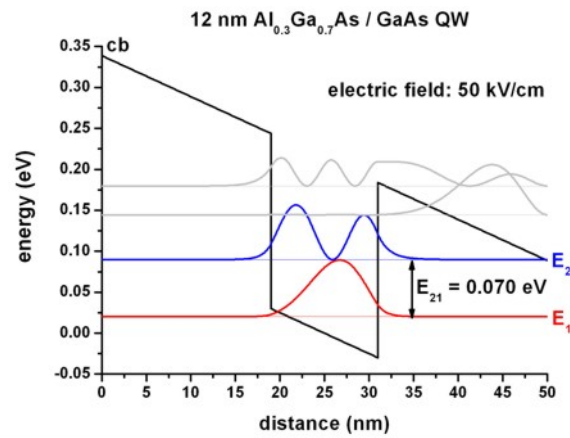


Figure 3.20.5.4: Calculated conduction band edge profile (black) and wave functions of confined electron states ( $E_1$  and  $E_2$ ), when electric field of -50 kV/cm is applied.

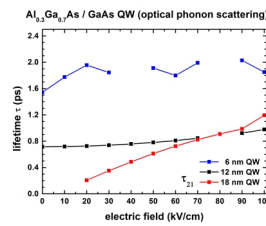


Figure 3.20.5.5: Calculated lifetimes  $\tau$  in a single quantum well as a function of applied electric field.

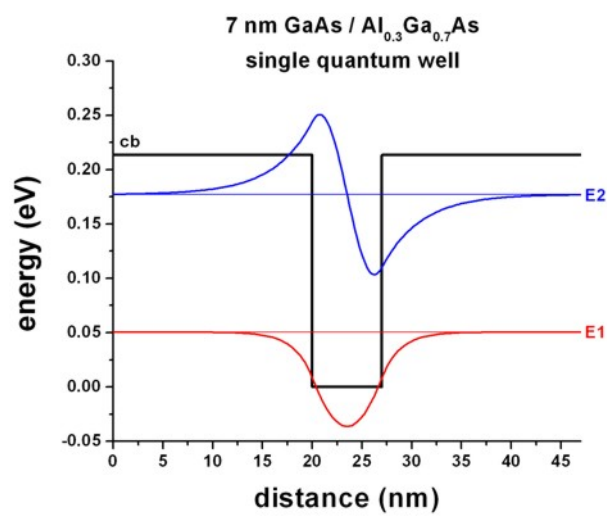


Figure 3.20.5.6: Calculated conduction band edge profile (black) and wave functions of confined electron states ( $E_1$  and  $E_2$ )

## Double quantum wells

Input file: *1DGaAs\_AlGaAs\_DoubleQW\_7nm\_nonsymmetric.in*

Here, we study the electron energy levels of a non-symmetric double quantum well structure as a function of quantum well width of the right quantum well: `$right_QW_width`. The right quantum well width can be varied easily by making use of the variable:

```
$right_QW_width = 7      # (DisplayUnit:nm) (ListOfValues:7.0,8.0,10.0,12.5,15.0,17.5,
→20.0,22.5,25.0,27.5,30.0,35.0,37.5,40.0,45.0,47.5,50.0,55.0,57.5,60.0,65.0,67.5,70.
→0,75.0,77.5,80.0,85.0,87.5,90.0,95.0,97.5,100.0)
```

which can be swept automatically using the *nextnanomat*'s Template feature. Open input file and select "List of values" and variable "right\_QW\_width". For the following figures, a grid resolution of 0.25 nm had been used.

Figure 3.20.5.7 shows the energy levels of a non-symmetric double quantum well structure (*GaAs* / *Al<sub>0.3</sub>Ga<sub>0.7</sub>As*) where the left quantum well always has the width 7 nm, and the right quantum well varies from 7 nm to 100 nm. The two *GaAs* wells are separated by a 5 nm *Al<sub>0.3</sub>Ga<sub>0.7</sub>As* barrier. The figure shows the energy levels as a function of the width of the larger quantum well.

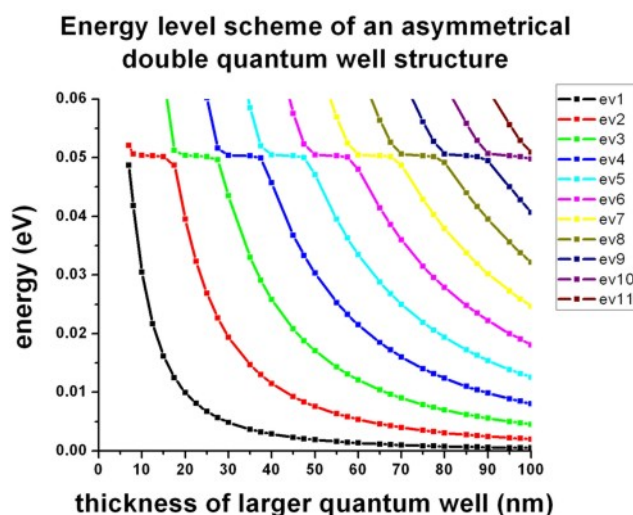


Figure 3.20.5.7: Calculated energy levels for different energy states as a function of  $L_{\text{QW, right}}$ .

One can see, that for certain widths of the larger quantum well, an anti-crossing due to bonding and anti-bonding states occurs. This happens whenever an eigenstate of the larger well matches the energy of the ground state of the smaller (7 nm) quantum well (which is at 50.4 meV, see example shown above: *1DGaAs\_AlGaAs\_SingleQW\_7nm.in*). Our calculations are in very good agreement with Fig. 9 in [FerreiraBastard1989].

The anti-crossing behavior and the plateaus at 50.4 meV of the energy level scheme (see Figure 3.20.5.7) can be illustrated by plotting the wave functions for different values of  $L_{\text{QW, right}}$ , see Figure 3.20.5.8, Figure 3.20.5.9, Figure 3.20.5.10 and Figure 3.20.5.11.

Figure 3.20.5.8 shows a symmetric double quantum well where both wells have the width 7 nm including the wave functions of the lowest confined states. If the barrier between these two wells had been very large, both wells would have had a ground state at 50.4 meV. However, due to the small barrier, coupling between these two wells becomes possible. The two lowest states form a bonding and an anti-bonding state. The bonding state now has a reduced energy of 48.7 meV and the anti-bonding state has an increased energy of 52.1 meV.

Figure 3.20.5.9 shows a non-symmetric double QW where the right QW has a width of 12.5 nm. In this case, the ground state can be found in the larger well, the second state in the 7 nm QW, whereas the third eigenstate is again localized in the larger well. Here, no bonding or anti-bonding states exist.

Figure 3.20.5.10 shows a non-symmetric double QW where the right QW has a width of 17.5 nm. In this case, the ground state can be again found in the larger well (similar to Figure 3.20.5.9), but this time, the third state of

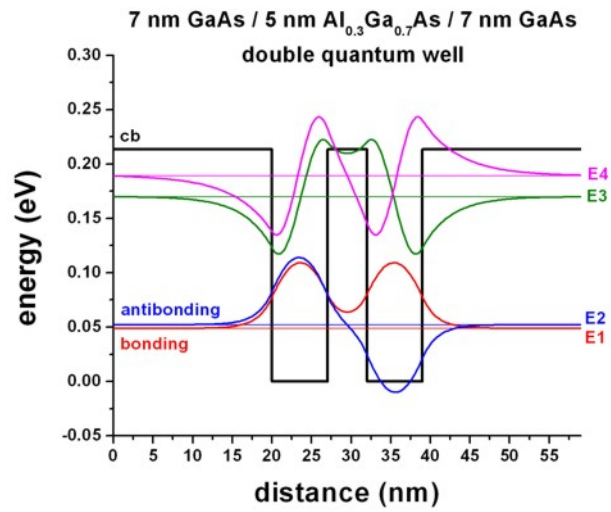


Figure 3.20.5.8: Calculated conduction band edge profile (black) for symmetric double quantum well:  $L_{\text{left}} = 7$  nm and  $L_{\text{right}} = 7$  nm, with wave functions of confined electron states.

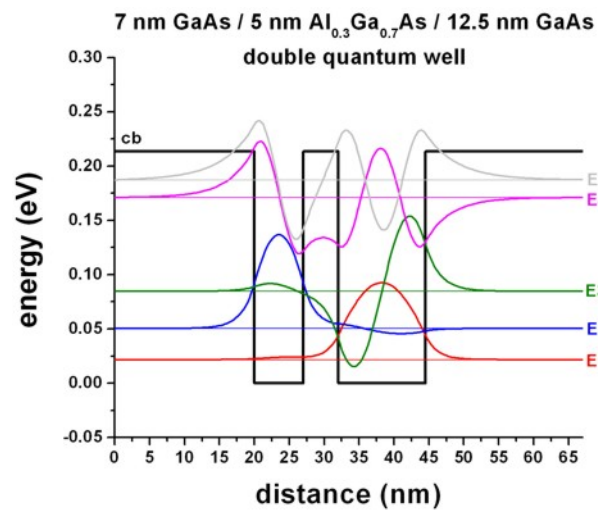


Figure 3.20.5.9: Calculated conduction band edge profile (black) for non-symmetric double quantum well:  $L_{\text{left}} = 7$  nm and  $L_{\text{right}} = 12.5$  nm, with wave functions of confined electron states.



moves down in energy and couples to the 7 nm ground state of the left well (compare with Figure 3.20.5.9). This coupling leads to the formation of a bonding and an anti-bonding states.

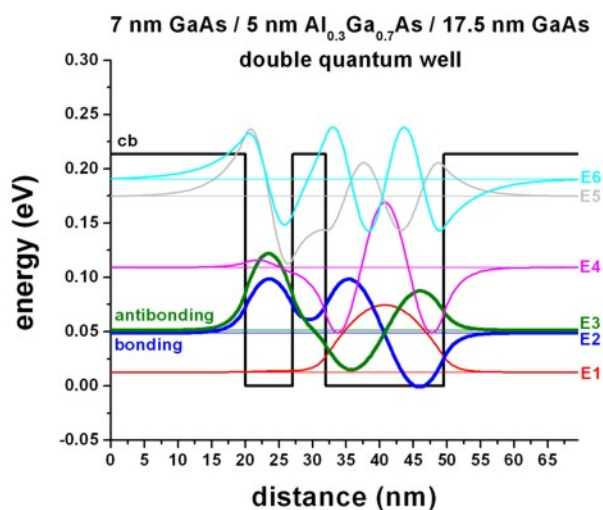


Figure 3.20.5.10: Calculated conduction band edge profile (black) for non-symmetric double quantum well:  $L_{\text{left}} = 7$  nm and  $L_{\text{right}} = 17.5$  nm, with wave functions of confined electron states.

Figure 3.20.5.11 shows a non-symmetric double QW where the right QW has a width of 25 nm. In this case, the ground state and the second state can be found in the larger well, whereas the third eigenstate is localized in the smaller (7 nm) well. The fourth eigenstate is localized in the larger well. Again, no bonding or anti-bonding states exist

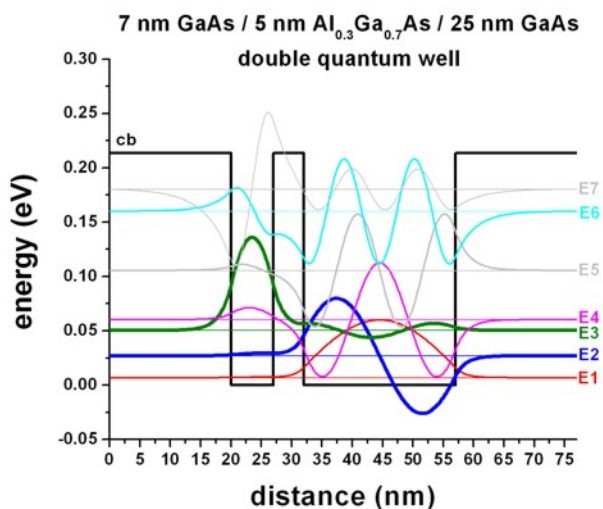


Figure 3.20.5.11: Calculated conduction band edge profile (black) for symmetric double quantum well:  $L_{\text{left}} = 7$  nm and  $L_{\text{right}} = 25$  nm, with wave functions of confined electron states.

### Biased double quantum well

Input file: *1DGaAs\_AlGaAs\_DoubleQW\_LifetimeFig12\_field.in*

Figure 3.20.5.12 shows the lifetime of the  $2 \rightarrow 1$  transition (“ground state of left quantum well to ground state of right quantum well transition”) as a function of electric field. The variable  $d$  is the thickness of the left well and the barrier region. The right well is assumed to have the same thickness as the left quantum well, i.e.  $d/2$ .

The parameter  $d$  can be varied easily by making use of the variable

```
$QWBarrierThickness = 6      # (DisplayUnit:nm) (ListOfValues:6,9)
```

which can be swept automatically using the *nextnanomat*'s Template feature. Open input file and select "List of values" and variable "QWBarrierThickness".

There seems to be qualitative agreement to Fig. 12 in [FerreiraBastard1989]. For  $d = 9$  nm, the LO phonon emission is forbidden for electric fields smaller than  $\sim |40 \text{ kV/cm}|$  because the transition energy is smaller than the LO phonon energy of 36 meV.

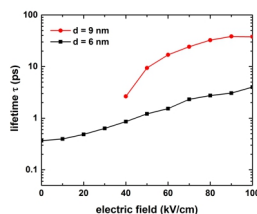


Figure 3.20.5.12: Calculated lifetimes  $\tau$  in a single quantum well as a function of applied electric field.

Last update: 27/05/2025

### 3.20.6 Exciton Binding Energy in an Infinite Quantum Well

#### Input File:

*1D\_exciton\_binding\_energy\_infinite\_QW\_nnp.in* *1D\_exciton\_binding\_energy\_infinite\_QW\_nn3.in*

#### Content:

In this tutorial we study the *exciton* binding energy between the electron ground state and heavy hole ground state ( $e_1 - hh_1$ ) in a single quantum well (ZnSe/CdTe/ZnSe). This energy correction is crucial, for example, when correlating computed optical transition energies in quantum wells with experimental results.

We aim to reproduce figures 6.4 (p. 196) and 6.5 (p. 197) of Paul Harrison's excellent book "*Quantum Wells, Wires and Dots*" [HarrisonQWWD2005], thus the following description is based on the explanations made therein. We are grateful that the book comes along with a CD so that we were able to look up the relevant material parameters and to check the results for consistency.

#### Output Files:

*\bias00000\Quantum\exciton\_spectrum\_QuantumRegion\_Gamma\_HH.dat*

#### Description of analytical formulas

We present briefly the analytical formulas for the exciton binding energy in 1) bulk material and 2) quantum well structure (type-I). A full derivation can be found in [HarrisonQWWD2005].

#### 1) Bulk

The 3D bulk exciton binding energy can be calculated analytically

$$E_{\text{ex,b}} = -\frac{\mu e^4}{32\pi^2 \hbar^2 e_r^2 e_0^2} = -\frac{\mu}{m_0 e_r^2} \cdot 13.61 \text{ eV},$$

where

- $\mu$  is the reduced mass of the electron-hole pair
- $\hbar$  is Planck's constant divided by  $2\pi$
- $e$  is the electron charge

- $\epsilon_r$  is the dielectric constant ( $\epsilon_{r,\text{GaAs}} = 12.93$ ,  $\epsilon_{r,\text{CdTe}} = 10.6$ )
- $\epsilon_0$  is the vacuum permittivity
- $m_0$  is the rest mass of the electron and
- 13.61 eV is the Rydberg energy.

The reduced mass of the electron–hole pair  $\mu$  is calculated by

$$\frac{1}{\mu} = \frac{1}{m_e} + \frac{1}{m_h},$$

with the effective masses of electrons and holes:  $m_e$  and  $m_h$ .

### Example

The *reduced mass* of GaAs and CdTe are

$$\frac{1}{\mu_{\text{GaAs}}} = \frac{1}{0.067} + \frac{1}{0.5} \Rightarrow \mu_{\text{GaAs}} = 0.0591$$

$$\frac{1}{\mu_{\text{CdTe}}} = \frac{1}{0.096} + \frac{1}{0.6} \Rightarrow \mu_{\text{CdTe}} = 0.0828$$

with respective *Bohr radius*

$$\lambda_{\text{GaAs}} = 11.6 \text{ nm}$$

$$\lambda_{\text{CdTe}} = 6.8 \text{ nm}.$$

From the *3D bulk exciton binding energies*

$$E_{\text{ex,b}}(\text{GaAs}) = -4.8 \text{ meV}$$

$$E_{\text{ex,b}}(\text{CdTe}) = -10.0 \text{ meV}$$

the energy of the band gap transition including excitonic effects reads:

$$E_{\text{ex, GaAs}} = E_{\text{gap}} + E_{\text{ex,b}} = 1.519 \text{ eV} - 0.005 \text{ eV} = 1.514 \text{ eV}$$

$$E_{\text{ex, CdTe}} = E_{\text{gap}} + E_{\text{ex,b}} = 1.606 \text{ eV} - 0.010 \text{ eV} = 1.596 \text{ eV}$$

## 2) Quantum well (type-I)

Analytical results for the exciton ground state transition ( $e_1\text{--}hh_1$ ) of a 1D quantum well (type-I) are only obtainable in the following two limits:

- infinitely thin quantum well (*2D limit*)

$$E_{\text{ex,QW}} = 4 \cdot E_{\text{ex}}$$

$$\lambda_{\text{ex,QW}} = \frac{\lambda_{\text{ex}}}{2}$$

- infinitely thick quantum well (*3D bulk exciton limit*)

$$E_{\text{ex,QW}} = E_{\text{ex}}$$

$$\lambda_{\text{ex,QW}} = \lambda_{\text{ex}}$$

Between these limits, the exciton correction, which depends on the well width, has to be calculated numerically, not only for the ground state but also for excited states (e.g.  $e_2\text{--}hh_2$ ,  $e_1\text{--}lh_1$ ).

## Numerical calculation

Our numerical approach is based on a variational principle. We use the separable wave function

$$\psi(r) = \sqrt{\frac{2}{\pi}} \frac{1}{\lambda} \exp\left(-\frac{r}{\lambda}\right), \quad (3.20.6.1)$$

see e.g. p. 562, Eq. (13.4.27), Section 13.4.3 *Variational Method for Exciton Problem* in [\[ChuangOpto1995\]](#) or [\[BastardPRB1982\]](#). The excitonic binding energy is then minimized with respect to the variational parameter  $\lambda$  (= Bohr radius).

## Simulation

We study the exciton binding energy of a CdTe quantum well (with infinite barriers) as a function of well width. We chose infinite barriers, in order to be able to compare the *nextnano++* calculations with standard textbook results, originally published by [\[BastardPRB1982\]](#).

## Input file

The material parameters used for CdTe are the following:

```
database{
  binary_zb{
    name = "CdTe"
    conduction_bands{
      Gamma{ mass = 0.096 }
      ...
    }
    valence_bands{
      HH{ mass = 0.6 }
      ...
    }
    dielectric_consts{
      static_a = 10.6
    }
    ...
  }
}
```

In order to calculate the exciton correction energy, the following group inside `quantum{ }` has to be used:

```
quantum{
  ...
  region{
    ...
    excitons{
      dielectric_const = 10.6
      electron_mass = 0.096
      hole_mass = 0.6
      energy_cutoff = 1000
      accuracy = 1e-10
    }
    ...
  }
}
```

## Parameter Sweep

The following screenshot ([Figure 3.20.6.1](#)) shows how to use the *Template* feature of *nextnanomat* in order to calculate the exciton binding energy as a function of the quantum well width.

Initialization and execution of parameter sweep:

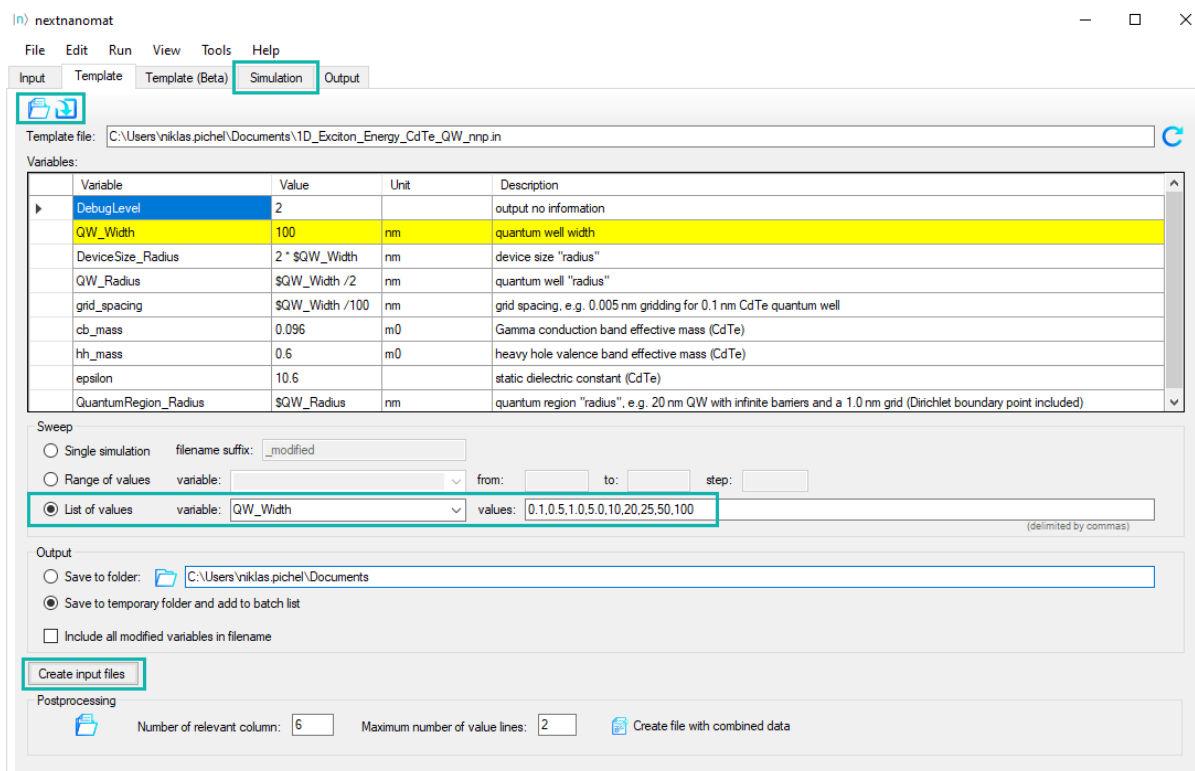


Figure 3.20.6.1: Initializing parameter sweep for QW\_Width in tab Template

- Open input file in Template tab.
- Select *List of values*, select variable QW\_Width. The corresponding list of values are loaded from the template input file.
- Click on *Create input files* to create an input file for each quantum well width.
- Switch to *Simulation* tab and start the batch list of jobs.

## Results

Figure 3.20.6.2 shows the exciton binding energy in an infinitely deep quantum well as a function of well width. The quantities are given in units of the 3D bulk exciton energy  $E_{\text{ex}}$  (also called *effective Rydberg energy*) and in units of the 3D bulk exciton Bohr radius  $\lambda_{\text{ex}}$  respectively. We see that the 3D limit is not reproduced correctly in our approach. To obtain the 3D limit, a nonseparable wave function  $\psi(r, z_e, z_h)$  has to be used instead of (3.20.6.1).

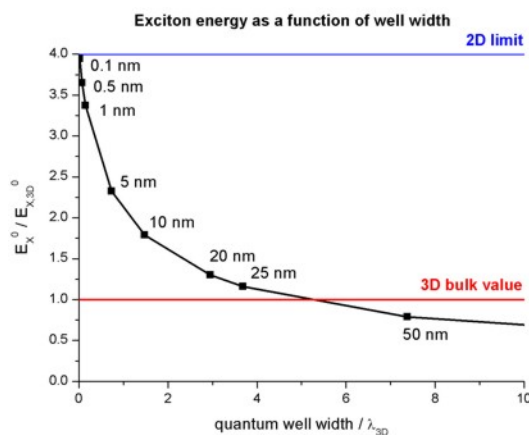


Figure 3.20.6.2: Exciton energy as a function of quantum well width

Figure 3.20.6.3 shows the exciton binding energy in an infinitely deep CdTe quantum well as a function of well width. The *nextnano++* results are in agreement with fig. 6.4 of [HarrisonQWWD2005], although we use a simpler trial wave function with only one variational parameter.

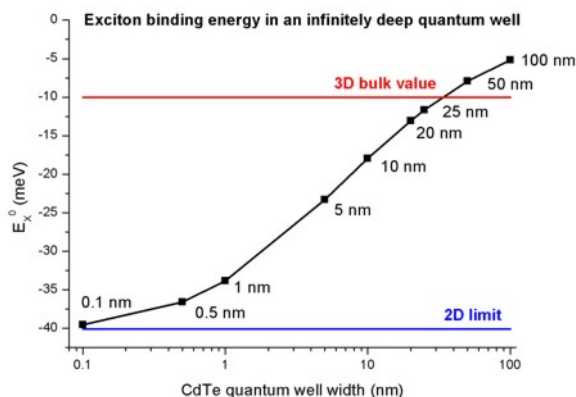


Figure 3.20.6.3: Exciton binding energy in an infinitely deep quantum well

Figure 3.20.6.4 shows the exciton Bohr radius as a function of well width

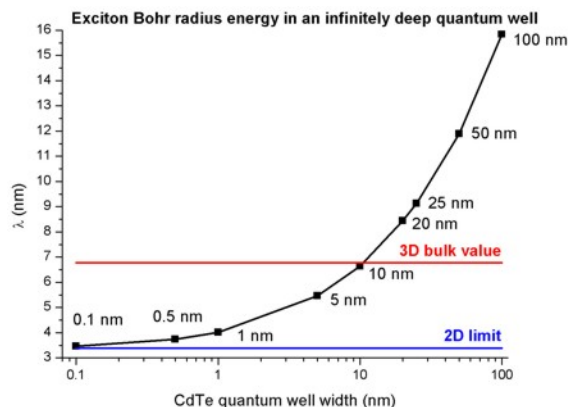


Figure 3.20.6.4: Exciton Bohr radius energy in an infinitely deep quantum well

### nextnano<sup>3</sup> tool

The following comments are specific to the input file syntax of *nextnano<sup>3</sup>*.

The material parameters used are the following (*\$binary-zb-default*).

```
!-----!
! Here we are overwriting the database entries for CdTe. !
!-----!
$binary-zb-default
binary-type           = CdTe-zb-default
apply-to-material-numbers = 2
conduction-band-masses = 0.096    0.096    0.096    ! Gamma    [m0]
...
valence-band-masses   = 0.6       0.6       0.6       ! heavy hole [m0]
...
static-dielectric-constants = 10.6    10.6    10.6    ! []
```

In order to calculate the exciton correction, the following flags have to be used:

```
$numeric-control
simulation-dimension      = 1
calculate-exciton         = yes    ! to switch on exciton correction
exciton-electron-state-number = 1    ! electron ground state
exciton-hole-state-number  = 1    ! hole    ground state
```

The output of the exciton binding energies can be found in this file: Schroedinger\_1band/exciton\_energy1D.dat

The output for the 5 nm CdTe QW looks like this:

```
Exciton correction for 1D quantum wells (type-I structures)
=====
static dielectric constant      = 10.6000000000 []
effective mass electron         = 0.0960000000 [m0]
effective mass hole             = 0.6000000000 [m0]
reduced mass                   = 0.0827586207 [m0]
Bulk Bohr radius of 3D exciton = 6.7778780735 [nm]
Bulk 3D exciton energy         = -10.0212560410 [meV]

lambda [nm]      exciton energy [meV]      exciton energy [Rydberg]
0.338893904E+001 -0.158496790E+002          0.158160603E+001
...
0.421888329E+001 -0.215591082E+002          0.215133793E+001
...
0.553296169E+001 -0.232757580E+002          0.232263879E+001
-----
Calculated lambda and exciton energy:
0.546379967E+001 -0.232817837E+002          0.232324009E+001
-----
```

The last iteration yields -23.28 meV for the exciton binding energy.  $\lambda$  is the variational parameter  $\lambda$  which is equivalent to the exciton Bohr radius in units of (nm).

Last update: nnnn/nn/nn

### 3.20.7 GaAs solar cell

- *Header*
- *Input files*
- *Reference*
- *Structure*
- *Simulation procedure*
- *How does a solar cell work? & How do we simulate it?*
  - *1. Solar spectrum*
  - *1. Generation rate (internal calculation)*
  - *1. Generation rate (import)*
  - *4. Current-Voltage characteristics*

## – 5. Solar efficiency

### Header

#### Files for the tutorial:

- *1DGaAs\_SolarCell\_nnp.in*
- *1DGaAs\_SolarCell\_nnp\_import\_generation.in*
- *1DGaAs\_SolarCell\_nnp\_local\_absorption.in*
- *1DGaAs\_SolarCell\_nnp\_complex\_refractive\_index.in*

Here we demonstrate that solar cells can be simulated using nextnano. The self-consistent solutions to the Poisson equation coupled with current (drift-diffusion) equation give the figure of merit of solar cells that consists of arbitrary materials. Current-Voltage (I-V) curves and corresponding power and solar cell efficiency as a function of bias voltage are exported to the output folder.

### Input files

Here the numerics parameters are optimized for convergence of the calculation in the bias range of interest. Please pay attention to the convergence of the calculation when you change device geometry etc.

In the simulation of input files *1DGaAs\_SolarCell\_nnp.in* and *1DGaAs\_SolarCell\_nn3.in*, the following data are used to calculate generation rate  $G(E, x)$  internally:

- Absorption spectrum  $\alpha(E)$
- Reflectivity  $R(E)$
- Solar spectral irradiance

In *1DGaAs\_SolarCell\_nnp.in* (nextnano++), these data are already specified in *database\_optional.in* for some materials. For example, you can use these by specifying `irradiation{}` as follows:

```
classical{
  ...

  irradiation{
    ...

    global_illumination{
      direction_x = 1

      database_spectrum{
        name = "Solar-ASTM-G173-global"
        concentration = 1.0
      }
    }

    global_reflectivity{
      database_spectrum{
        name = "Al0.80Ga0.20As"
      }
    }

    global_absorption_coeff{
      database_spectrum{
        name = "GaAs"
      }
    }
  }
}
```

(continues on next page)



(continued from previous page)

```

    }
  }
}

```

If you want to use the materials that are not in the database or rewrite the database, you can specify the new data in `database{ }` as you want.

In `1DGaAs_SolarCell_nn3.in` (nextnano<sup>3</sup>), you need to prepare the above three data as external files to calculate generation rate  $G(E, x)$  internally. We have the external data files, whose data are identical with those used in `1DGaAs_SolarCell_nnp.in`, for nextnano<sup>3</sup>.

You can also import the data of generation rate itself. In the simulation of `1DGaAs_SolarCell_nnp_import_generation.in` and `1DGaAs_SolarCell_nn3_import_generation.in`, the following output file of nextnano<sup>3</sup> must be read in.

- `/optics/GenerationRateLight_vs_Position_sun1.dat`

This data file is also in the sample file folder.

## Reference

- J. Nelson, *The Physics of Solar Cells* (Imperial College Press, 2003)
- S.M. Sze and Kwok K. Ng, *Physics of Semiconductor Devices* (Wiley, 2007)

## Structure

Figure 3.20.7.1 shows the band edges and quasi Fermi levels of the device. The device structure is as follows:

- 0-30 nm Al<sub>0.8</sub>Ga<sub>0.2</sub>As Window layer
- 30-530 nm p-doped GaAs
- 530-3530 nm n-doped GaAs
- 3530-3630 nm n-doped GaAs back surface field layer

Strain is not calculated in this example.

The left side of the device ( $x=0$  nm) is illuminated by the sun. As shown in Figure 3.20.7.6, mobile electrons and holes are created mainly in the p-layer. Electrons then flow to the right because of the AlGaAs ternary barrier (0-30 nm), and holes to the left. The back of the cell (3530-3630 nm) is doped with 10 times larger concentration, so that it prevents the minority carrier (hole) from leaking to the right contact. Since the current from p-layer to n-layer is defined to be positive, the photo-induced current has negative sign.

## Simulation procedure

The workflow of the simulation is summarized in Figure 3.20.7.2. To obtain the figures shown in this tutorial,

1. Specify in the input file the three data, namely (1) spectral irradiance (solar spectrum), (2) reflectivity at the front surface and (3) absorption spectrum. (Referring the database or rewriting the database)

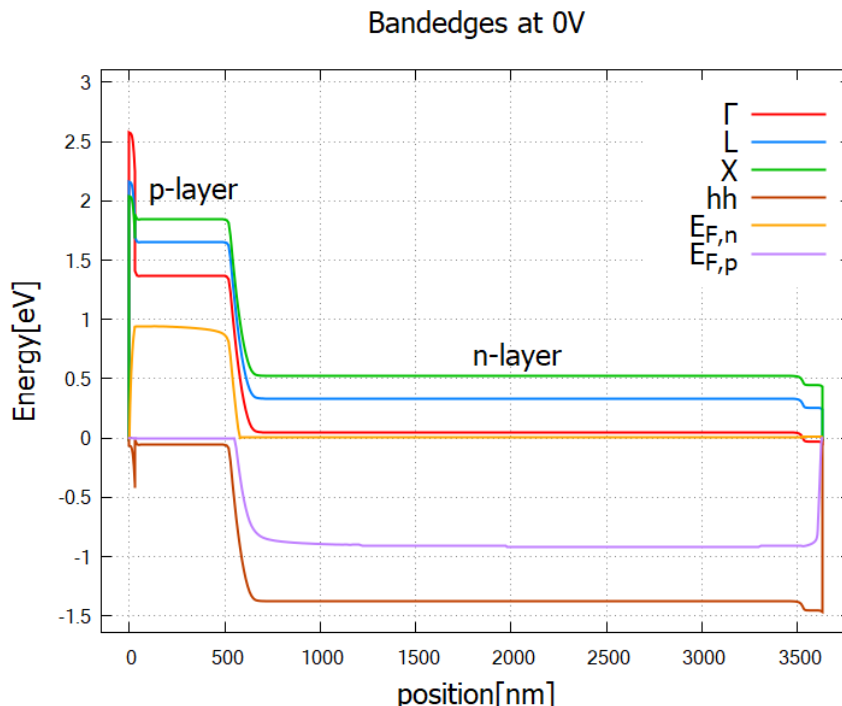


Figure 3.20.7.1: Band edges and quasi-Fermi levels of the solar cell at zero bias *bias\_000000/bandedges.dat* (*nextnano++*) / *band\_structure/BandEdges.dat* (*nextnano<sup>3</sup>*)

2. Run *nextnano++*, and all of your *nextnano++* results are in your output folder! Generation rate  $G(E, x)$  is internally calculated before the current-Poisson iteration starts. The efficiency-voltage curve is generated as a final result.
3. **If you already have generation rate profile as a .dat file, you**  
can either import it into *nextnano++* or in *nextnano<sup>3</sup>*.

## How does a solar cell work? & How do we simulate it?

### 1. Solar spectrum

The sun emits light with a range of wavelengths ranging from the ultraviolet, visible to infrared region. The extraterrestrial solar spectrum resembles the spectrum of a black body at  $T_{\text{sun}} = 5760\text{K}$  [Nelson Chapter 2]:

$$\frac{2\pi \sin^2 \theta_{\text{sun}}}{h^3 c^2} \frac{E^2}{e^{E/k_B T_{\text{sun}}} - 1},$$

where  $E$  is the photon energy and  $\theta_{\text{sun}} = 1.44 \times 10^{-3} \pi [\text{rad}]$  when measured from the earth. The solar light travels from the sun to the earth, and then from the outer space to our solar cell devices, during which the spectrum attenuates and changes its shape. The standard solar spectrum assumed in solar cell analysis is called AM1.5G (AM = air mass), which takes into account the attenuation of the intensity and illumination from all angles (rather than direct from the sun) due to scattering in the atmosphere. The spectral photon flux, i.e. the spectrum of the number of incident photons per area per time, is denoted by  $\phi(E)$  [ $\text{m}^{-2}\text{s}^{-1}\text{eV}^{-1}$ ]. The spectral irradiance, namely the spectrum of the amount of energy supplied per area per time, is given by  $L(E) = E\phi(E)$  with the unit of [ $\text{Wm}^{-2}\text{eV}^{-1}$ ]. We have taken the AM1.5G spectral irradiance data from [this website](#) (Figure 3.20.7.3). If you have space applications in mind, please use the extraterrestrial spectrum, namely air mass zero (AM0).

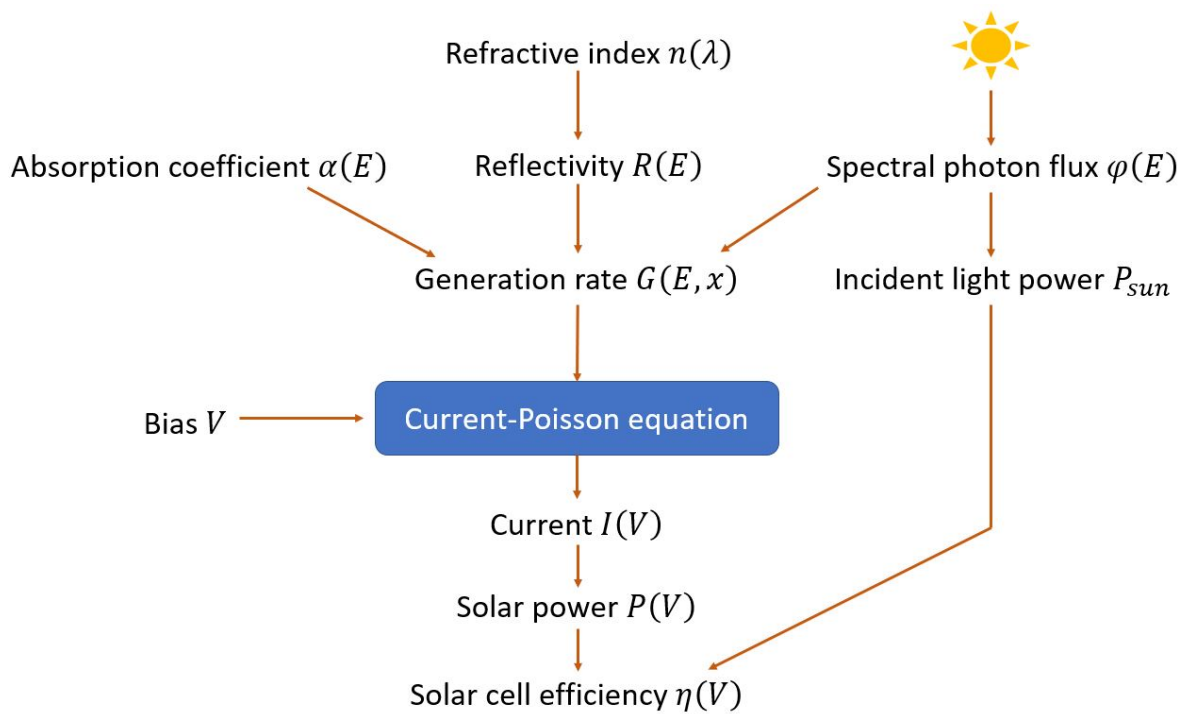


Figure 3.20.7.2: Workflow of solar cell simulation. Each quantity is explained in the following section.

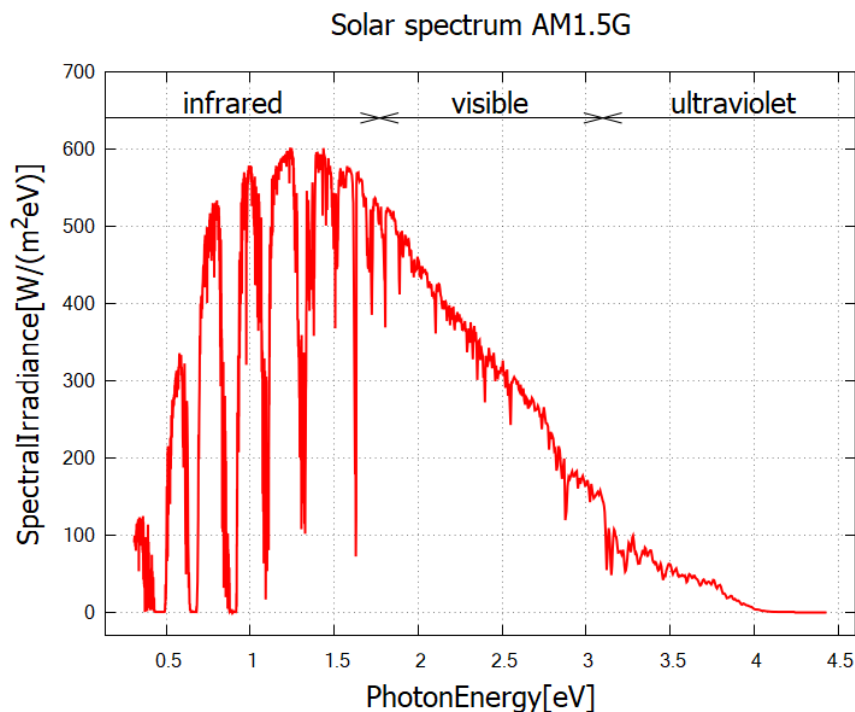


Figure 3.20.7.3: The AM1.5G spectral irradiance  $L(E)$ , that is, the solar spectrum measured on the earth. The *nextnano++* tool reads from the predefined data `Solar-ASTM-G173-global` and stores it in the output file `Irradiation/illumination_spectrum_power_eV.dat`. The *nextnano<sup>3</sup>* tool reads in the data file `ASTMG173_AM15G.dat` and stores it in the output file `optics/SolarSpectralIrradiance_eV.dat`.

The power of incident light

$$P_{\text{sun}} = \int_0^{\infty} L(E) dE = 1000 \text{ Wm}^{-2},$$

is solely determined by the condition of the sun and the atmosphere of the earth (for AM0  $P_{\text{sun}} = 1353 \text{ Wm}^{-2}$ ). The ultimate challenge of solar cell research is to achieve the most efficient conversion of this energy input into electric power  $P_{\text{out}} [\text{Wm}^{-2}]$ . The figure of merit is therefore defined as  $\eta = \frac{P_{\text{out}}}{P_{\text{sun}}}$ .

## 1. Generation rate (internal calculation)

(If you already have available data for generation rate, you can skip this section.)

When the sunlight illuminates the device, some photons are reflected at the front surface (air-semiconductor interface) and the rest enters the device. This effect is taken into account by considering the reflectivity of  $\text{Al}_{0.8}\text{Ga}_{0.2}\text{As}$ . Through the absorption of one photon, a pair of mobile electron and hole is created, while the photon flux attenuates exponentially with respect to the penetration depth. The generation rate thus depends not only on the incident photon flux  $\phi(E)$  but also on the absorption coefficient  $\alpha(E)$  of the material and the reflectivity  $R(E)$  at the surface (Figure 3.20.7.2):

$$G(E, x) = N\phi(E) \cdot (1 - R(E)) \cdot \alpha(E)e^{-\alpha(E)x},$$

where “the number of suns”  $N$  is multiplied to the photon flux  $\phi(E)$  to take into account the concentration of sunlight. The corresponding keyword is `concentration` (*nextnano++*) / `number-of-suns` (*nextnano<sup>3</sup>*).

In the sample input file for *nextnano++*, predefined value is used for  $L(\lambda)$ .  $\alpha(\lambda)$  and  $R(\lambda)$  are defined in `database{ }`. In the group `optics{ }`, one can specify which data to use as those variables. These spectra are translated into  $\alpha(E)$ ,  $R(E)$ ,  $N\phi(E)$  and substituted into the generation rate formula.

```
# nextnano++
optics{
    ...

    global_illumination{
        direction_x = 1

        database_spectrum{
            name = "Solar-ASTM-G173-global"
            concentration = 1.0          # e.g. 1 sun
        }
    }

    global_reflectivity{
        database_spectrum{
            name = "Al0.8Ga0.2As"
        }
    }

    global_absorption_coeff{
        database_spectrum{
            name = "GaAs"
        }
    }
}
```

In the input file for *nextnano<sup>3</sup>*,  $\alpha(\lambda)$ ,  $R(\lambda)$ ,  $L(\lambda)$  and  $N$  are imported as specified in keyword *\$Optical-absorption*.

```
! nextnano3
$Optical-absorption
```

(continues on next page)

(continued from previous page)

```

...
import-absorption-spectrum = yes
file-absorption-spectrum  = "(directory)\AbsorptionCoefficient_GaAs_300K.dat"

import-reflectivity-spectrum = yes
file-reflectivity-spectrum  = "(directory)\Reflectivity_Al0.80Ga0.20As.dat"

import-solar-spectrum       = yes
file-solar-spectrum        = "(directory)\ASTMG173_AM15G.dat" ! G = global, i.e. ↪
↪including diffuse light

!number-of-suns             = 0.0   ! switch off sun if generation rate is imported
number-of-suns              = 1.0
!number-of-suns             = 100.0 ! optical concentration
$end_optical-absorption

```

If no reflectivity data is specified, perfect interface (zero reflection) is assumed. The reflectivity data of  $\text{Al}_{0.8}\text{Ga}_{0.2}\text{As}$  used in this simulation (`optical_reflectivity` in `database_nnp_optional.in` for `nextnano++`, `Reflectivity_Al0.80Ga0.20As.dat` for `nextnano3`) have been generated through the Fresnel formula for perpendicular incident light

$$R(\lambda) = |r(\lambda)|^2 = \left| \frac{1 - [n(\lambda) + i\kappa(\lambda)]}{1 + [n(\lambda) + i\kappa(\lambda)]} \right|^2$$

where the refractive index  $n$  and extinction coefficient  $\kappa$  of GaAs and AlAs are taken from [here](#). To obtain the values of ternary  $\text{Al}_{0.8}\text{Ga}_{0.2}\text{As}$ , we performed linear interpolation.

If you consider a textured surface to reduce surface light reflection, please prepare the corresponding reflectivity data and import to the nextnano simulation. For `nextnano++`,  $\alpha(E)$ ,  $R(E)$  and  $L(E)$  are stored in the output folder `Irradiation` with file names `absorption_spectrum_eV.dat`, `reflectivity_spectrum_eV.dat` and `illumination_spectrum_eV.dat`, respectively. For `nextnano3`, they are stored in the output folder `optics` with file names `AbsorptionCoefficient.dat`, `Reflectivity.dat` and `SolarSpectralIrradiance.dat`, respectively.

The resulting generation rate is shown in [Figure 3.20.7.4](#), [Figure 3.20.7.5](#) and [Figure 3.20.7.6](#).

### 1. Generation rate (import)

If the generation rate data  $G(x) = \int G(E, x) dE$  ([Figure 3.20.7.6](#)) is available from literature or publications, you can import the `.dat` file without worrying about the above-mentioned calculation. The data must contain position [nm] in the first column and generation rate [ $10^{18}\text{cm}^{-3}\text{s}^{-1}$ ] in the second. In the sample file `1DGaAs_SolarCell_nnp_import_generation.in` (`nextnano++`) and `1DGaAs_SolarCell_nn3_import_generation.in` (`nextnano3`), we import the data generated from `1DGaAs_SolarCell_nn3.in`.

```

# nextnano++
structure{
  region{
    everywhere{}
    generation{
      import{ import_from = "GenImportProfile" }
    }
  }
}

import{
  file{
    name      = "GenImportProfile"
    filename  = "(directory path)\GenerationRateLight_vs_Position_sun1.dat"
    format    = DAT
  }
}

```

(continues on next page)

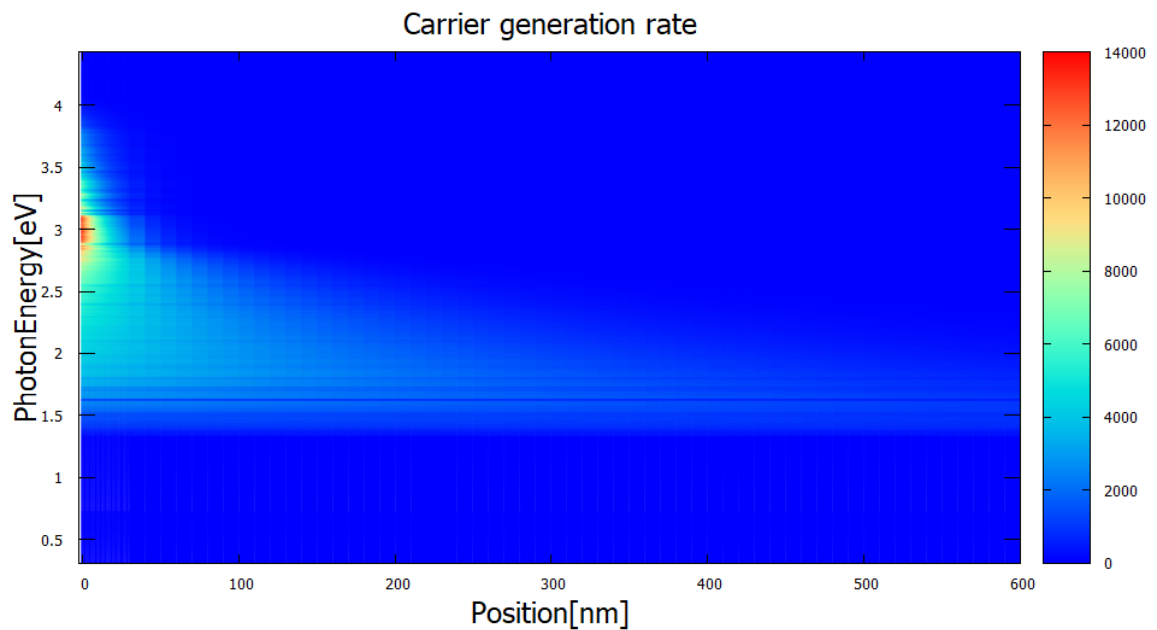


Figure 3.20.7.4: Generation rate as a function of position and energy *optics/GenerationRateLight\_vs\_Position\_and\_Energy\_2Dplot\_sun1.dat* (nextnano<sup>3</sup>) in units of  $10^{18} \text{cm}^{-3} \text{eV}^{-1} \text{s}^{-1}$ . The corresponding file for nextnano++ is *Irradiation/photo\_generation\_energy\_resolved.fld*. This quantity is internally calculated using the absorption coefficient, reflectivity of the front surface and solar spectrum AM1.5G (Figure 3.20.7.3). Photons at around 3V are largely absorbed near the front surface due to a large absorption coefficient, which can be seen in the output *optics/AbsorptionCoefficient\_eV.dat/Irradiation/absorption\_spectrum\_eV.dat* (not shown). Photons with lower energy, in contrast, travel a longer distance in the device.

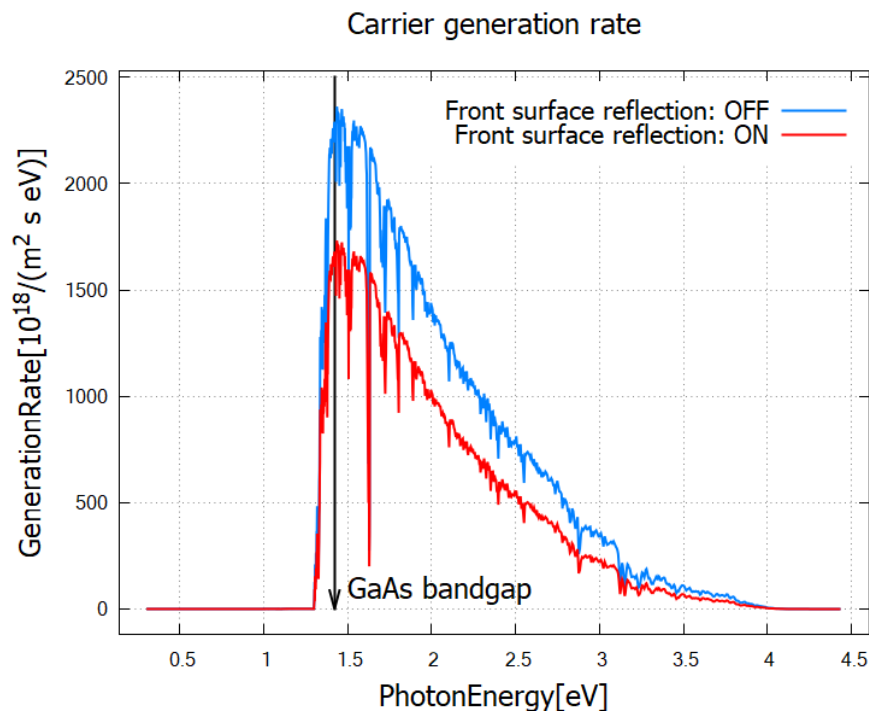


Figure 3.20.7.5: Generation rate as a function of energy *optics/GenerationRateLight\_vs\_Energy\_sun1.dat* (*nextnano<sup>3</sup>*), red curve. The corresponding file for *nextnano++* is *Irradiation/photo\_generation\_integrated.dat*. Also shown is the result for zero reflection at the front surface (blue curve). Obviously, the generation rate becomes larger when the reflection at the front surface is neglected. One can also clearly see, by comparing with [Figure 3.20.7.4](#), that the low energy photons below the band gap cannot contribute to the carrier generation. For this reason the band gap of semiconductors affects the solar cell efficiency and is discussed in the context of the [Shockley-Queisser efficiency limit](#).

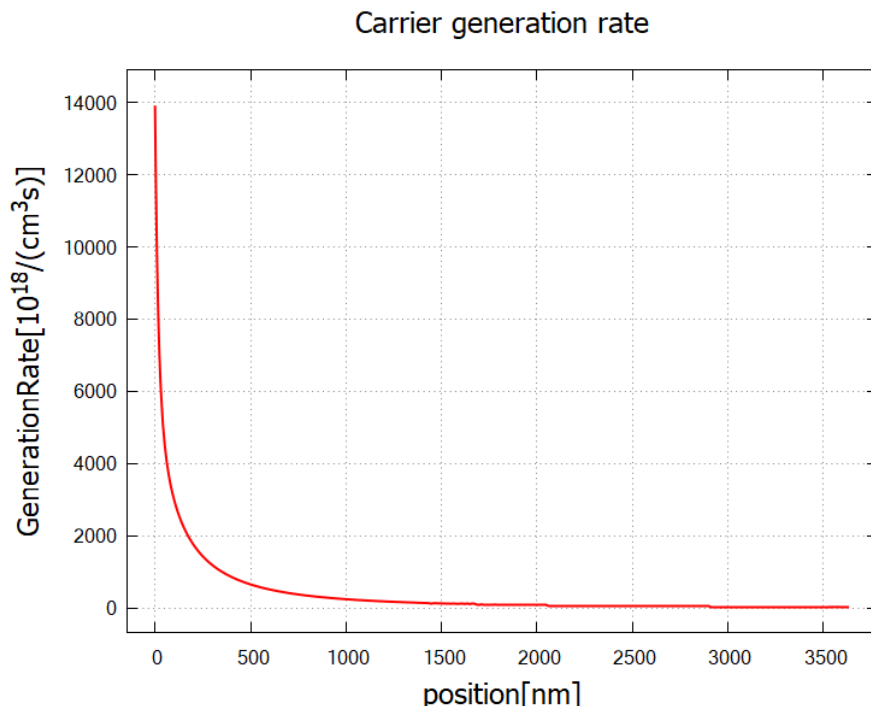


Figure 3.20.7.6: Generation rate as a function of position *optics/GenerationRateLight\_vs\_Position\_sun1.dat* (*nextnano<sup>3</sup>*). The corresponding file for (*nextnano++*) is *Irradiation/photogeneration.dat*. This data is obtained by integrating Figure 3.20.7.4 over energy  $E$ . When the photon flux travels through the device, the intensity diminishes exponentially, leading to the exponential decrease in generation rate. Most of the carrier generation occurs within 500 nm from the front surface, i.e. within the p-layer (30–530 nm).

(continued from previous page)

```

    scale      = 1e18 # import data is multiplied by this scaling factor (optional,
    ↪ default value is 1.0)
  }
}

```

```

! nextnano3
$import-data-on-material-grid
  source-directory = "(directory path)"
  import-generation = yes
  filename-generation = "GenerationRateLight_vs_Position_sun1.dat"
$end_import-data-on-material-grid

$optical-absorption
...
  number-of-suns = 0.0 ! switch off sun if generation rate is imported
$end_optical-absorption

```

#### 4. Current-Voltage characteristics

The calculated or imported generation rate contributes to the right-hand side of the coupled current equations for electrons and holes,

$$\begin{aligned}
 -e \frac{\partial n}{\partial t} + \nabla \cdot \mathbf{j}_n &= -e(G - R), \\
 e \frac{\partial p}{\partial t} + \nabla \cdot \mathbf{j}_p &= e(G - R),
 \end{aligned}$$



where  $G$  and  $R$  are the (position-dependent) generation and recombination rates for electron-hole pairs. Here the charge current density  $\mathbf{j}_{n,p}$  has a dimension of (charge)(area)<sup>-1</sup>(time)<sup>-1</sup> and the generation rate has (volume)<sup>-1</sup>(time)<sup>-1</sup>. The recombination rate is the sum of three different processes  $R = R_{\text{rad}} + R_{\text{Auger}} + R_{\text{SRH}}$ . See our “Laser diode tutorial” for nextnano++, [Nelson] or other literature for details.

By solving this current equation and the Poisson equation self-consistently, the program obtains the current density at each bias step. The resulting I-V curve is shown in Figure 3.20.7.7 and Figure 3.20.7.8. For comparison, the dark current has been simulated by setting

```
# nextnano++
structure{
  region{
    generation{
      constant{ rate = 0.0 }
    }
  }
}
```

```
! nextnano3
$optical-absorption
...
import-solar-spectrum = yes
number-of-suns        = 0.0
$end_optical-absorption
```

The dark current in the present device behaves like in a diode under forward bias. When the sun illuminates the device, electrons and holes are created and current flows in the reverse direction.

If you change the device geometry or materials and the I-V curve is no longer reasonable, it is likely that the numerical calculation did not converge. Please check the .log file. For the convergence of the current-Poisson equation, you might need to change the settings under *\$numeric-control* using nextnano<sup>3</sup>.

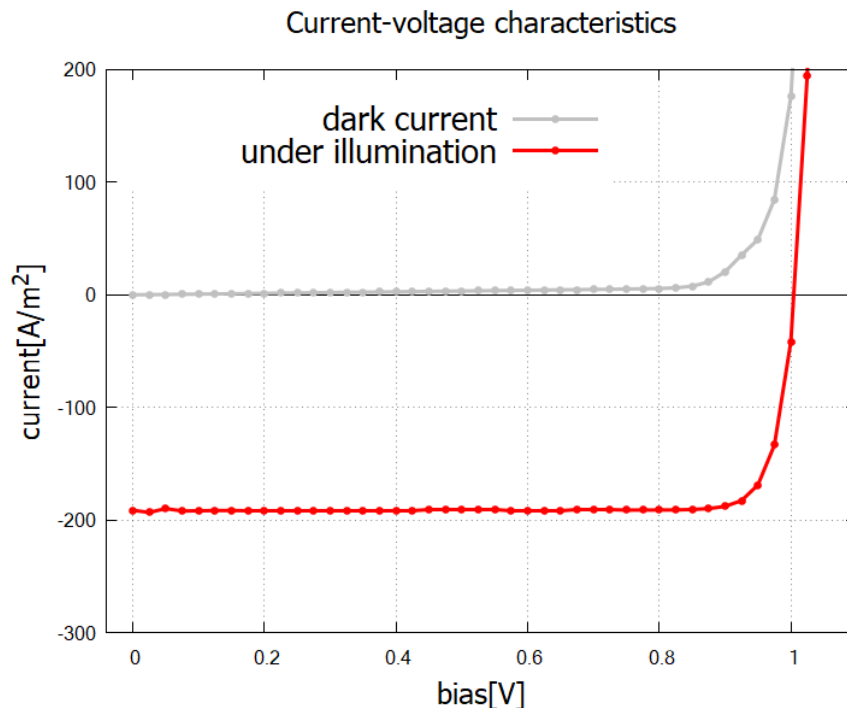


Figure 3.20.7.7: I-V characteristics of the solar cell *IV\_characteristics.dat* (nextnano++). In the bias regime 0-1 V the system works as a solar cell.

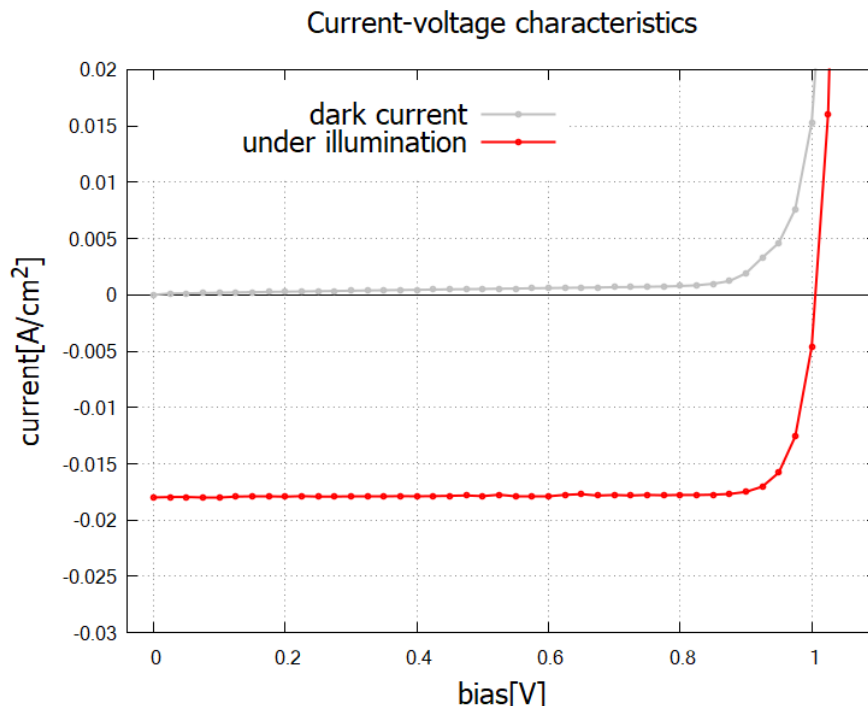


Figure 3.20.7.8: I-V characteristics of the solar cell *currentIV\_characteristics\_new.dat* (nextnano<sup>3</sup>). In the bias regime 0-1 V the system works as a solar cell.

## 5. Solar efficiency

From the I-V curve the solar cell power density  $P_{\text{out}} = -IV$  and the efficiency  $\eta = \frac{P_{\text{out}}}{P_{\text{sun}}}$  are calculated. For the present device under 1 sun, the maximum efficiency of **15.8%** (nextnano++) / **17.0%** (nextnano3) is achieved at the bias 0.9 V (Figure 3.20.7.9, Figure 3.20.7.10 red). The theoretical limit for GaAs (band gap 1.42 eV at  $T = 300\text{ K}$ ) is around 30% under the AM1.5 condition without concentration [Sze].

The maximum efficiency of the present device increases to **21.6%** (nextnano++) / **22.3%** (nextnano3) for **100-sun concentration**, mainly due to the increase in open circuit voltage (Figure 3.20.7.9, Figure 3.20.7.10 blue). This means one cell operating under 100 suns can produce the same power output as  $\frac{100P_{\text{sun}} \times 0.216}{P_{\text{sun}} \times 0.158} = 137$  cells under 1 sun. Optical concentration reduces the total cost of solar cells since concentrator materials are usually less expensive than the ones for solar cells [Sze].

The .log file and the file *solar\_cell\_info.txt* (nextnano<sup>3</sup>) contain additional properties of the solar cell.

```
Solar cell results
*****
short-circuit current:      I_sc      =      184.149021 [A/m^2]   (photo current: It_
↳increases with smaller band gap.)
open-circuit voltage:      U_oc      =      -1.012500 [V]       (U_oc <= built-in_
↳potential ~ band gap)
current at maximum power:  I_max     =      180.613633 [A/m^2]
voltage at maximum power:  U_max     =      -0.900000 [V]
maximum power output:      P_max     = U_max * I_max =      -162.552270 [W/m^2]_
↳(condition for maximum power output: dP/dV = 0)
maximum extracted power:   P_solar   = - P_max      =      162.552270 [W/m^2]
incident power:            P_in      =      1000.369631 [W/m^2]
ideal conversion efficiency: eta     = P_max / P_in   =      16.249221 %
fill factor:               FF       =      0.871824
In practice, a good fill factor is around 0.8.
All these results are approximations.
```

(continues on next page)

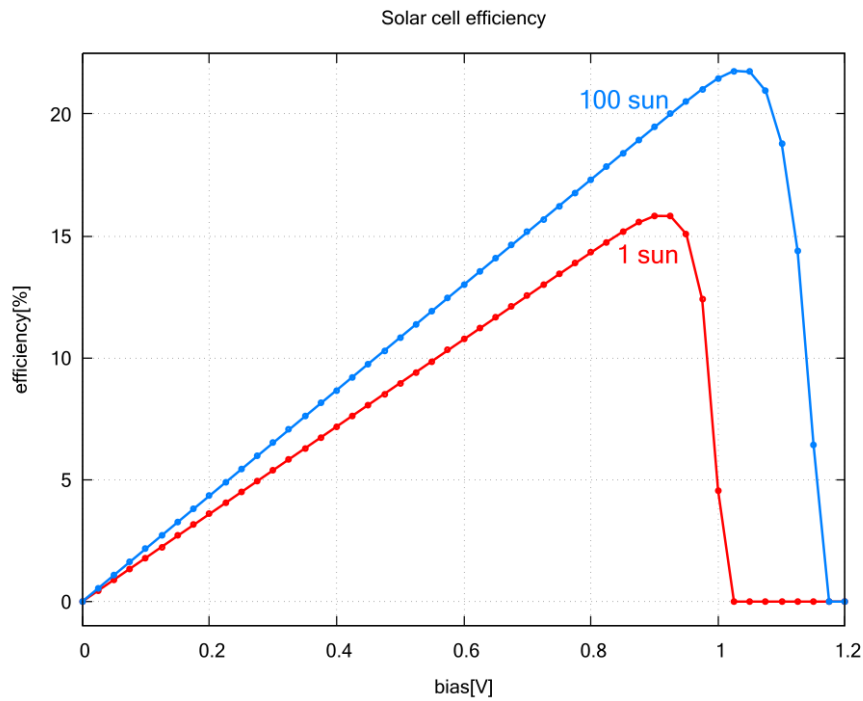


Figure 3.20.7.9: Solar cell efficiency  $\eta$  for no sunlight concentration (red) and 100-sun concentration (blue) by *nextnano++*. The data is contained in *solar\_cell\_efficiency.dat*.

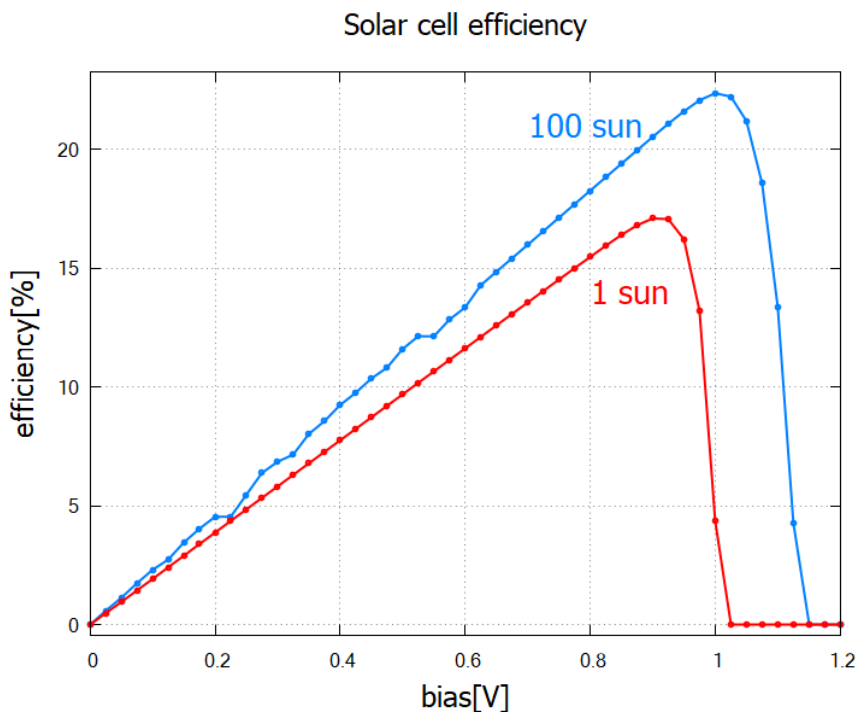


Figure 3.20.7.10: Solar cell efficiency  $\eta$  for no sunlight concentration (red) and 100-sun concentration (blue) by *nextnano<sup>3</sup>*. The data is contained in *current/solar\_cell\_efficiency.dat*.

(continued from previous page)

They are only correct if a lot of voltage steps have been used (i.e. a high  $\rightarrow$  resolution of bias steps).

*The convergence of the simulation is sensitive to the device settings such as the number of suns. If the convergence fails in your original device, please consider changing the settings in `$numeric-control`.*

**Input Files for nextnano<sup>3</sup>:**

- `1DGaAs_SolarCell_nn3.in`
- `1DGaAs_SolarCell_nn3_import_generation.in`

---

*Last update: nnnn/nn/nn*

### 3.20.8 Intersubband absorption of an infinite quantum well

**Input files for nextnano<sup>3</sup>:**

- `1D_IntersubbandAbsorption_InfiniteWell_GaAs_Chuang_sg_nn3.in`
- `1D_IntersubbandAbsorption_InfiniteWell_GaAs_Chuang_kp_nn3.in`

**Input files for nextnano++:**

- `1D_IntersubbandAbsorption_InfiniteWell_GaAs_Chuang_sg_nnp.in`
- `1D_IntersubbandAbsorption_InfiniteWell_GaAs_Chuang_Gamma_nnp.in`
- `1D_IntersubbandAbsorption_InfiniteWell_GaAs_Chuang_kp_nnp.in`

This tutorial presents calculation of intersubband absorption spectrum of a GaAs quantum well with infinite barriers.

Input files for both the nextnano++ and nextnano<sup>3</sup> tools are available.

The following input file was used:

- `1D_IntersubbandAbsorption_InfiniteWell_GaAs_Chuang_sg_nn3.in` (single-band effective mass approximation)

This tutorial aims to reproduce the example discussed on p. 376f of Section 9.6.2 *Intersubband Absorption Spectrum* of [ChuangOpto1995].

## Structure

| Property                      | Symbol         | unit             | <a href="#">[ChuangOpto1995]</a> | nextnano             |
|-------------------------------|----------------|------------------|----------------------------------|----------------------|
| quantum well width            | L              | nm               | 10.0                             | 10.0                 |
| barrier height                | E <sub>b</sub> | eV               | infinite quantum well model      | 1000                 |
| effective electron mass       | m <sub>e</sub> | m <sub>0</sub>   | 0.0665                           | 0.0665               |
| refractive index              | n <sub>r</sub> |                  | 3.3                              | 3.3                  |
| doping concentration (n-type) | N <sub>D</sub> | cm <sup>-3</sup> | 1 · 10 <sup>18</sup>             | 1 · 10 <sup>18</sup> |
| linewidth (FWHM)              | Γ              | meV              | 30                               | 30                   |
| temperature                   | T              | K                | 300                              | 300                  |

[\[ChuangOpto1995\]](#) models the infinite quantum well using the analytical solution while we are using a numerical model with a barrier height of 1000 eV.

## Results

[\[ChuangOpto1995\]](#) uses the analytical infinite quantum well model and calculates the energy levels, and the inter-subband dipole moment exactly. Our calculated transition energies differ by 3 meV which is acceptable as we use a finite grid spacing of 0.05 nm. Our calculated dipole moment is also reasonable. More difficult are the densities. In our calculation we solve the Schrödinger-Poisson equation self-consistently. For that reason, the quantum well bottom is not entirely flat but slightly bent. At T = 300 K, the second subband shows a small density which is larger than in the model of [\[ChuangOpto1995\]](#). The difference in subband densities leads to a slight deviation for the peak of the absorption spectrum because the occupation of the second level N<sub>2</sub> reduces absorption. Nevertheless, the agreement is reasonable.

| Property                        | Symbol            | unit             | <a href="#">[ChuangOpto1995]</a> | nextnano                |
|---------------------------------|-------------------|------------------|----------------------------------|-------------------------|
| energy level                    | E <sub>1</sub>    | meV              | 56.5 (exact)                     |                         |
| energy level                    | E <sub>2</sub>    | meV              | 226 (exact)                      |                         |
| transition energy               | E <sub>21</sub>   | meV              | 169.5 (exact)                    | 166.5                   |
| dipole moment                   | x <sub>21</sub>   | nm               | -1.8 (exact)                     | -1.82                   |
| E <sub>F</sub> - E <sub>1</sub> |                   | eV               | 78                               | 28.2                    |
| subband density                 | N <sub>1</sub>    | cm <sup>-2</sup> | 7.19 · 10 <sup>11</sup>          | 9.92 · 10 <sup>11</sup> |
| subband density                 | N <sub>2</sub>    | cm <sup>-2</sup> |                                  | 3 · 10 <sup>9</sup>     |
| peak in absorption              | α <sub>peak</sub> | cm <sup>-1</sup> | 1.015 · 10 <sup>4</sup>          | 0.986 · 10 <sup>4</sup> |

The following figures show the

- lowest eigenstates (probability densities) of the infinite quantum well
- absorption spectra  $\alpha(\omega)$  in units of cm<sup>-1</sup>
- position dependent absorption spectra  $\alpha(\omega, x)$  in units of cm<sup>-1</sup>

The peak in the absorption spectra occurs at the transition energy E<sub>21</sub>.

Then we perform two parameter sweeps:

- We vary the **quantum well width** (Variable: \$QuantumWellWidth).
- We vary the **doping concentration** (Variable: \$DopingConcentration).

Results and explanations for the sweeps can be found further below.

---

— Begin —

**Automatic documentation: Running simulations, generating figures and reStructured Text (\*.rst) using nextnanopy**

The following documentation and figures were generated automatically using *nextnanopy*.

The following Python script was used: `intersubband_InfiniteQW_nextnano3.py`

The following figures have been generated using *nextnano<sup>3</sup>*. Self-consistent Schrödinger-Poisson calculations have been performed for an infinite quantum well.

A single-band effective mass approach has been used, i.e. not  $\mathbf{k} \cdot \mathbf{p}$ .

The absorption spectra have been calculated assuming a parabolic energy dispersion  $E(k)$ .

**Infinite Quantum Well** (QuantumWellWidth = 10 nm)

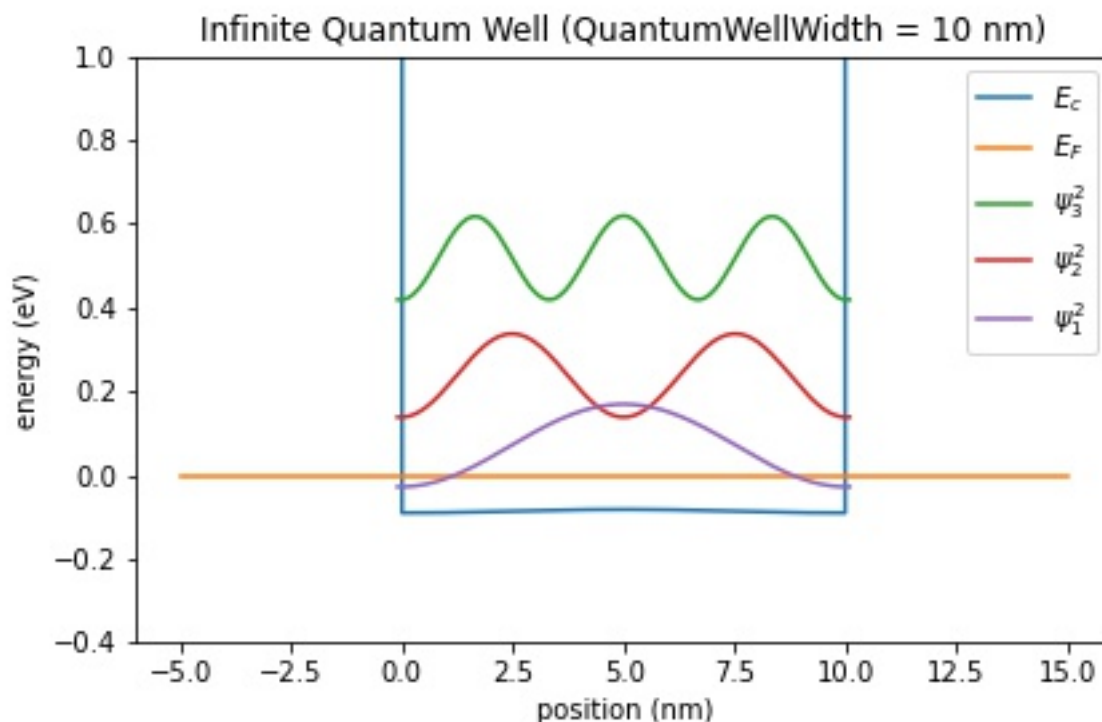


Figure 3.20.8.1: Conduction band edge, Fermi level and confined electron states of an infinite quantum well (QuantumWellWidth = 10 nm)

**Infinite Quantum Well** (QuantumWellWidth = 13 nm)

**Infinite Quantum Well** (QuantumWellWidth = 16 nm)

**Infinite Quantum Well** (QuantumWellWidth = 19 nm)

**Parameter sweep: Well width**

Figure 3.20.8.10 shows the absorption spectra for different **quantum well widths** (Variable: `$QuantumWellWidth`). The larger the well, the closer the energy level spacings. Therefore the peak occurs at smaller energies. The larger wells show absorption also for transitions other than  $E_{21}$ .

**Parameter sweep: Doping concentration**

Figure 3.20.8.11 shows the absorption spectra for different **doping concentrations** (Variable: `$DopingConcentration`). The peak absorption coefficient increases with the doping concentration  $N_D$ .

Last update: nnnn/nn/nn

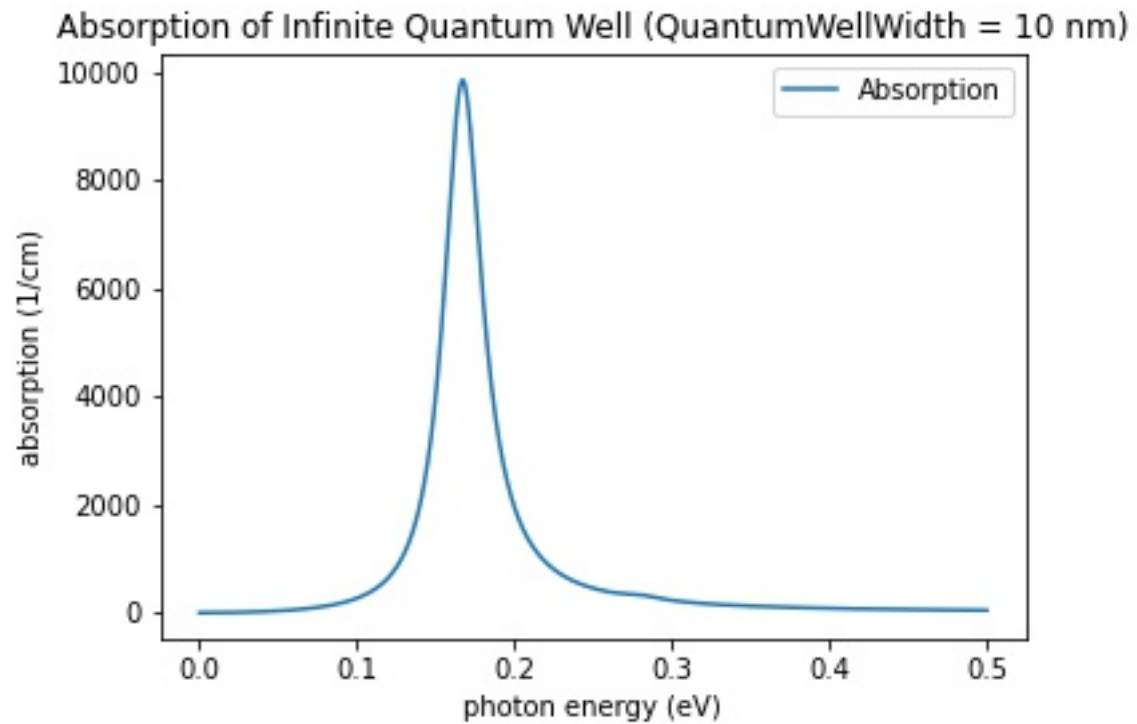


Figure 3.20.8.2: Calculated absorption spectra  $\alpha(E)$  of an infinite quantum well (QuantumWellWidth = 10 nm)

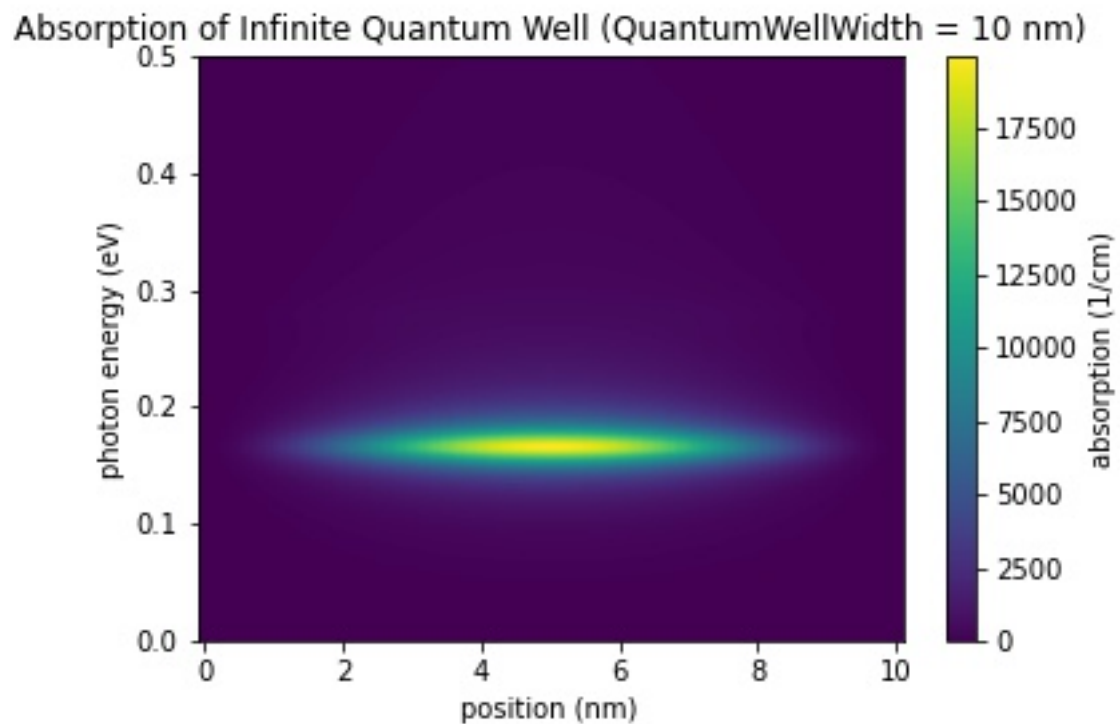


Figure 3.20.8.3: Calculated spatially resolved absorption spectrum  $\alpha(x, E)$  of an infinite quantum well (QuantumWellWidth = 10 nm)

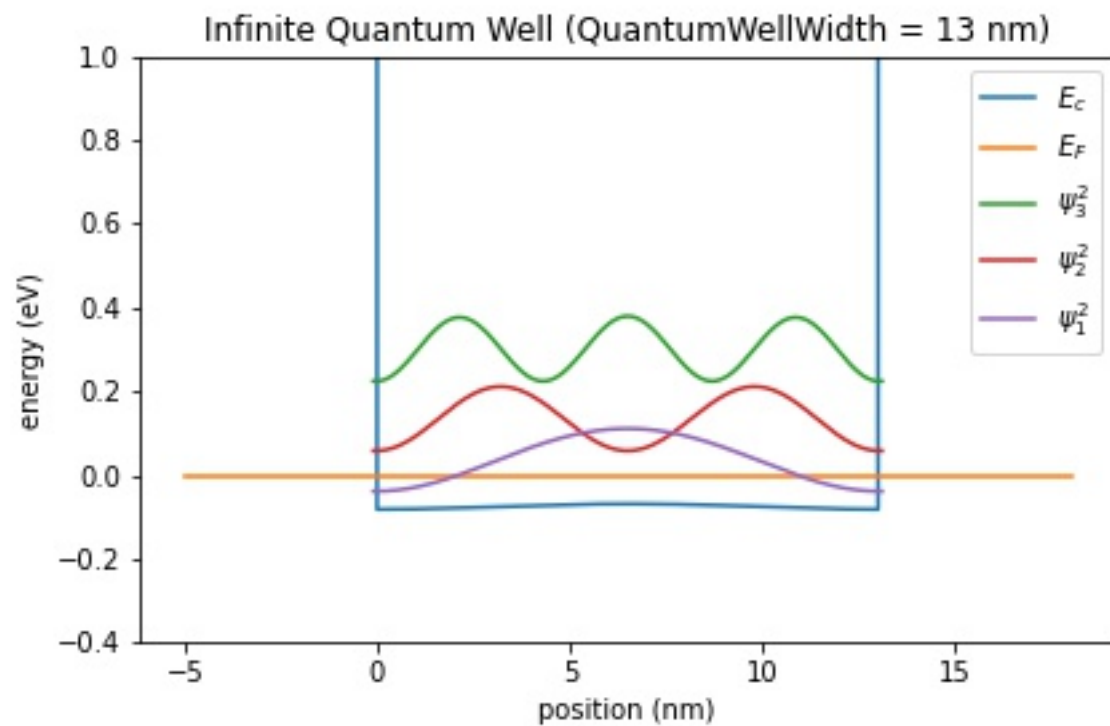


Figure 3.20.8.4: Conduction band edge, Fermi level and confined electron states of an infinite quantum well (QuantumWellWidth = 13 nm)

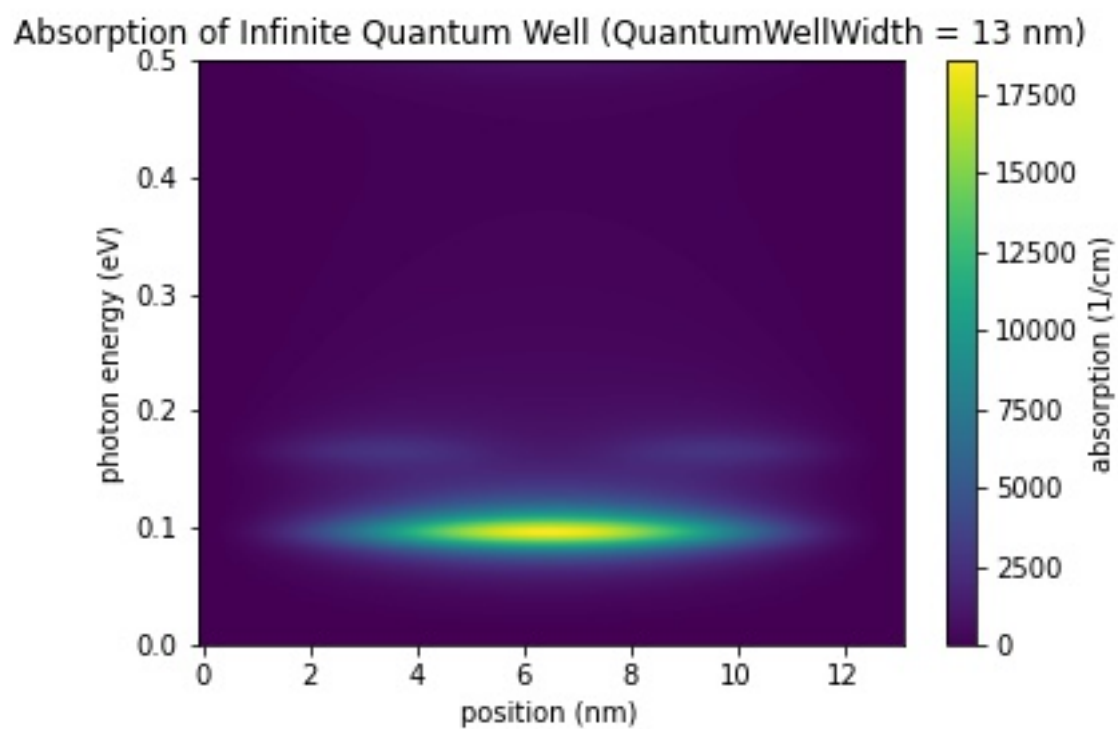


Figure 3.20.8.5: Calculated spatially resolved absorption spectrum  $\alpha(x, E)$  of an infinite quantum well (QuantumWellWidth = 13 nm)



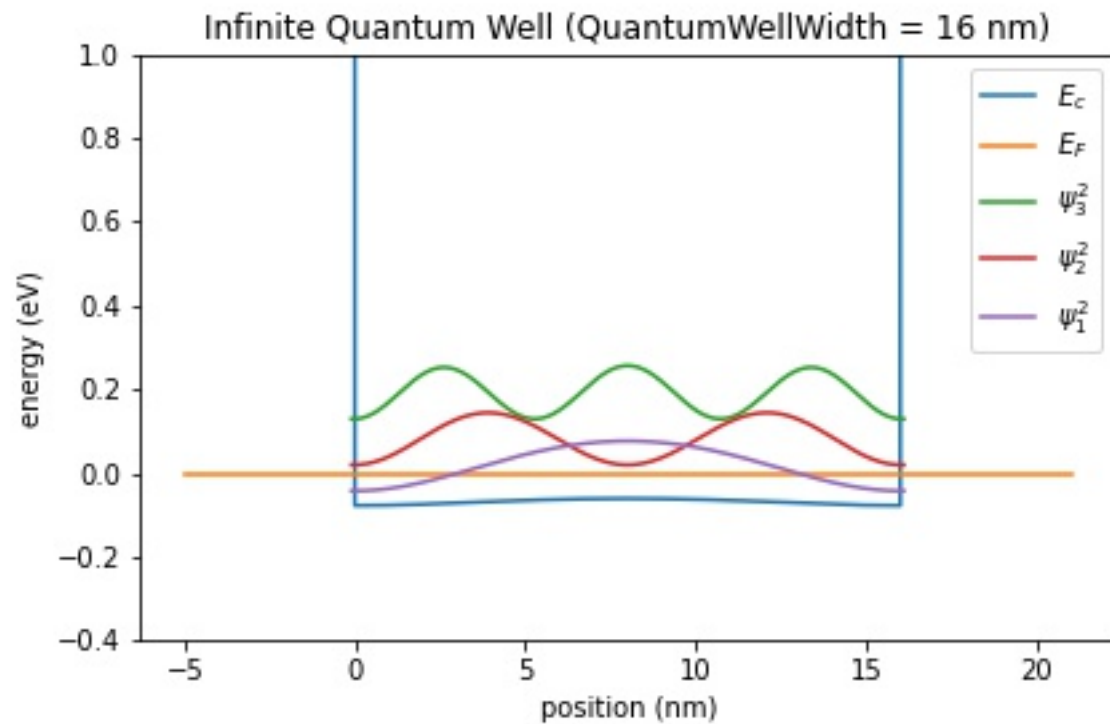


Figure 3.20.8.6: Conduction band edge, Fermi level and confined electron states of an infinite quantum well (QuantumWellWidth = 16 nm)

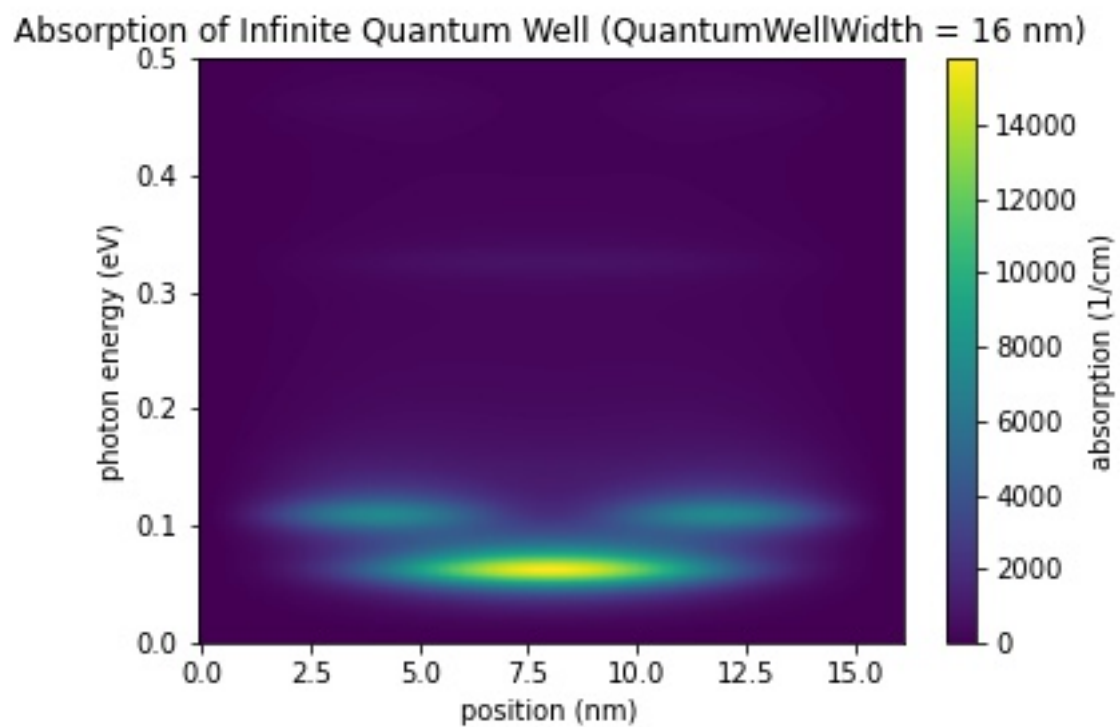


Figure 3.20.8.7: Calculated spatially resolved absorption spectrum  $\alpha(x, E)$  of an infinite quantum well (QuantumWellWidth = 16 nm)

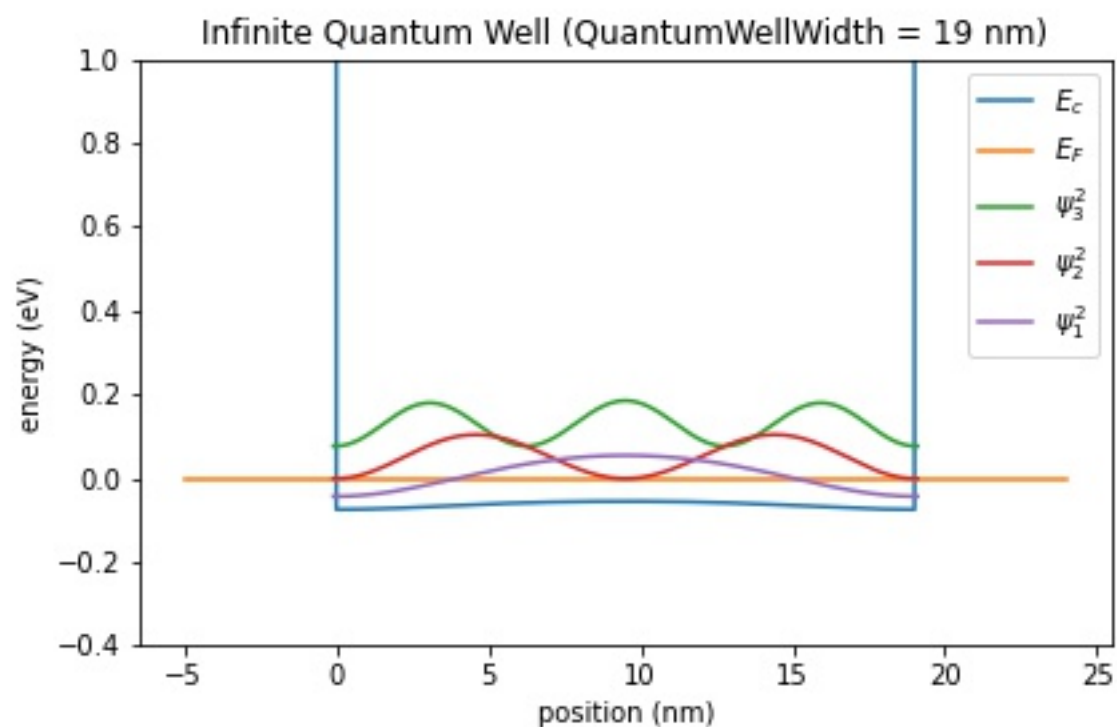


Figure 3.20.8.8: Conduction band edge, Fermi level and confined electron states of an infinite quantum well (QuantumWellWidth = 19 nm)

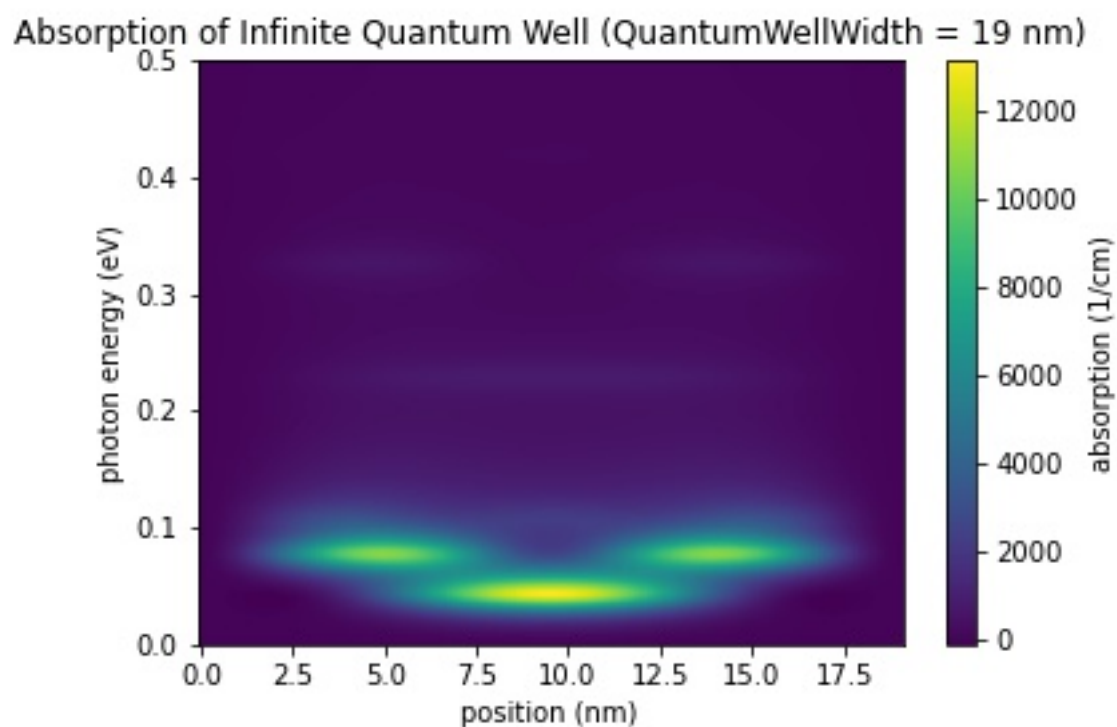


Figure 3.20.8.9: Calculated spatially resolved absorption spectrum  $\alpha(x, E)$  of an infinite quantum well (QuantumWellWidth = 19 nm)

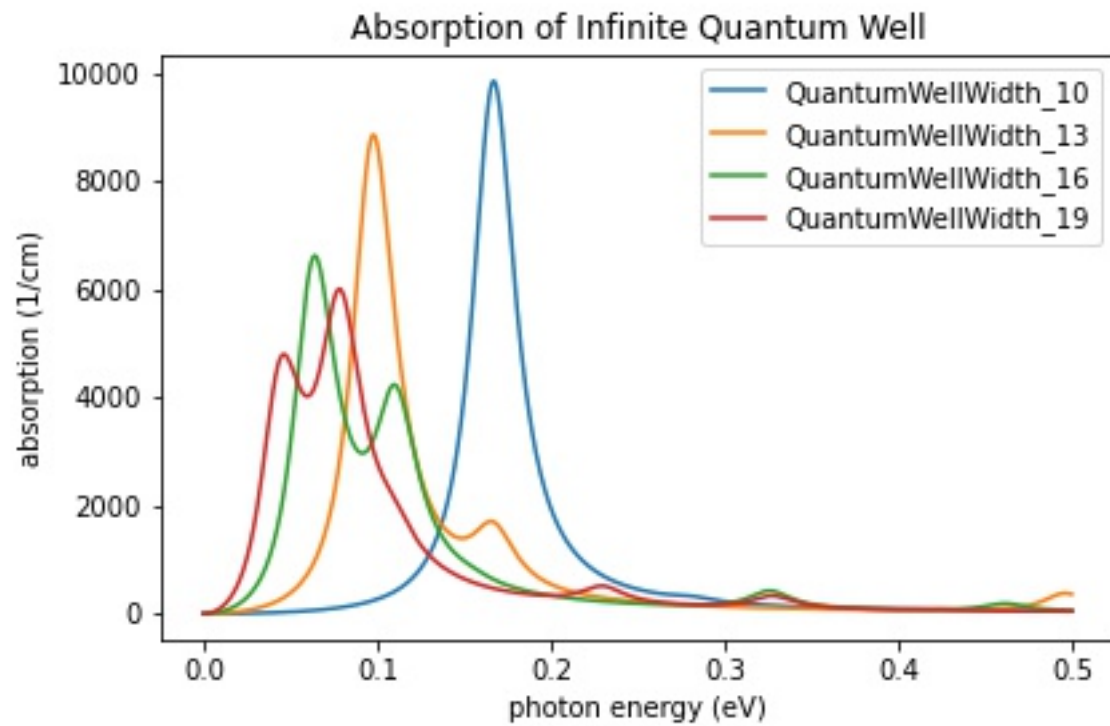


Figure 3.20.8.10: Calculated absorption spectra  $\alpha(E)$  of an infinite quantum well for different well widths

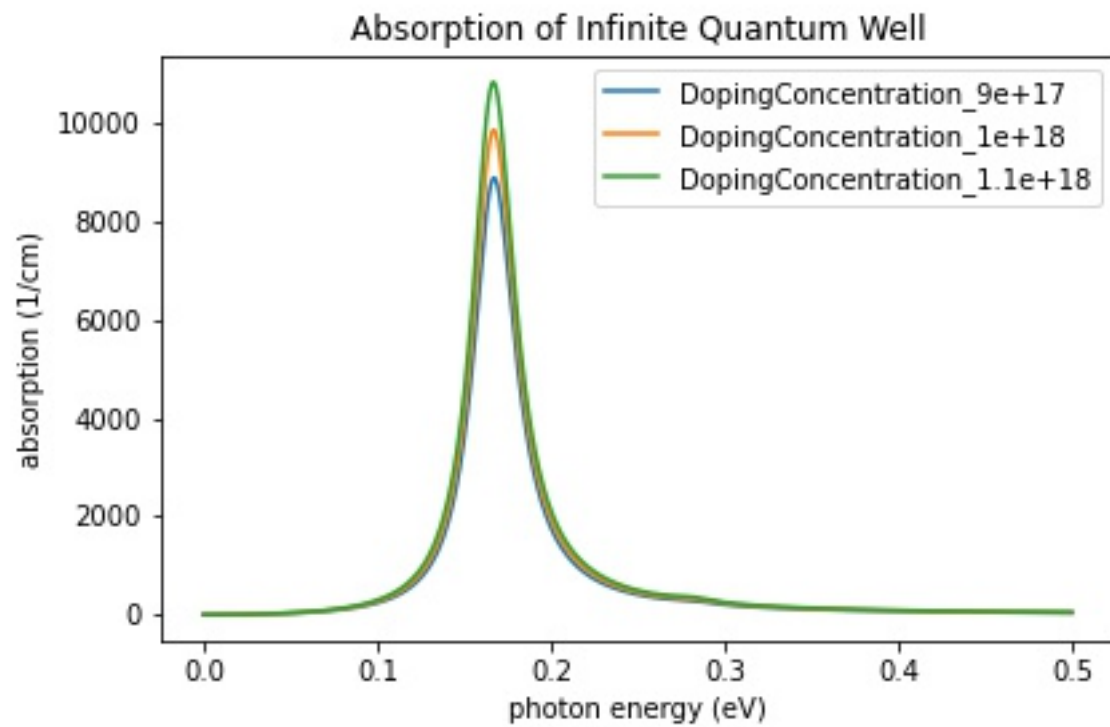


Figure 3.20.8.11: Calculated absorption spectra  $\alpha(E)$  of an infinite quantum well for different doping concentrations

### 3.20.9 Intersubband transitions in InGaAs/AlInAs multiple quantum well systems

This tutorial calculates the eigenstates of a single, double and triple quantum wells. It compares the energy levels and wave functions of the single-band effective mass approximation with the 8-band  $\mathbf{k} \cdot \mathbf{p}$  model. Finally, the intersubband absorption spectrum is calculated.

Input files for both the *nextnano++* and *nextnano<sup>3</sup>* tools are available.

The following input files were used:

- Single Quantum Well
  - 1DSirtoriPRB1994\_OneWell\_sg\_self-consistent\_nn\*.in (single-band effective mass approximation)
  - 1DSirtoriPRB1994\_OneWell\_kp\_self-consistent\_nn\*.in (8-band  $\mathbf{k} \cdot \mathbf{p}$ )
  - 1DSirtoriPRB1994\_OneWell\_sg\_quantum-only\_nn\*.in (single-band effective mass approximation)
  - 1DSirtoriPRB1994\_OneWell\_kp\_quantum-only\_nn\*.in (8-band  $\mathbf{k} \cdot \mathbf{p}$ )
- Two coupled Quantum Wells
  - 1DSirtoriPRB1994\_TwoCoupledWells\_sg\_self-consistent\_nn\*.in (single-band effective mass approximation)
  - 1DSirtoriPRB1994\_TwoCoupledWells\_kp\_self-consistent\_nn\*.in (8-band  $\mathbf{k} \cdot \mathbf{p}$ )
  - 1DSirtoriPRB1994\_TwoCoupledWells\_sg\_quantum-only\_nn\*.in (single-band effective mass approximation)
  - 1DSirtoriPRB1994\_TwoCoupledWells\_kp\_quantum-only\_nn\*.in (8-band  $\mathbf{k} \cdot \mathbf{p}$ )
- Three coupled Quantum Wells
  - 1DSirtoriPRB1994\_ThreeCoupledWells\_sg\_self-consistent\_nn\*.in (single-band effective mass approximation)
  - 1DSirtoriPRB1994\_ThreeCoupledWells\_kp\_self-consistent\_nn\*.in (8-band  $\mathbf{k} \cdot \mathbf{p}$ )
  - 1DSirtoriPRB1994\_ThreeCoupledWells\_sg\_quantum-only\_nn\*.in (single-band effective mass approximation)
  - 1DSirtoriPRB1994\_ThreeCoupledWells\_kp\_quantum-only\_nn\*.in (8-band  $\mathbf{k} \cdot \mathbf{p}$ )

This tutorial aims to reproduce Figs. 4 and 5 of [SirtoriPRB1994].

This tutorial nicely demonstrates that for the ground state energy the single-band effective mass approximation is sufficient whereas for the higher lying states a nonparabolic model, like the 8-band  $\mathbf{k} \cdot \mathbf{p}$  approximation, is necessary. This is important for e.g. quantum cascade lasers where higher lying states have a dominant role.

#### Layer sequence

We investigate three structures:

- a) a single quantum well
- b) two coupled quantum wells
- c) three coupled quantum wells

#### Material parameters

We use  $\text{In}_{0.53}\text{Ga}_{0.47}\text{As}$  as the quantum well material and  $\text{Al}_{0.48}\text{In}_{0.52}\text{As}$  as the barrier material. Both materials are lattice matched to the substrate material InP. Thus we assume that the InGaAs and AlInAs layers are unstrained with respect to the InP substrate. The publication [SirtoriPRB1994] lists the following material parameters:

|                                |                                                                                     |                      |
|--------------------------------|-------------------------------------------------------------------------------------|----------------------|
| conduction band offset         | Al <sub>0.48</sub> In <sub>0.52</sub> As / In <sub>0.53</sub> Ga <sub>0.47</sub> As | 0.510 eV             |
| conduction band effective mass | Al <sub>0.48</sub> In <sub>0.52</sub> As                                            | 0.072 m <sub>0</sub> |
| conduction band effective mass | In <sub>0.53</sub> Ga <sub>0.47</sub> As                                            | 0.043 m <sub>0</sub> |

The temperature is set to 10 Kelvin.

## Method

### Single-band effective mass approximation

Because our structure is doped, we have to solve the single-band Schrödinger-Poisson equation self-consistently. The doping is such that the electron ground state is below the Fermi level and all other states are far away from the Fermi level, i.e. only the ground state is occupied and contributes to the charge density.

For *nextnano++* we use:

```
# '0' solve Schrödinger equation only
# '1' solve Schrödinger and Poisson equations self-consistently
$SELF_CONSISTENT = 1

run{
  !IF($SELF_CONSISTENT)
    poisson{ }
    quantum_poisson{ iterations = 50 } # Schrödinger-Poisson
  !ELSE
    quantum{ } # Schrödinger only
  !ENDIF
}

quantum {
  region{
    ...
    Gamma{ # single-band
      num_ev = 3 # 3 eigenstates
    }
  }
}
```

For *nextnano<sup>3</sup>* we use:

```
# 'QUANTUM_ONLY': solve Schrödinger and Poisson equations self-
→consistently
# 'SELF_CONSISTENT': solve Schrödinger equation only
%QUANTUM_ONLY = .FALSE.
%SELF_CONSISTENT = .TRUE.

$simulation-flow-control
!IF %QUANTUM_ONLY flow-scheme = 3 # Schrödinger only
!IF %SELF_CONSISTENT flow-scheme = 2 # Schrödinger-Poisson

!IF %QUANTUM_ONLY raw-potential-in = yes
!IF %SELF_CONSISTENT raw-potential-in = no

$quantum-model-electrons
...
model-name = effective-mass # single-band

number-of-eigenvalues-per-band = 3 # 3 eigenstates
```

**Note**

Single-band eigenstates are two-fold spin degenerate.

The Fermi level is always equal to 0 eV in our simulations and the band profile is shifted accordingly to meet this requirement.

**8-band  $\mathbf{k} \cdot \mathbf{p}$  approximation**

*Old version of this tutorial:*

Because both, the single-band and the 8-band  $\mathbf{k} \cdot \mathbf{p}$  ground state energy and the corresponding wave functions are almost identical, we can read in the self-consistently calculated electrostatic potential of the single-band approximation and calculate for this potential the 8-band  $\mathbf{k} \cdot \mathbf{p}$  eigenstates and wave functions for  $k_{\parallel} = 0$ .

For *nextnano*<sup>3</sup> we use:

```
$simulation-flow-control
!IF %QUANTUM_ONLY    flow-scheme = 3    # Schrödinger only

!IF %QUANTUM_ONLY    raw-directory-in = raw_data/
!IF %QUANTUM_ONLY    raw-potential-in = yes

$quantum-model-electrons
...
model-name = 8x8kp                # 8-band  $\mathbf{k} \cdot \mathbf{p}$ 

number-of-eigenvalues-per-band = 6    # 6 eigenstates
```

**Note**

One  $\mathbf{k} \cdot \mathbf{p}$  eigenstate for each spin component.

**New version of this tutorial:**

We provide input files for:

- self-consistent single-band Schrödinger equation (because the structure is doped)
- single-band Schrödinger equation (without self-consistency)
- 8-band  $\mathbf{k} \cdot \mathbf{p}$  single-band Schrödinger equation (without self-consistency)

For a), although the structure is doped, the band bending is very small. Thus we omit for the single-band /  $\mathbf{k} \cdot \mathbf{p}$  comparison in b) and c) the self-consistent cycle.

**Results****Single quantum well**

Figure 3.20.9.1 shows the lowest two electron eigenstates for an  $\text{In}_{0.53}\text{Ga}_{0.47}\text{As} / \text{Al}_{0.48}\text{In}_{0.52}\text{As}$  quantum well structure calculated with single-band effective mass approximation and with a nonparabolic 8-band  $\mathbf{k} \cdot \mathbf{p}$  model.

The energies (and square of the wave functions  $\psi^2$ ) for the ground state are identical in both models but the second eigenstate differs substantially. Clearly the single-band model leads to an energy which is far too high for the upper state.

Our calculated value for the intersubband transition energy  $E_{12}$  of 255 meV compares well with both, the calculated value of [SirtoriPRB1994] (258 meV) and their measured value (compare with absorption spectrum in Fig. 4 of [SirtoriPRB1994]).

The calculated intersubband dipole moments are:

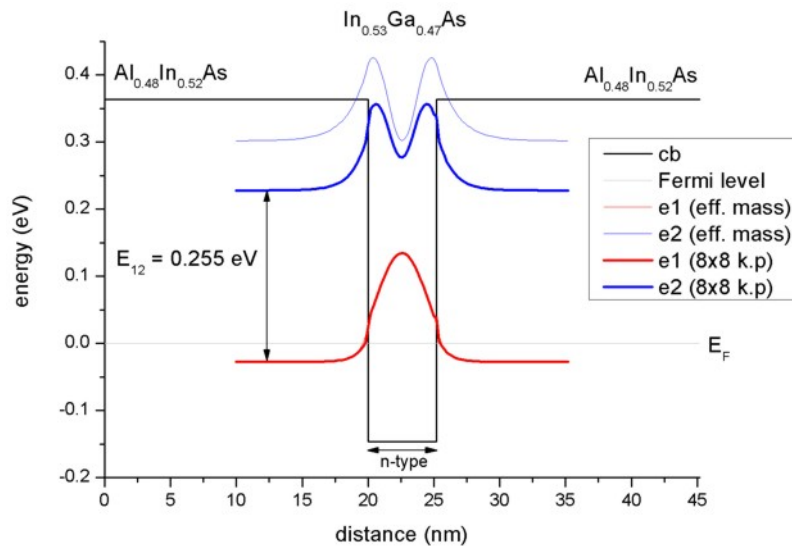


Figure 3.20.9.1: Conduction band edge, Fermi level and confined electron states of a quantum well

- $z_{12} = 1.55$  nm (single-band)

For comparison:  $z_{12} = 1.53$  nm (exp.),  $z_{12} = 1.48$  nm (th.) ([SirtoriPRB1994])

The influence of doping on the eigenenergies is negligible (smaller than 1 meV).

### Two coupled quantum wells

Figure 3.20.9.2 shows the lowest three electron eigenstates for an  $\text{In}_{0.53}\text{Ga}_{0.47}\text{As} / \text{Al}_{0.48}\text{In}_{0.52}\text{As}$  double quantum well structure calculated with single-band effective mass approximation and with a nonparabolic 8-band  $\mathbf{k} \cdot \mathbf{p}$  model.

The energies (and square of the wave functions  $\psi^2$ ) for the ground state are very similar in both models but the second and especially the third eigenstate differ substantially. Clearly the single-band model leads to energies which are far too high for the higher lying states.

Our calculated values for the intersubband transition energies  $E_{12} = 150$  meV and  $E_{13} = 267$  meV compare well with both, the calculated values of [SirtoriPRB1994] (150 meV and 271 meV) and their measured values (compare with absorption spectrum in Fig. 5 (a) of [SirtoriPRB1994]).

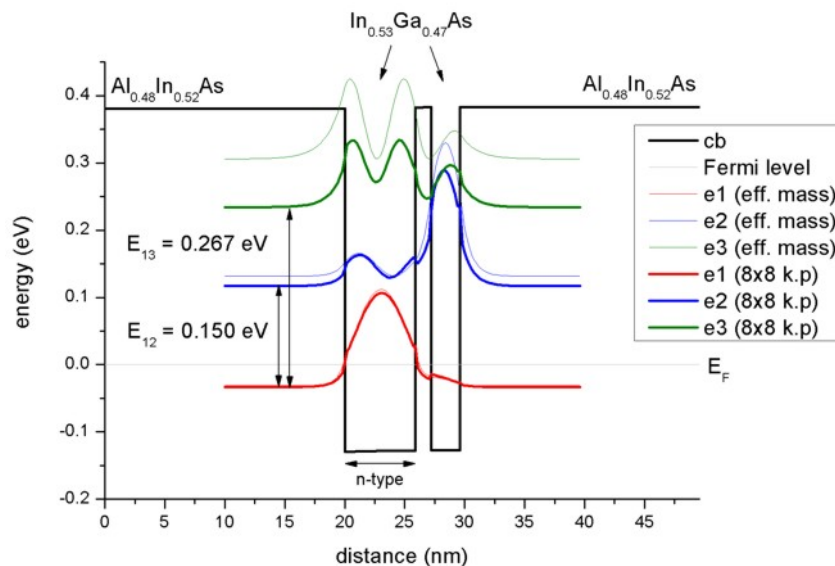


Figure 3.20.9.2: Conduction band edge, Fermi level and confined electron states of two coupled quantum wells



The calculated intersubband dipole moments are:

- $z_{12} = 1.61$  nm (single-band)
- $z_{13} = 1.11$  nm (single-band)

For comparison:  $z_{12} = 1.64$  nm (exp.),  $z_{12} = 1.65$  nm (th.) ([SirtoriPRB1994])

The influence of doping on the eigenenergies is almost negligible (between 0 and 2 meV).

### Three coupled quantum wells

Figure 3.20.9.3 shows the lowest four electron eigenstates for an  $\text{In}_{0.53}\text{Ga}_{0.47}\text{As}$  /  $\text{Al}_{0.48}\text{In}_{0.52}\text{As}$  triple quantum well structure calculated with single-band effective mass approximation and with a nonparabolic 8-band  $\mathbf{k} \cdot \mathbf{p}$  model.

The energies (and square of the wave functions  $\psi^2$ ) for the ground state are similar in both models but the second and especially the third and fourth eigenstates differ substantially. Clearly the single-band model leads to energies which are far too high for the higher lying states.

Our calculated values for the intersubband transition energies  $E_{12} = 118$  meV,  $E_{13} = 261$  and  $E_{14} = 370$  meV compare well with both, the calculated values of [SirtoriPRB1994] (116 meV, 257 meV and 368 meV) and their measured values (compare with absorption spectrum in Fig. 5 (b) of [SirtoriPRB1994]).

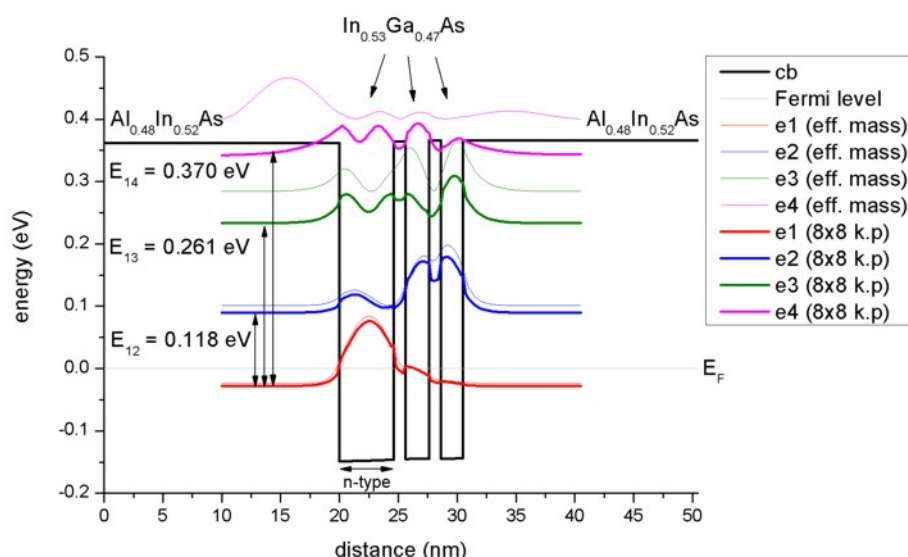


Figure 3.20.9.3: Conduction band edge, Fermi level and confined electron states of three coupled quantum wells

The calculated intersubband dipole moments are:

- $z_{12} = 1.81$  nm (single-band)
- $z_{13} = 0.77$  nm (single-band)
- $z_{14} = 0.30$  nm (single-band)

For comparison:  $z_{12} = 1.86$  nm (exp.),  $z_{12} = 1.84$  nm (th.) [SirtoriPRB1994]

The influence of doping on the eigenenergies is almost negligible (between 0 and 4 meV).

— Begin —

**Automatic documentation: Running simulations, generating figures and reStructured Text (\*.rst) using nextnanopy**

The following documentation and figures were generated automatically using *nextnanopy*.

The following Python script was used: intersubband\_MQW\_nextnano3.py



The following figures have been generated using *nextnano*<sup>3</sup>. Self-consistent Schrödinger-Poisson calculations have been performed for three different structures.

- Single Quantum Well
- Two coupled Quantum Wells
- Three coupled Quantum Wells

The single-band effective mass and the 8-band  $\mathbf{k} \cdot \mathbf{p}$  results are compared to each other. In both cases the wave functions and the quantum density are calculated self-consistently. The  $\mathbf{k} \cdot \mathbf{p}$  quantum density has been calculated taking into account the solution at different  $k_{\parallel}$  vectors.

The absorption spectrum has been calculated using a simple model assuming a parabolic energy dispersion. The dipole moment  $z_{ij} = \langle i | z | j \rangle$  has been evaluated only at  $k_{\parallel} = 0$ . The subband density is used to calculate the absorption spectrum. For the  $\mathbf{k} \cdot \mathbf{p}$  calculation, the density was calculated taking into account a nonparabolic energy dispersion, i.e. including all relevant  $k_{\parallel}$  vectors.

#### Quantum Well (single-band)

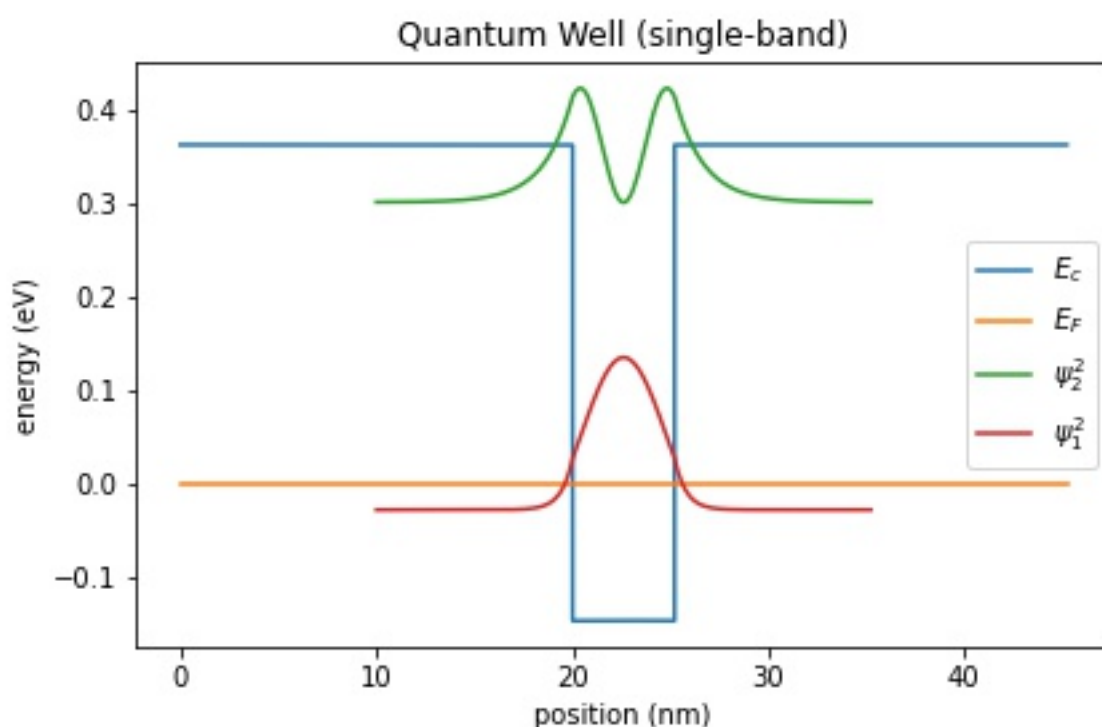


Figure 3.20.9.4: Conduction band edge, Fermi level and confined electron states of a quantum well

#### Quantum Well (k.p)

##### Two Coupled Quantum Wells (single-band)

##### Two Coupled Quantum Wells (k.p)

##### Three Coupled Quantum Wells (single-band)

##### Three Coupled Quantum Wells (k.p)

Automatic documentation: Running simulations, generating figures and reStructured Text (\*.rst) using nextnanopy

— End —

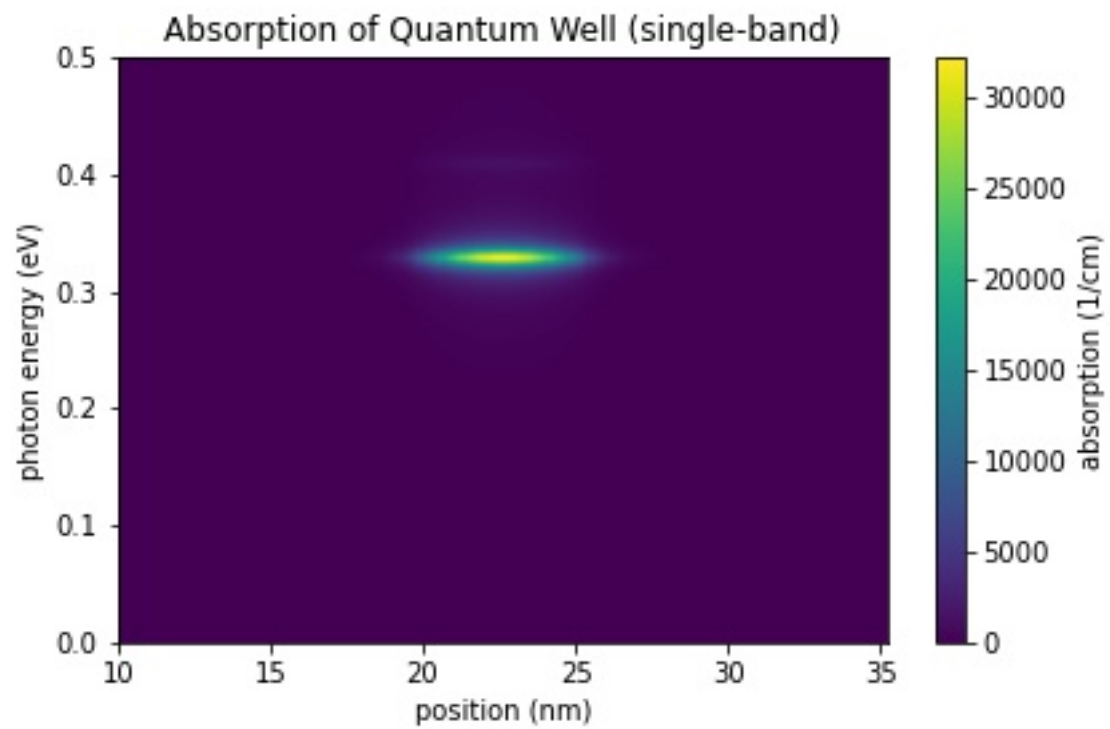


Figure 3.20.9.5: Calculated spatially resolved absorption spectrum  $\alpha(x, E)$  of a quantum well

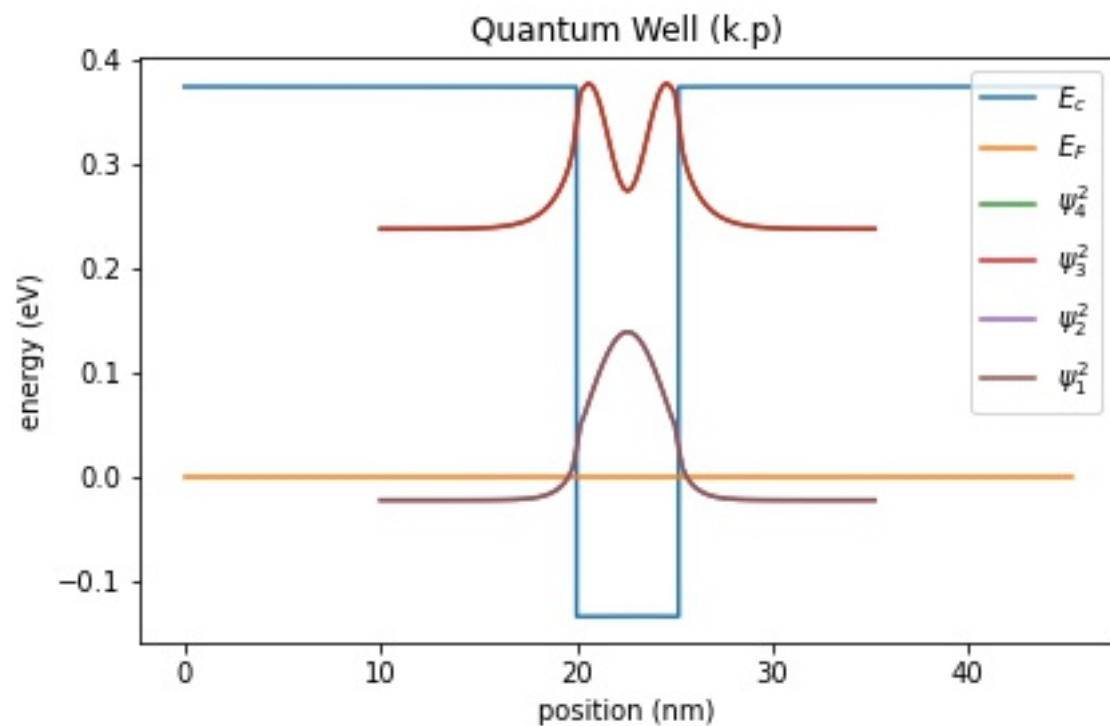


Figure 3.20.9.6: Conduction band edge, Fermi level and confined electron states of a quantum well

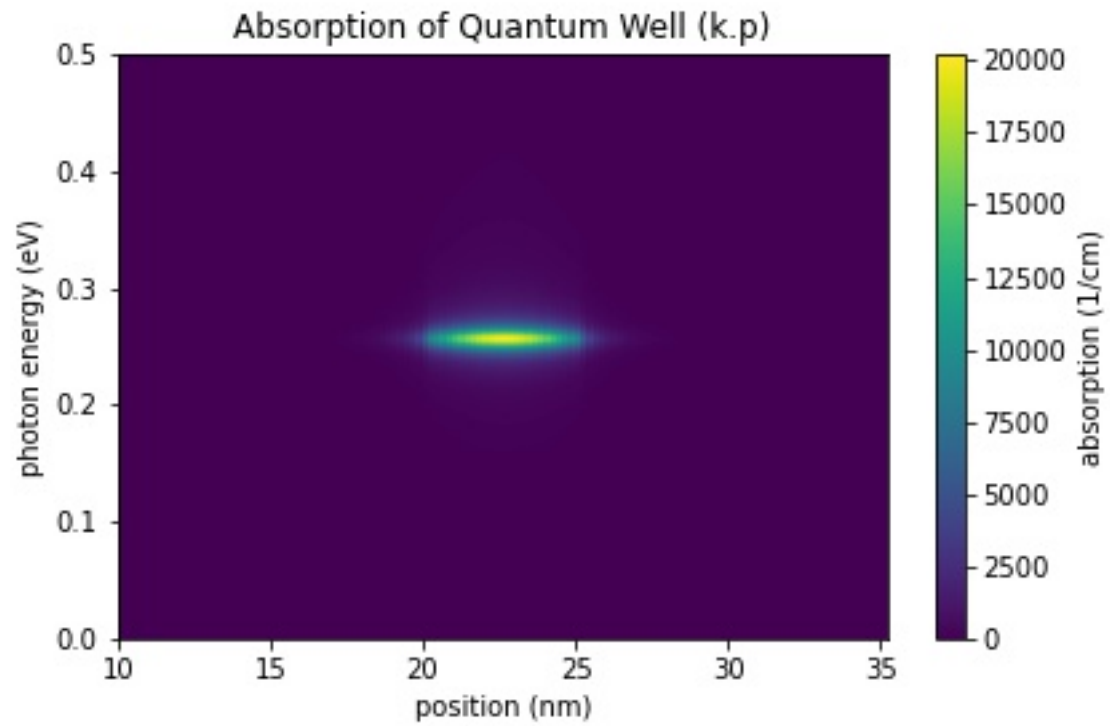


Figure 3.20.9.7: Calculated spatially resolved absorption spectrum  $\alpha(x, E)$  of a quantum well

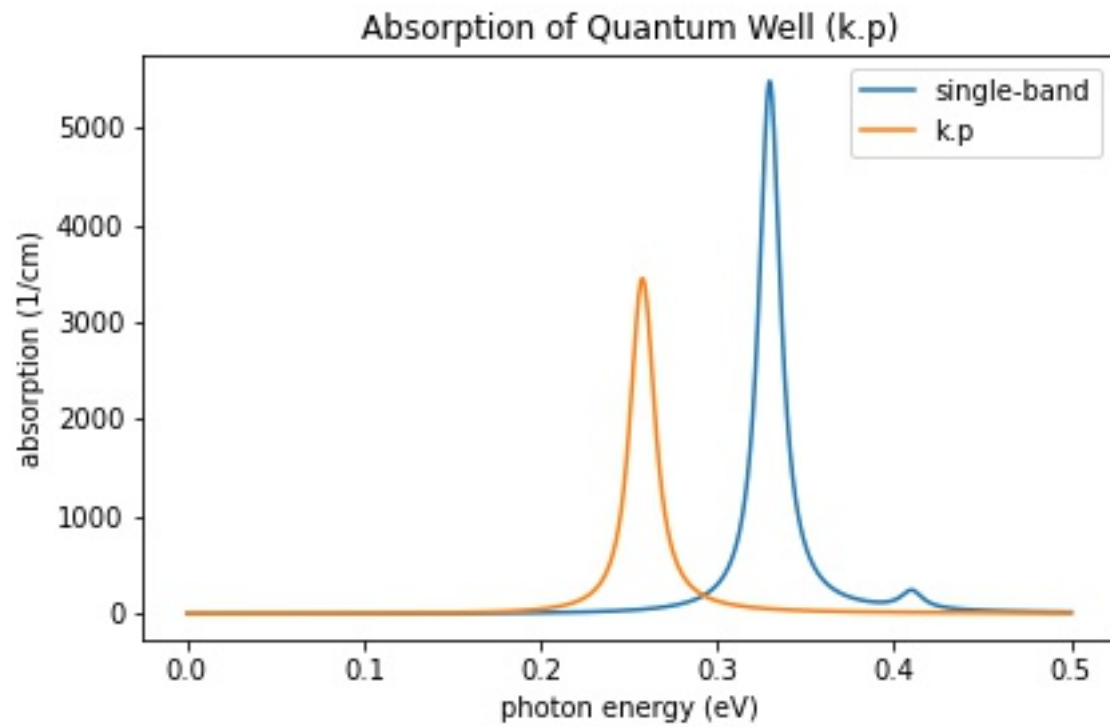


Figure 3.20.9.8: Calculated absorption spectra  $\alpha(E)$  of a quantum well

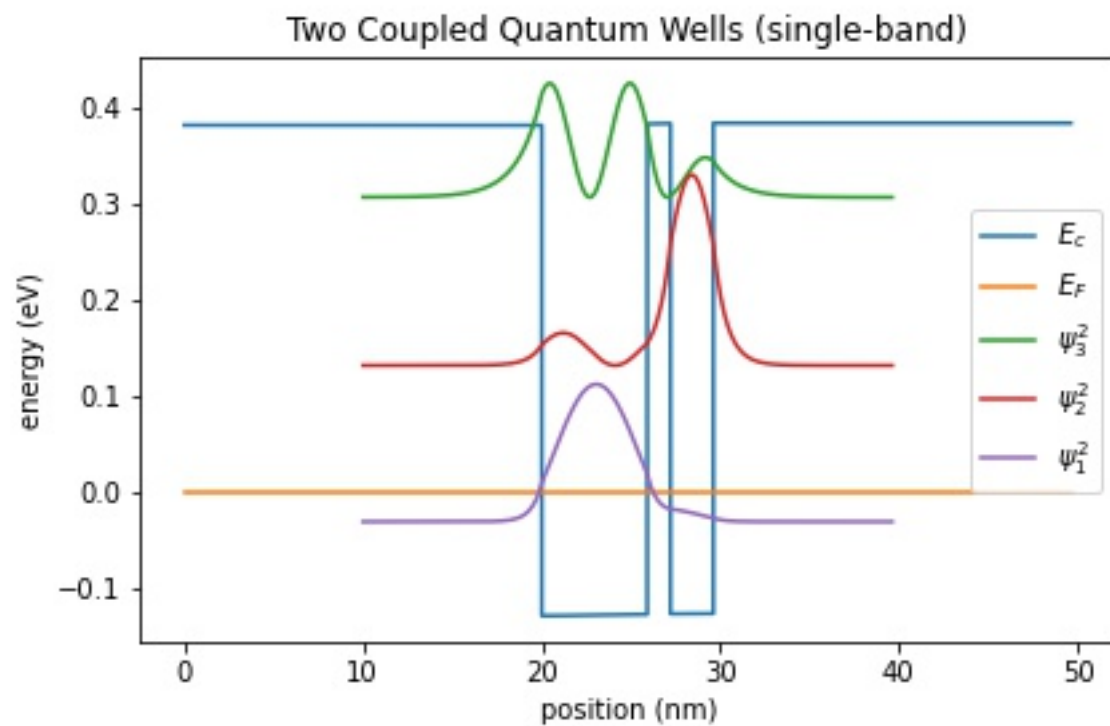


Figure 3.20.9.9: Conduction band edge, Fermi level and confined electron states of two coupled quantum wells

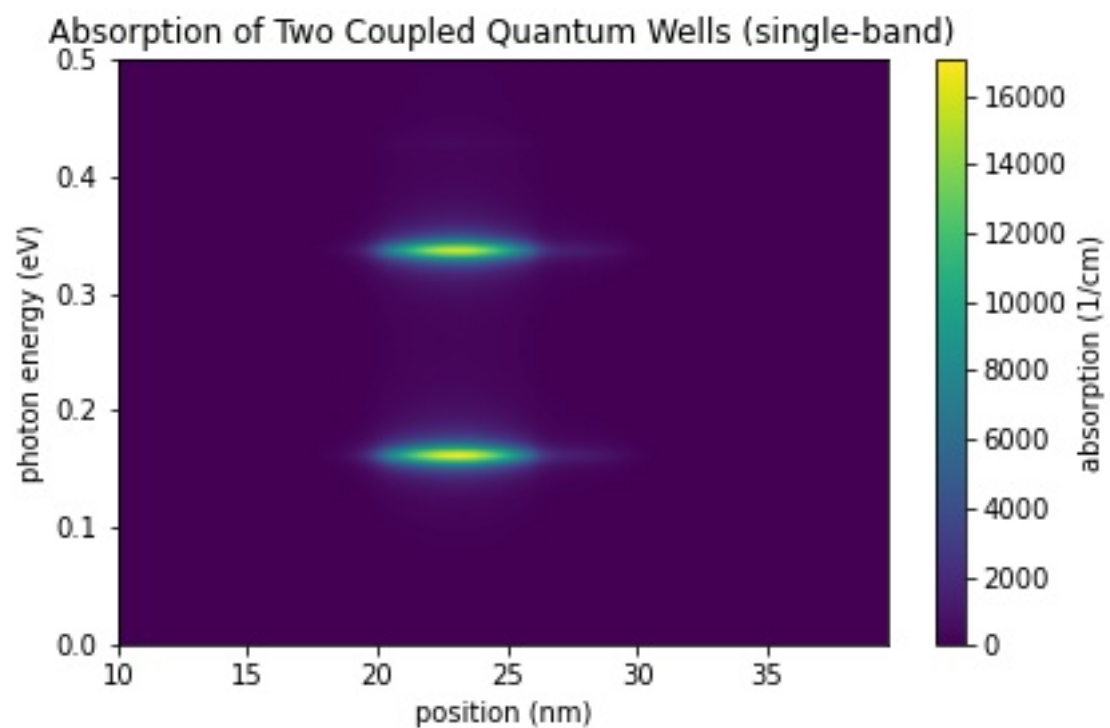


Figure 3.20.9.10: Calculated spatially resolved absorption spectrum:  $\alpha(x, E)$  of two coupled quantum wells

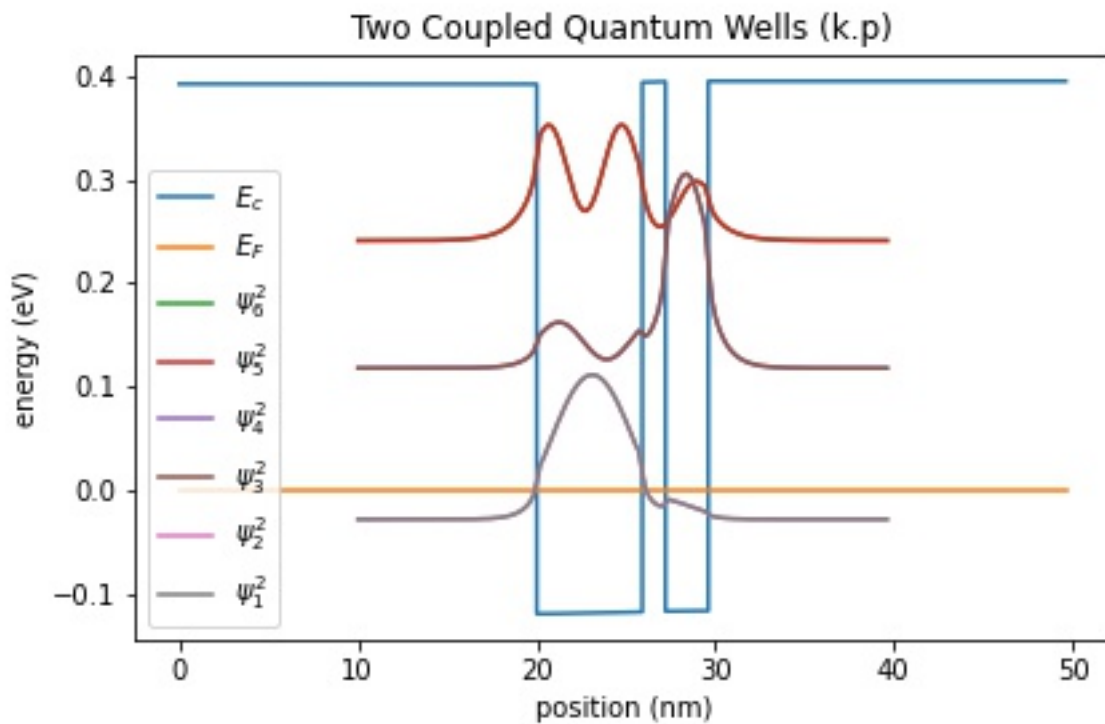


Figure 3.20.9.11: Conduction band edge, Fermi level and confined electron states of two coupled quantum wells

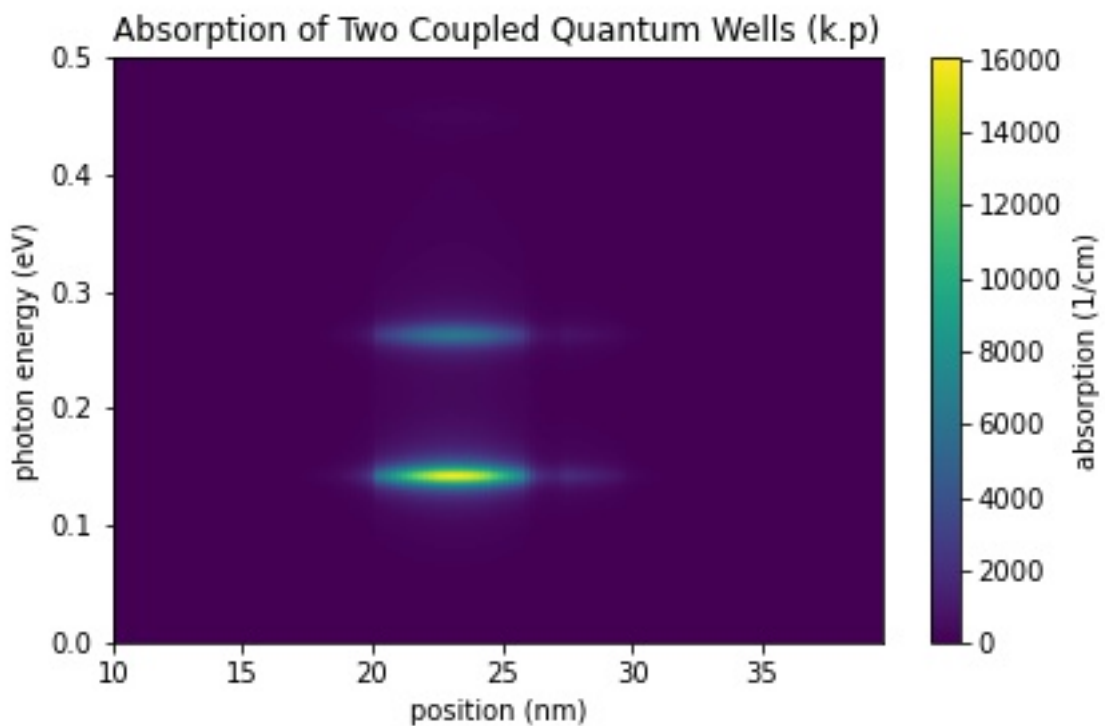


Figure 3.20.9.12: Calculated spatially resolved absorption spectrum:  $\alpha(x, E)$  of two coupled quantum wells

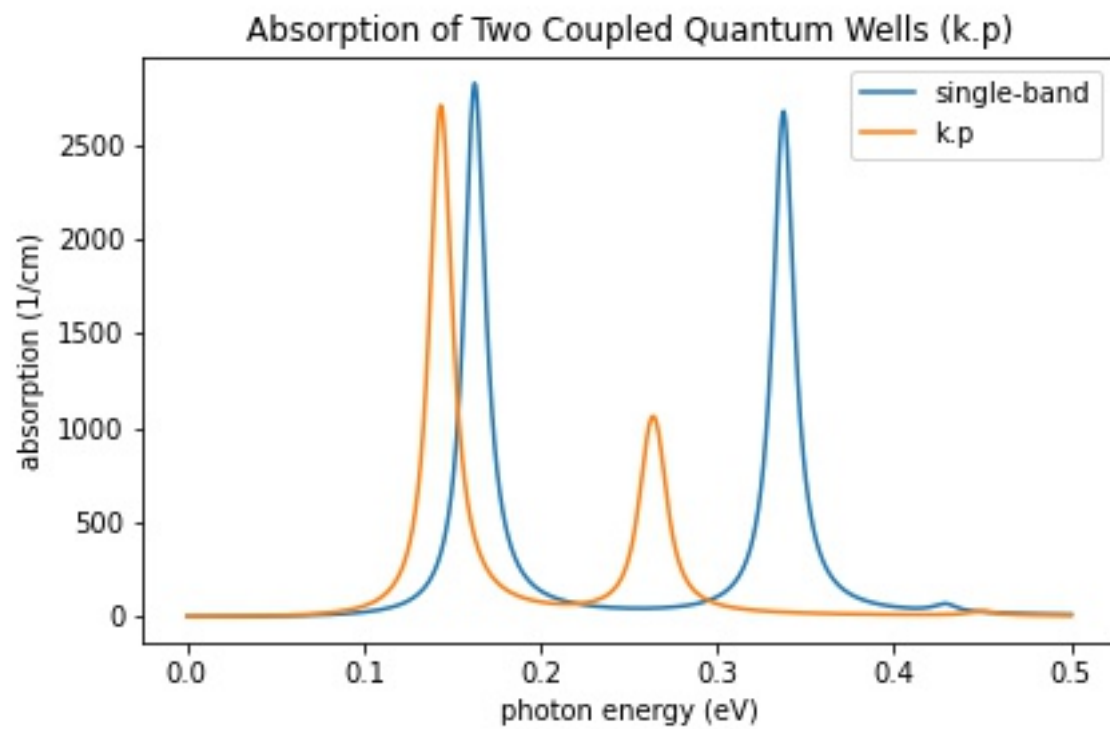


Figure 3.20.9.13: Calculated absorption spectra  $\alpha(E)$  of two coupled quantum wells

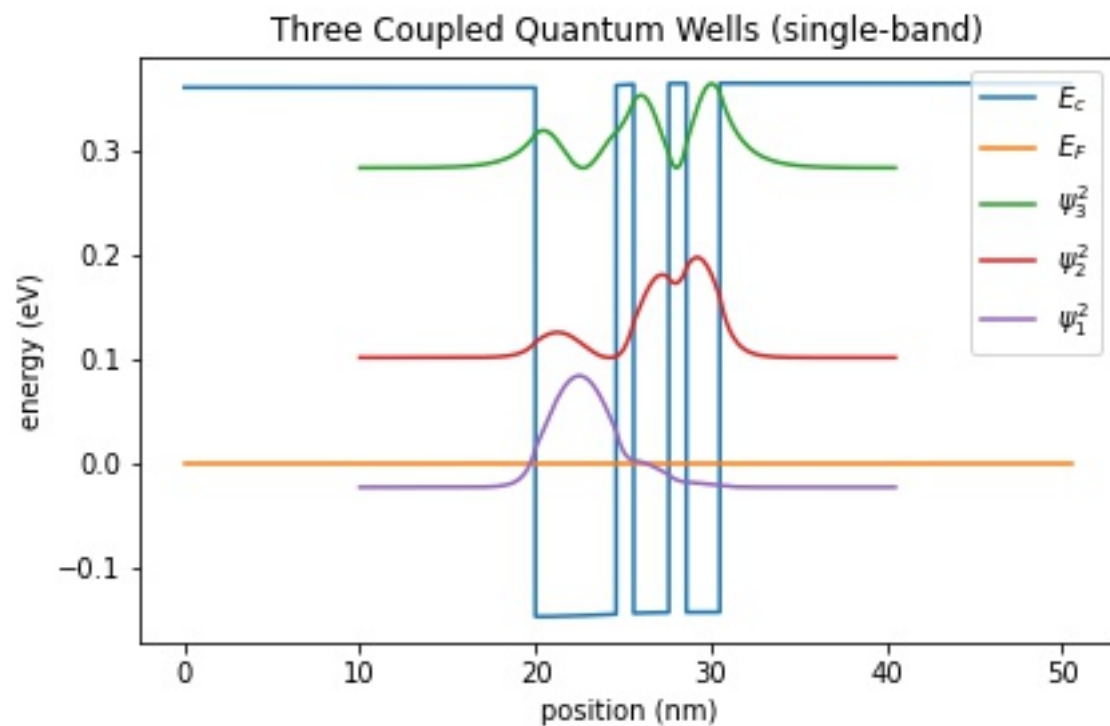


Figure 3.20.9.14: Conduction band edge, Fermi level and confined electron states of three coupled quantum wells

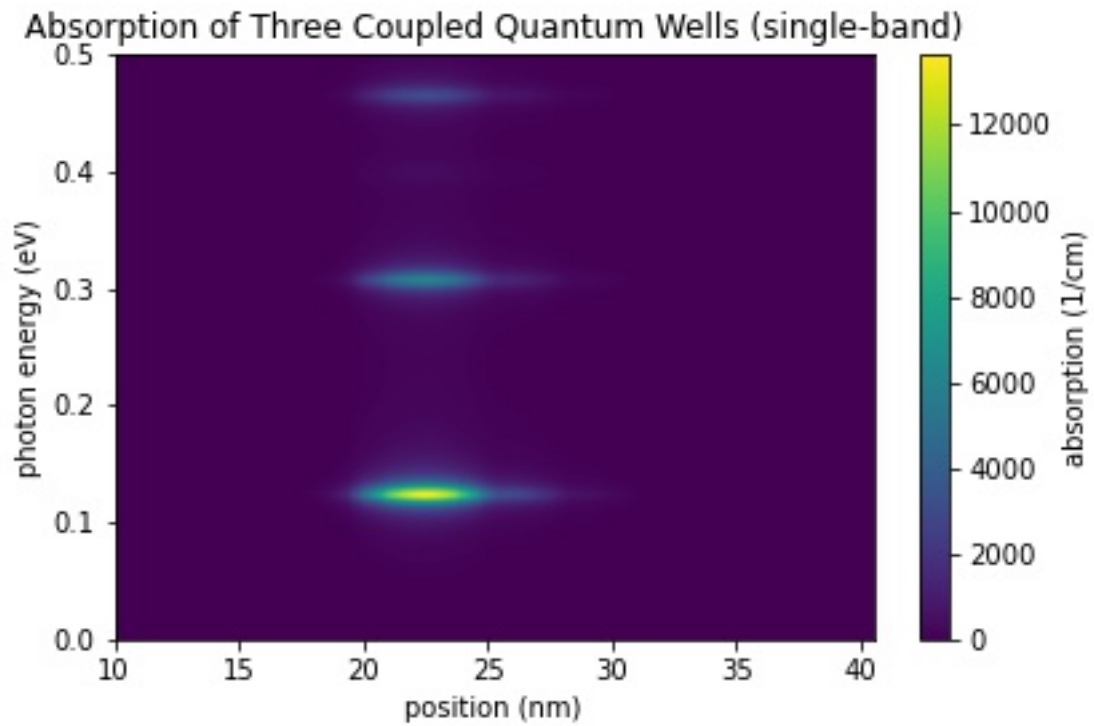


Figure 3.20.9.15: Calculated spatially resolved absorption spectrum  $\alpha(x, E)$  of three coupled quantum wells

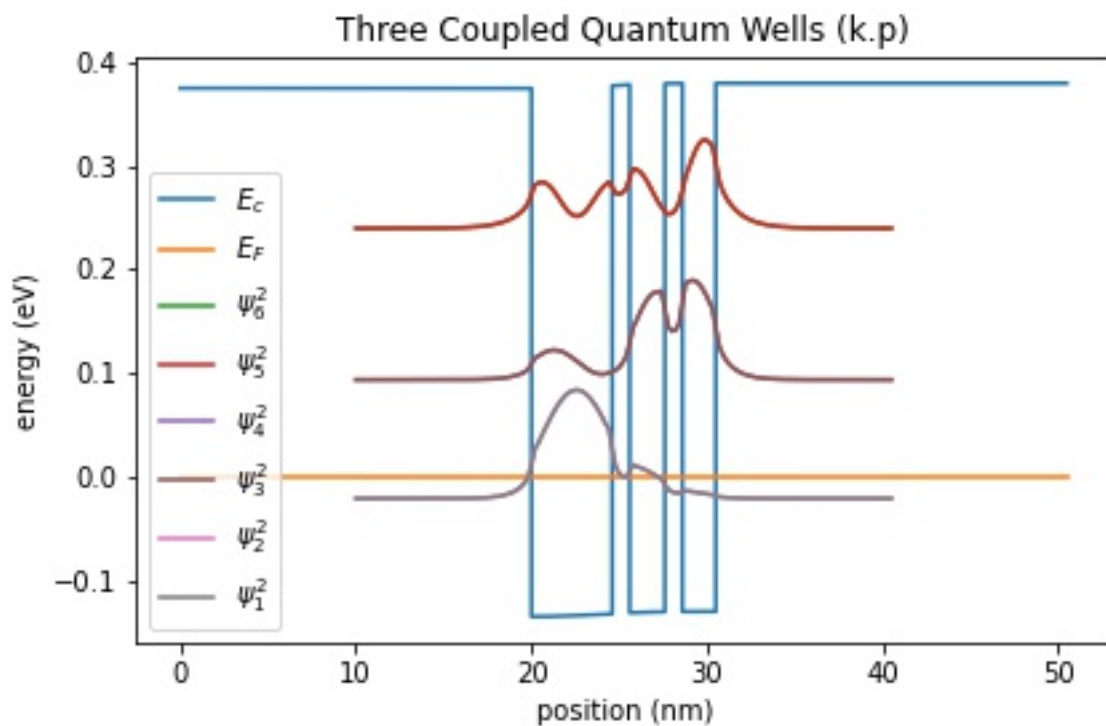


Figure 3.20.9.16: Conduction band edge, Fermi level and confined electron states of three coupled quantum wells

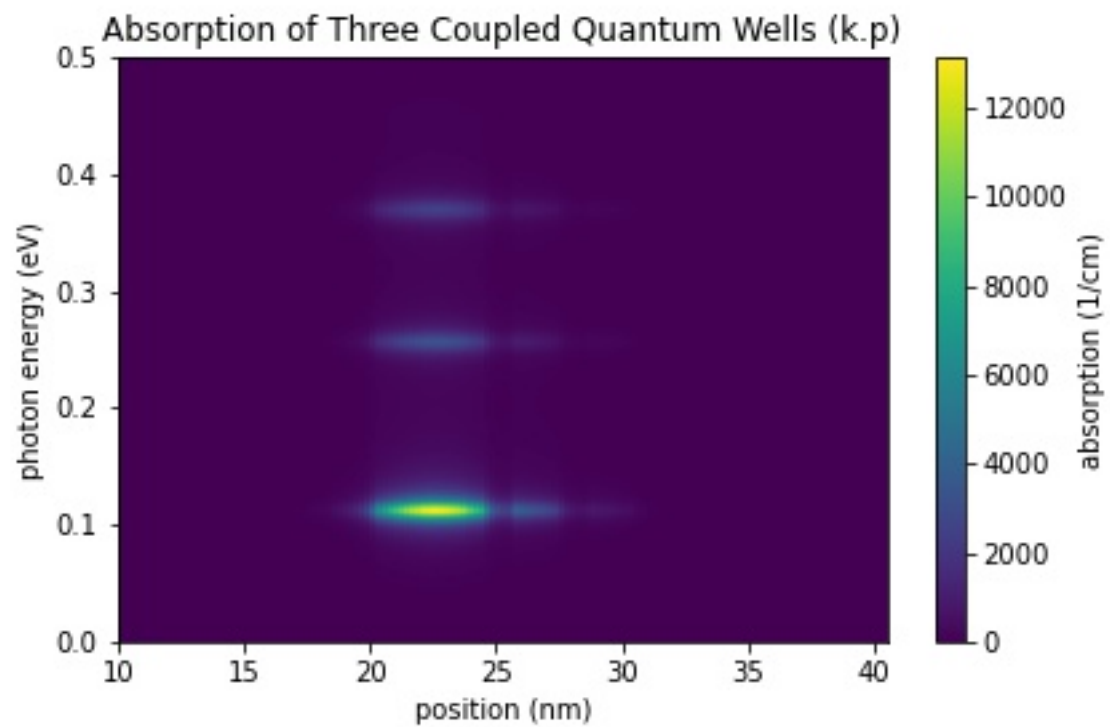


Figure 3.20.9.17: Calculated spatially resolved absorption spectrum  $\alpha(x, E)$  of three coupled quantum wells

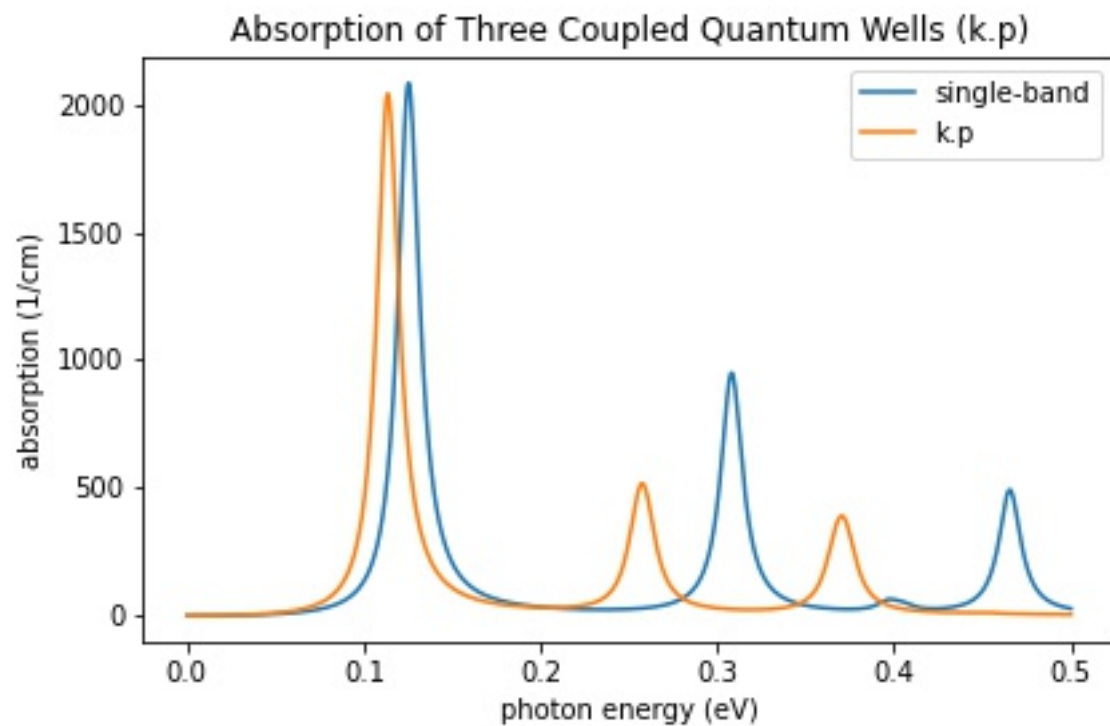


Figure 3.20.9.18: Calculated absorption spectra  $\alpha(E)$  of three coupled quantum wells



Last update: nnnn/nn/nn

### 3.20.10 Optical intraband transitions in a quantum well - Momentum matrix elements and selection rules

#### Input files:

- *IDQW\_intraband\_matrixelements\_infinite\_nnpp.in*
- *IDQW\_intraband\_matrixelements\_infinite\_kp\_nnpp.in*

#### Scope:

We consider a 10 nm *GaAs* quantum well embedded between *AlAs* barriers. The structure is assumed to be unstrained. We assume “infinite” *AlAs* barriers. (This can be achieved by choosing a band offset of 100 eV.) This way we can compare our results to analytical text books results.

#### Eigenstates and wave functions in the quantum well

Input file: *IDQW\_intraband\_matrixelements\_infinite\_nnpp.in*

```
quantum{
  ...
  momentum_matrix_elements{ # output spatial overlap of wave functions
    Gamma{}
    HH{}
    LH{}
    SO{}
    output_oscillator_strengths = yes # default is no
  }
  dipole_moment_matrix_elements{ # output dipole moment matrix elements
    Gamma{}
    HH{}
    LH{}
    SO{}
    output_oscillator_strengths = yes # default is no
  }
  transition_energies{ # output transition energies
    Gamma{}
    HH{}
    LH{}
    SO{}
  }
}
```

Figure 3.20.10.1 shows the six lowest eigenfunctions of the 1D *GaAs* quantum well. The conduction band edge of *GaAs* is assumed to be located at 0 eV.

For “infinite” barriers we obtain using single-band Schrödinger effective-mass approximation (i.e. isotropic and parabolic effective masses) the following eigenvalues:

|                 |                               |
|-----------------|-------------------------------|
| E1 = 0.05652 eV | (0.05655)                     |
| E2 = 0.22601 eV | (0.22618 = 2 <sup>2</sup> E1) |
| E3 = 0.50831 eV | (0.50891 = 3 <sup>2</sup> E1) |
| E4 = 0.90314 eV | (0.90473 = 4 <sup>2</sup> E1) |
| E5 = 1.41011 eV | (1.41365 = 5 <sup>2</sup> E1) |
| E6 = 2.02872 eV | (2.03565 = 6 <sup>2</sup> E1) |

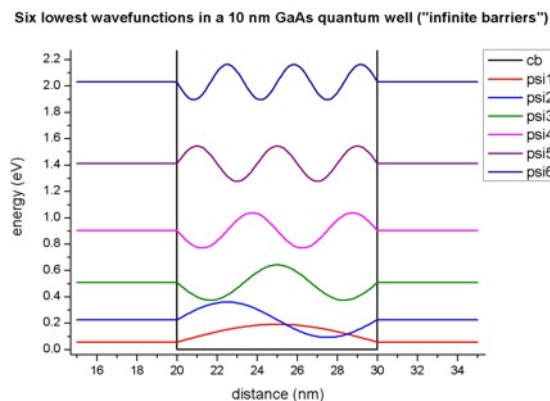


Figure 3.20.10.1: Calculated conduction band edge (black) and wave functions of confined electron states.

The analytic formula in the infinite barrier QW model reads:

$$E_n = \frac{\hbar^2}{2m_0} \left( \frac{\pi n}{L} \right)^2 = 0.056546 \cdot n^2 \text{ eV}$$

where  $L$  is the width of the quantum well ( $L = 10 \text{ nm}$ ). The analytically calculated values are given in brackets and are in excellent agreement.

### Momentum matrix elements

Light that propagates normal to the quantum well layers cannot be absorbed by intraband transitions. However, if the light propagates in the plane of the well (i.e. the electric field is oriented normal to the quantum well layers), intersubband absorption occurs.

To understand optical intraband (= intersubband) transitions for light that travels in the plane of the QW, we have to examine the intersubband dipole moment:

$$M_{fi} = \langle \psi_f | x | \psi_i \rangle = \int_{-\infty}^{\infty} \psi_f^*(x) x \psi_i(x) dx$$

where  $|\psi\rangle$  is the envelope function of the relevant state (within the same band).

In our case, we have a symmetric quantum well with infinite barriers, thus our envelope functions are either symmetric or antisymmetric. The intersubband matrix elements will vanish if the envelope functions have the same parity, e.g.  $M_{13} = M_{31} = 0$ . In this simple example, the matrix elements can be calculated analytically, e.g.  $M_{12} = (16/9\pi^2) L = 1.8013 \text{ nm}$ . *nextnano++* gives the following results:

$$M_{12} = M_{21} = 1.80143 \text{ nm}$$

$$M_{13} = M_{31} = 1.9463e^{-15} \text{ nm}$$

For the “infinite” QW barrier model, this matrix element is independent of the effective mass, thus the matrix elements in the conduction band are the same as in the valence bands (single-band approximation).

A useful quantity is the oscillator strength  $f_{fi}$  which is defined as follows:

$$f_{fi} = \frac{2m_0}{\hbar^2} (E_f - E_i) \cdot |M_{fi}|^2.$$

$f_{21}$  for our simple infinite barrier example is given by  $f_{21} = 256/(27 \pi^2) = 0.9607$  and is independent of the well width. The *nextnano++* result is:

$$f_{21} = 0.9603 = -f_{12}$$

We can also see that this is a strong transition because all transitions from state ‘1’ to state ‘f’ must add up to unity (so-called “f-sum rule”):

$$\sum_f f_{fi} = 1.0$$

(Thomas-Kuhn sum rule for constant effective mass  $m^*$ .) Thus all other transitions are much weaker.

It is interesting to look at the transitions starting from the second level  $i = 2$ . The lowest oscillator strength  $f_{12} = -0.96$  is negative, but the sum over all  $f_{f2}$  must still give unity, thus oscillator strengths larger than 1.0 are possible, e.g.  $f_{32} = 1.87$ .

The intersubband dipole moments and the oscillator strengths are contained in these files:

- *bias\_00000\Quantum\dipole\_moment\_matrix\_elements\_quantum\_region\_Gamma\_100.txt*
- *bias\_00000\Quantum\dipole\_moment\_matrix\_elements\_quantum\_region\_HH\_100.txt*
- *bias\_00000\Quantum\dipole\_moment\_matrix\_elements\_quantum\_region\_LH\_100.txt*
- *bias\_00000\Quantum\dipole\_moment\_matrix\_elements\_quantum\_region\_SO\_100.txt*

For each transition, the transition energy is given in

- *bias\_00000\Quantum\transition\_energies\_quantum\_region\_Gamma.txt*
- *bias\_00000\Quantum\transition\_energies\_quantum\_region\_HH.txt*
- *bias\_00000\Quantum\transition\_energies\_quantum\_region\_LH.txt*
- *bias\_00000\Quantum\transition\_energies\_quantum\_region\_SO.txt*

The effective masses that have been used for the calculation of the oscillator strengths are also indicated. They are calculated by building an average of the parallel effective masses for each grid point, weighted by the square of the wave function on each grid point. In this particular example, the effective masses are constant and do not vary with position ( $m_{||} = 0.0665 m_0$ ). (Assuming that the masses are isotropic, it is fine to use the parallel effective masses.)

```
-----
Intersubband transitions
=> Gamma conduction band
-----

Electric field in z-direction [kV/cm]: 0.0000000E+00
-----

Intersubband dipole moment | < psi_f* | z | psi_i > | [Angstrom]
-----|-----|-----|-----|-----|
Oscillator strength []
-----|-----|-----|-----|-----|
Energy of transition [eV]
-----|-----|-----|-----|-----|
m* [m_0]
-----|-----|-----|-----|-----|
<psi001*|z|psi001> 249.0000
<psi002*|z|psi001> 18.01673 0.9602799 0.1694912 6.6500001E-02
<psi003*|z|psi001> 6.1430171E-07 2.9757722E-15 0.4517909 6.6500001E-02 (same
->parity: symmetric)
<psi004*|z|psi001> 1.441336 3.0698571E-02 0.8466209 6.6500001E-02
<psi005*|z|psi001> 1.6007220E-07 6.0536645E-16 1.353592 6.6500001E-02 (same
->parity: symmetric)
<psi006*|z|psi001> 0.3971010 5.4281605E-03 1.972205 6.6500001E-02
<psi007*|z|psi001> 5.1874160E-08 1.2690011E-16 2.701849 6.6500001E-02 (same
->parity: symmetric)
<psi008*|z|psi001> 0.1634139 1.6508275E-03 3.541806 6.6500001E-02
...
<psi020*|z|psi001> 1.0178176E-02 3.9451432E-05 21.81846 6.6500001E-02
Sum rule of oscillator strength: f_psi001 = 0.9994023
<psi001*|z|psi002> 18.01673 -0.9602799 -0.1694912 6.6500001E-02
```

(continues on next page)

(continued from previous page)

```

<psi002*|z|psi002> 249.0000
<psi003*|z|psi002> 19.45806      1.865556      0.2822997  6.6500001E-02
<psi004*|z|psi002> 2.0636767E-06  5.0333130E-14  0.6771297  6.6500001E-02 (same
↳parity: antisymmetric)
<psi005*|z|psi002> 1.838436      6.9852911E-02  1.184101   6.6500001E-02
<psi006*|z|psi002> 1.4976163E-08  7.0571038E-18  1.802713   6.6500001E-02 (same
↳parity: antisymmetric)
<psi007*|z|psi002> 0.5605143      1.3886644E-02  2.532358   6.6500001E-02
<psi008*|z|psi002> 8.7380023E-08  4.4941879E-16  3.372315   6.6500001E-02 (same
↳parity: antisymmetric)
<psi009*|z|psi002> 0.2461317      4.5697703E-03  4.321757   6.6500001E-02
<psi010*|z|psi002> 8.3240280E-07  6.5062044E-14  5.379748   6.6500001E-02 (same
↳parity: antisymmetric)
<psi011*|z|psi002> 0.1302904      1.9393204E-03  6.545245   6.6500001E-02
...
<psi020*|z|psi002> 2.7233656E-07  2.8025147E-14  21.64897   6.6500001E-02
Sum rule of oscillator strength: f_psi002 = 0.9975320

<psi001*|z|psi003> 6.1430171E-07 -2.9757722E-15 -0.4517909  6.6500001E-02 (same
↳parity: symmetric)
<psi002*|z|psi003> 19.45806      -1.865556      -0.2822997  6.6500001E-02
<psi003*|z|psi003> 249.0000
<psi004*|z|psi003> 19.85515      2.716784      0.3948300   6.6500001E-02
<psi005*|z|psi003> 6.4708888E-07  6.5907892E-15  0.9018011   6.6500001E-02 (same
↳parity: symmetric)
<psi006*|z|psi003> 2.001849      0.1063465      1.520414   6.6500001E-02
<psi007*|z|psi003> 3.9201248E-07  6.0352080E-15  2.250058   6.6500001E-02 (same
↳parity: symmetric)
<psi008*|z|psi003> 0.6432316      2.2314854E-02  3.090015   6.6500001E-02
<psi009*|z|psi003> 2.6240454E-07  4.8547223E-15  4.039457   6.6500001E-02 (same
↳parity: symmetric)
...
<psi020*|z|psi003> 3.1797737E-02  3.7707522E-04  21.36667   6.6500001E-02
Sum rule of oscillator strength: f_psi003 = 0.9945912

```

The commonly used intersubband dipole moment  $\langle \psi_f | x | \psi_i \rangle$  [nm] depends on the choice of origin for the matrix elements when  $f = i$ , thus the user might prefer to output the Intersubband dipole moment  $\langle \psi_f | p_x | \psi_i \rangle$  which are the intersubband dipole moments

$$N_{fi} = \langle \psi_f | \hat{p}_x | \psi_i \rangle = \int_{-\infty}^{\infty} \psi_f^*(x) \hat{p}_x \psi_i(x) dx = -i\hbar \int_{-\infty}^{\infty} \psi_f^*(x) \frac{\partial}{\partial x} \psi_i(x) dx$$

and the oscillator strengths

$$f_{fi} = \frac{2m_0}{\hbar^2} (E_f - E_i) \cdot |M_{fi}|^2 = \frac{2}{m_0(E_f - E_i)} \cdot |N_{fi}|^2$$

between all calculated states in each band from min to max eigenvalues. In the simple QW of this tutorial, the matrix elements can be calculated analytically, e.g.  $N_{21} = 8\hbar/3L = 0.2666 \hbar/\text{nm}$ . The *nextnano++* result is:

$$N_{21} = N_{12} = 0.265957 \hbar/\text{nm}$$

$$N_{31} = N_{13} = 7.05011e^{-17}$$

The oscillator strength  $f_{21}$  for our simple infinite barrier example is given by  $f_{21} = 256/(27\pi^2) = 0.9607$  and is independent of the well width. The *nextnano++* result is:

$$f_{21} = -f_{12} = 0.9603$$

The intersubband dipole moments and the oscillator strengths are contained in these files:

- *bias\_00000\Quantum\momentum\_matrix\_elements\_quantum\_region\_Gamma\_100.txt*
- *bias\_00000\Quantum\intraband\_\_matrix\_elements\_quantum\_region\_HH\_100.txt*
- *bias\_00000\Quantum\momentum\_matrix\_elements\_quantum\_region\_LH\_100.txt*
- *bias\_00000\Quantum\momentum\_matrix\_elements\_quantum\_region\_SO\_100.txt*

The numbers show a comparison between the  $x$  and the  $p_x$  matrix elements for *nextnano*<sup>3</sup>:

| -----                                                 |               |                                                     |                                        |               |                          |
|-------------------------------------------------------|---------------|-----------------------------------------------------|----------------------------------------|---------------|--------------------------|
| Intersubband dipole moment                            |               |                                                     | < psi_f*   z   psi_i >   [Angstrom]    |               |                          |
| Intersubband dipole moment                            |               |                                                     | < psi_f*   p   psi_i >   [h_bar / m_0] |               |                          |
| -----                                                 |               |                                                     |                                        |               |                          |
| Oscillator strength []                                |               |                                                     |                                        |               |                          |
| -----                                                 |               |                                                     |                                        |               |                          |
| Energy of transition [eV]                             |               |                                                     |                                        |               |                          |
| -----                                                 |               |                                                     |                                        |               |                          |
| m* [m_0]                                              |               |                                                     |                                        |               |                          |
| -----                                                 |               |                                                     |                                        |               |                          |
| <psi001* z psi001>                                    | 249.0000      | (matrix element <1 1> depends on choice of origin!) |                                        |               |                          |
| <psi001* p psi001>                                    | 4.3405972E-19 | (matrix element <1 1> independent of origin)        |                                        |               |                          |
| <psi002* z psi001>                                    | 18.01673      | 0.9602799                                           | 0.1694912                              | 6.6500001E-02 |                          |
| <psi002* p psi001>                                    | 2.6649671E-02 | 0.9602799                                           | 0.1694912                              | 6.6500001E-02 |                          |
| <psi003* z psi001>                                    | 6.1430171E-07 | 2.9757722E-15                                       | 0.4517909                              | 6.6500001E-02 | (same parity: symmetric) |
| <psi003* p psi001>                                    | 2.7325134E-18 |                                                     |                                        |               |                          |
| <psi004* z psi001>                                    | 1.441336      | 3.0698571E-02                                       | 0.8466209                              | 6.6500001E-02 |                          |
| <psi004* p psi001>                                    | 1.0649348E-02 | 3.0698579E-02                                       | 0.8466209                              | 6.6500001E-02 |                          |
| <psi005* z psi001>                                    | 1.6007220E-07 | 6.0536645E-16                                       | 1.353592                               | 6.6500001E-02 | (same parity: symmetric) |
| <psi005* p psi001>                                    | 6.9518724E-18 |                                                     |                                        |               |                          |
| <psi006* z psi001>                                    | 0.3971010     | 5.4281605E-03                                       | 1.972205                               | 6.6500001E-02 |                          |
| <psi006* p psi001>                                    | 6.8347314E-03 | 5.4281540E-03                                       | 1.972205                               | 6.6500001E-02 |                          |
| <psi007* z psi001>                                    | 5.1874160E-08 | 1.2690011E-16                                       | 2.701849                               | 6.6500001E-02 | (same parity: symmetric) |
| <psi007* p psi001>                                    | 2.8686024E-19 |                                                     |                                        |               |                          |
| <psi008* z psi001>                                    | 0.1634139     | 1.6508275E-03                                       | 3.541806                               | 6.6500001E-02 |                          |
| <psi008* p psi001>                                    | 5.0510615E-03 | 1.6508278E-03                                       | 3.541806                               | 6.6500001E-02 |                          |
| ...                                                   |               |                                                     |                                        |               |                          |
| <psi020* z psi001>                                    | 1.0178176E-02 | 3.9451432E-05                                       | 21.81846                               | 6.6500001E-02 |                          |
| <psi020* p psi001>                                    | 1.9380626E-03 | 3.9452334E-05                                       | 21.81846                               | 6.6500001E-02 |                          |
| Sum rule of oscillator strength: f_psi001 = 0.9994023 |               |                                                     |                                        |               |                          |
| Sum rule of oscillator strength: f_psi001 = 0.9994023 |               |                                                     |                                        |               |                          |

### 8-band k.p calculation for $k_{||} = (K_y, k_z) = 0$

The following input file performs the same calculations as above but this time using the 8-band k.p model: *IDQW\_intraband\_matrixelements\_infinite\_kp\_nnp.in*.

We modified the 8-band k.p parameters and decoupled (!) the electrons from the holes ( $EP = 0$  eV,  $S = 1/m_e$ ). This way we have an effective single-band model, and thus we are able to compare the k.p results to the single-band

results in order to check for consistency.

The numbering of the k.p eigenstates differs slightly from the single-band eigenstates because the k.p eigenstates are two-fold spin-degenerate. The actual values for the matrix elements are identical (assuming a decoupled k.p Hamiltonian, i.e. a single-band Hamiltonian).

Note that the single-band definition of the oscillator strength does not really make sense for a k.p calculation where the masses usually are anisotropic, non-parabolic and are different on each grid point (due to different materials and different strain tensors).

For the calculation of the oscillator strength in a k.p calculation, the user can specify suitable masses by overwriting the default entries. Of course, the masses that are used to calculate the k.p eigenstates have to be specified via the 6-band and 8-band k.p parameters (inside the database{ } group).

The intersubband dipole moments and the oscillator strengths are contained in this file:

- *bias\_00000\Quantum\momentum\_matrix\_elements\_quantum\_region\_kp8\_100.txt* ( $p_x$  elements)
- *bias\_00000\Quantum\dipole\_moment\_matrix\_elements\_quantum\_region\_kp8\_100.txt* ( $x$  elements)

Note that the two-fold spin-degeneracy in single-band is counted explicitly in k.p.

| -----              |                            |                                                     |               |               |
|--------------------|----------------------------|-----------------------------------------------------|---------------|---------------|
|                    | Intersubband dipole moment | < psi_f*   z   psi_i >                              | [Angstrom]    |               |
| ↪Angstrom]         | Intersubband dipole moment | < psi_f*   p   psi_i >                              | [h_bar / m_0] |               |
| -----              |                            |                                                     |               |               |
|                    | Oscillator strength []     |                                                     |               |               |
| -----              |                            |                                                     |               |               |
|                    | Energy of transition [eV]  |                                                     |               |               |
| -----              |                            |                                                     |               |               |
|                    | m* [m_0]                   |                                                     |               |               |
| -----              |                            |                                                     |               |               |
| <psi001* z psi001> | 249.0000                   | (matrix element <1 1> depends on choice of origin!) |               |               |
| <psi002* z psi001> | 249.0000                   | (matrix element <2 1> depends on choice of origin!) |               |               |
| <psi001* p psi001> | 1.8126842E-18              | (matrix element <1 1> independent of origin)        |               |               |
| <psi002* p psi001> | 1.8126842E-18              | (matrix element <2 1> independent of origin)        |               |               |
|                    |                            |                                                     |               |               |
| <psi003* z psi001> | 18.01673                   | 0.9602799                                           | 0.1694912     | 6.6500001E-02 |
| <psi004* z psi001> | 18.01673                   | 0.9602799                                           | 0.1694912     | 6.6500001E-02 |
| <psi003* p psi001> | 2.6649671E-02              | 0.9602798                                           | 0.1694912     | 6.6500001E-02 |
| <psi004* p psi001> | 2.6649671E-02              | 0.9602798                                           | 0.1694912     | 6.6500001E-02 |
|                    |                            |                                                     |               |               |
| <psi005* z psi001> | 3.5382732E-13              |                                                     |               |               |
| <psi006* z psi001> | 3.5382732E-13              |                                                     |               |               |
| <psi005* p psi001> | 2.1414240E-15              |                                                     |               |               |
| <psi006* p psi001> | 2.1414240E-15              |                                                     |               |               |
|                    |                            |                                                     |               |               |
| <psi007* z psi001> | 1.441336                   | 3.0698583E-02                                       | 0.8466209     | 6.6500001E-02 |
| <psi008* z psi001> | 1.441336                   | 3.0698583E-02                                       | 0.8466209     | 6.6500001E-02 |
| <psi007* p psi001> | 1.0649348E-02              | 3.0698583E-02                                       | 0.8466209     | 6.6500001E-02 |
| <psi008* p psi001> | 1.0649348E-02              | 3.0698583E-02                                       | 0.8466209     | 6.6500001E-02 |
|                    |                            |                                                     |               |               |
| <psi009* z psi001> | 7.2598817E-13              |                                                     |               |               |
| <psi010* z psi001> | 7.2598817E-13              |                                                     |               |               |
| <psi009* p psi001> | 1.0445775E-14              |                                                     |               |               |
| <psi010* p psi001> | 1.0445775E-14              |                                                     |               |               |
|                    |                            |                                                     |               |               |
| <psi011* z psi001> | 0.3971008                  | 5.4281550E-03                                       | 1.972205      | 6.6500001E-02 |
| <psi012* z psi001> | 0.3971008                  | 5.4281550E-03                                       | 1.972205      | 6.6500001E-02 |
| <psi011* p psi001> | 6.8347319E-03              | 5.4281550E-03                                       | 1.972205      | 6.6500001E-02 |

(continues on next page)

(continued from previous page)

```

<psi012*|p|psi001> 6.8347319E-03 5.4281550E-03 1.972205 6.6500001E-02
...
<psi039*|z|psi001> 1.0178294E-02 3.9452352E-05 21.81846 6.6500001E-02
<psi040*|z|psi001> 1.0178294E-02 3.9452352E-05 21.81846 6.6500001E-02
<psi039*|p|psi001> 1.9380630E-03 3.9452349E-05 21.81846 6.6500001E-02
<psi040*|p|psi001> 1.9380630E-03 3.9452349E-05 21.81846 6.6500001E-02

Sum rule of oscillator strength: f_psi001 = 0.9994023
Sum rule of oscillator strength: f_psi001 = 0.9994023

```

Last update: 27/05/2025

### 3.20.11 — FREE — GaAs p–n junction

Author Stefan Birner

#### Note

See a [tutorial on IV curves](#) for pn junctions described here

#### Input Files:

- `pn_junction_GaAs_1D_nnp.in`
- `pn_junction_GaAs_2D_nnp.in`
- `pn_junction_GaAs_3D_nnp.in`

This tutorial discusses the `nextnano++` input file. Identical results can be achieved with the `nextnano3` input files listed above.

This tutorial aims to reproduce Figure 3.1 (p. 51) of Joachim Piprek’s book “*Semiconductor Optoelectronic Devices - Introduction to Physics and Simulation*” (Section 3.2 “pn-junctions”)

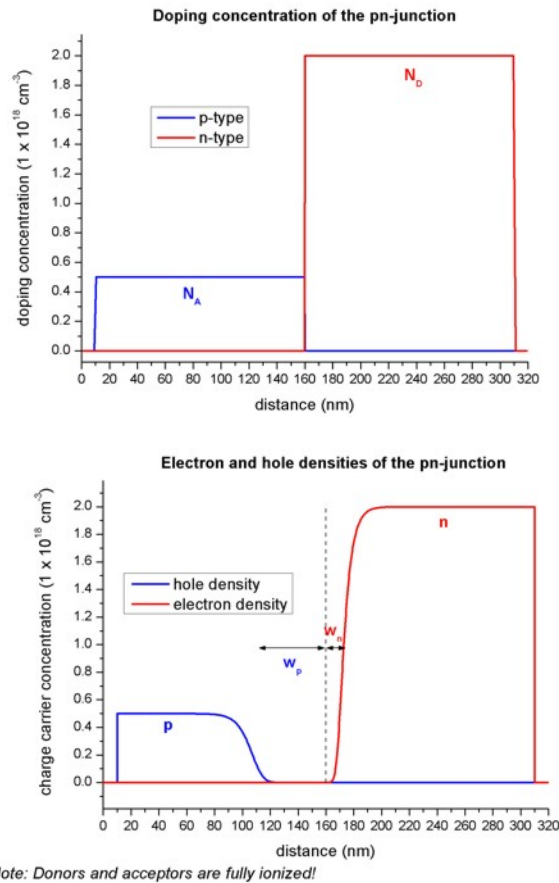
#### Doping concentration

The structure consists of 300 nm GaAs. At the left and right boundaries, metal contacts are connected to the GaAs semiconductor (i.e. from 0 nm to 10 nm, and from 310 nm to 320 nm). The structure is p-type doped from 10 nm to 160 nm and n-type doped from 160 nm and 310 nm.

The following figure shows the concentration of donors and acceptors of the p-n junction. In the p-type region between 10 nm and 160 nm, the number of acceptors,  $N_A$  is  $0.5 \times 10^{18} \text{ cm}^{-3}$ . In the n-type region between 160 nm and 310 nm, the number of donors,  $N_D$  is  $2.0 \times 10^{18} \text{ cm}^{-3}$ .

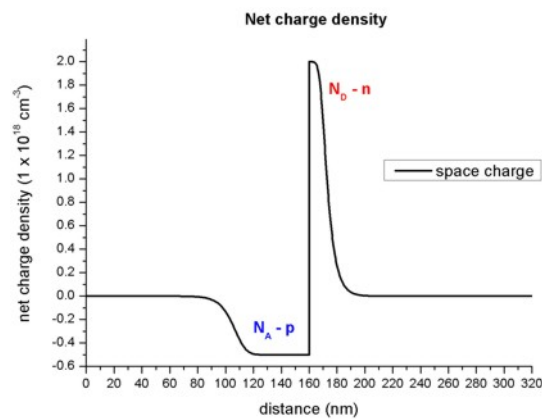
#### Carrier concentrations

The equilibrium condition for a p-n junction is achieved by a small transfer of electrons from the n region to the p region, where they recombine with holes. This leads to a **depletion region** (depletion width =  $w_p + w_n$ ), i.e. the region around the p-n junction only has very few free carriers left. The following figure shows the electron and hole densities and the depletion region around the p-n junction at 160 nm. Here, we assumed that all donors and acceptors are fully ionized.



### Net charges (space charge)

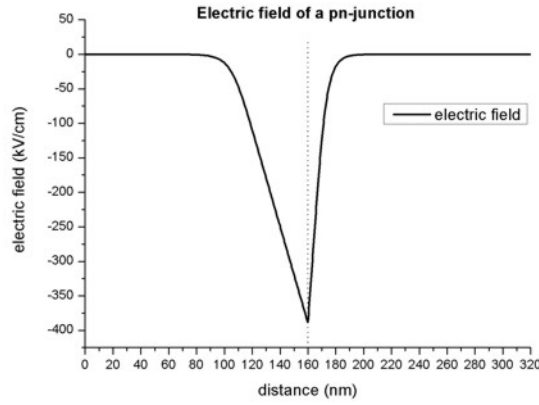
In the depletion region, a net charge results from the ionized donors  $N_D$  and ionized acceptors  $N_A$ . The following figure shows the net charge density of the p-n junction.



### Electric field

The slope of the electric field is proportional to the net charge (Poisson equation), thus the extremum of the electric field is expected to be at the p-n junction. In regions without charges, the electric field is zero. The following figure shows the electric field of the p-n junction.





The extremum of the electric field  $F_{max}$  (at 160 nm) can be approximated as follows:

$$F_{max} = \frac{-eN_A w_p}{\epsilon \epsilon_0} = -6.997 \times 10^{14} \text{V/m}^2 w_p = 387 \text{kV/cm}$$

$$= \frac{-eN_D w_n}{\epsilon \epsilon_0} = -2.799 \times 10^{15} \text{V/m}^2 w_n = 386 \text{kV/cm}$$

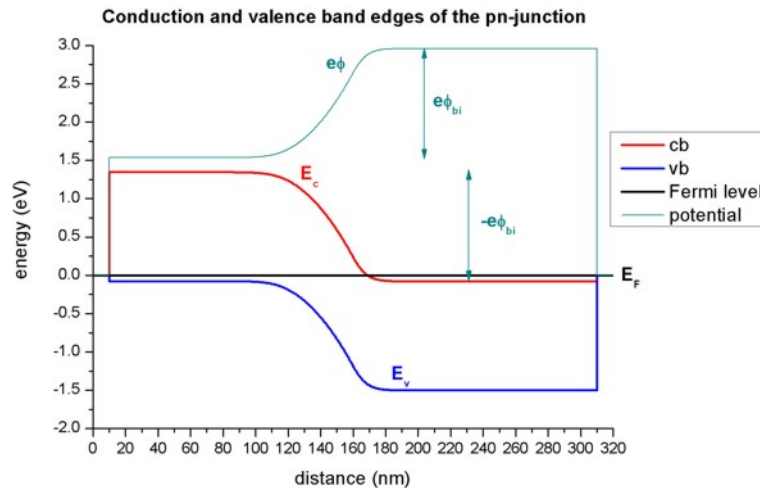
| Symbol       | Value                                 |
|--------------|---------------------------------------|
| e            | $1.6022 \times 10^{-19} \text{As}$    |
| $\epsilon$   | 12.93 (Dielectric constant of GaAs)   |
| $\epsilon_0$ | $8.854 \times 10^{12} \text{As/(Vm)}$ |
| $N_A$        | $0.5 \times 10^{18} \text{cm}^{-3}$   |
| $N_D$        | $2.0 \times 10^{18} \text{cm}^{-3}$   |
| $w_p$        | 55.3 nm                               |
| $w_n$        | 13.8 nm                               |

### Electrostatic potential, conduction and valence band edges

In regions, where the electric field is zero, the electrostatic potential is constant. The electrostatic potential  $\phi$  determines the conduction and valence band edges:

- $E_c = E_{c0} - e\phi$
- $E_v = E_{v0} - e\phi$

The following figure shows the conduction and valence band edges, the electrostatic potential and the Fermi level of the p-n junction.



Without external bias (i.e. equilibrium), the Fermi level  $E_F$  is constant ( $E_F = 0 \text{eV}$ ).

The built-in potential  $\phi_{bi}$  was calculated by *nextnano++* to be equal to 1.426 V. It can be approximated as follows:

$$\phi_{bi} = F_{\max}(w_p + w_n)/2$$

Assuming  $F_{\max} = 387 \text{ kV/cm}$ , this would result in a depletion width:  $w_p + w_n = 73.7 \text{ nm}$

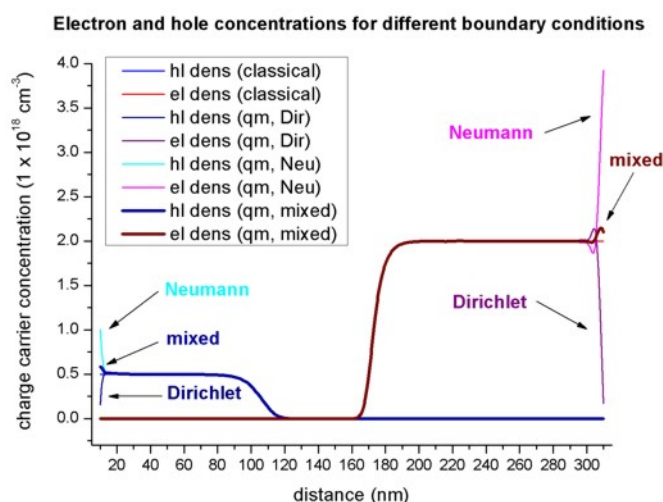
To allow for a constant chemical potential (i.e. constant Fermi level  $E_F$ ), a total potential difference of  $-e\phi_{bi}$  is required.

### Quantum mechanical solution

Using the *nextnano<sup>3</sup>* input file *pn\_junction\_GaAs\_1D\_QM\_nn3.in*, we can solve the Schrödinger equation for the electrons, light and heavy holes in the single-band approximation over the whole device, rather than classically. We calculate up to 300 eigenvalues for each band. Thus the electron and hole densities are calculated **purely quantum mechanically**. The following figure shows the electron and hole concentrations for the classical and quantum mechanical calculations. For the QM calculations, different boundary conditions were used.

- **Dirichlet** boundary conditions force the wave functions to be zero at the boundaries, thus the density goes to zero at the boundaries which is unphysically.
- **Neumann** boundary conditions lead to unphysically large values at the boundaries.

For the classical calculation, the densities at the boundaries are constant. Nevertheless, in the interesting region around the p-n junction, all four options lead to identical densities.



The following figure shows the band edges of the p-n junction for the four cases:

- Classical calculation
- Quantum mechanical calculation with **Dirichlet** boundary conditions
- Quantum mechanical calculation with **Neumann** boundary conditions
- Quantum mechanical calculation with mixed boundary conditions (this feature is no longer supported)

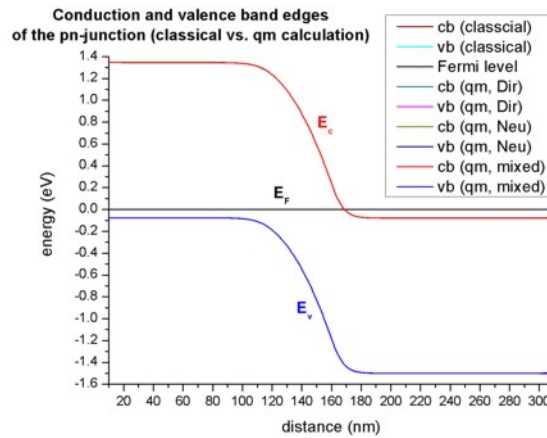
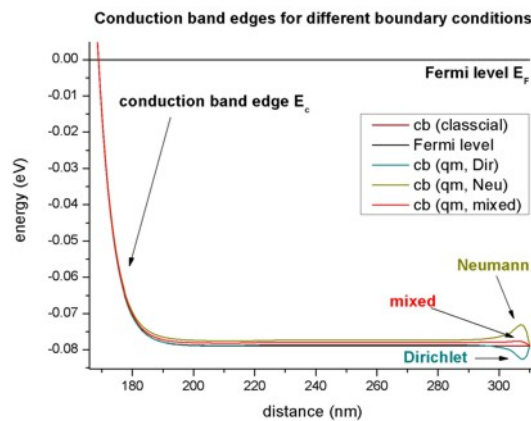
For all cases the band edges are identical in the area around the p-n junction. Tiny deviations exist at the boundaries of the device.

This figure is a zoom into the right boundary of the conduction band edge. On this scale, the tiny deviations for the different boundary conditions can be clearly seen.

### Non-equilibrium

So-called “quasi-Fermi levels” which are different for electrons ( $E_{Fn}$ ) and holes ( $E_{Fp}$ ) are used to describe nonequilibrium carrier concentrations.

In equilibrium the quasi-Fermi levels are constant and have the same value for both electrons and holes ( $E_{Fn} = E_{Fp} = 0 \text{ eV}$ ). The current is proportional to the mobility and the gradient of the quasi-Fermi level  $E_F$ .



## 2D/3D Simulations

- `pn_junction_GaAs_2D_nn3.in` / `*_nnp.in` - input file for the *nextnano*<sup>3</sup> and *nextnano++* tools
- `pn_junction_GaAs_3D_nn3.in` / `*_nnp.in` - input file for the *nextnano*<sup>3</sup> and *nextnano++* tools

These input files are for the same p-n junction structure as in the 1D case, but extended into 2D and 3D.

- 2D: rectangle of dimension 320 nm x 200 nm
- 3D: cuboid of dimension 320 nm x 200 nm x 100 nm

## Complete input file for nextnano++

```
global{
simulate1D{

temperature = 300.0 #_
↪Kelvin

substrate{ name = "GaAs" }

crystal_zb{
    x_hkl = [1, 0, 0]
    y_hkl = [0, 1, 0]
}
}

grid{
#
```

(continues on next page)

(continued from previous page)

```

# For consistency reasons, we use the same nonuniform grid spacing as the
→nextnano3 input file.
# However, using jumps in the grid spacing (e.g. at x=100.0 where the grid
→spacing changes abruptly)
# is not a good practice, as numerical errors increase.
#
xgrid{
  line{ pos = 0.0      spacing = 2.0  }
  line{ pos = 10.0     spacing = 2.0  }
  line{ pos = 10.0     spacing = 1.0  }
  line{ pos = 100.0    spacing = 1.0  }
  line{ pos = 100.0    spacing = 0.5  }
  line{ pos = 140.0    spacing = 0.5  }
  line{ pos = 140.0    spacing = 0.25 }
  line{ pos = 180.0    spacing = 0.25 }
  line{ pos = 180.0    spacing = 0.5  }
  line{ pos = 220.0    spacing = 0.5  }
  line{ pos = 220.0    spacing = 1.0  }
  line{ pos = 310.0    spacing = 1.0  }
  line{ pos = 310.0    spacing = 2.0  }
  line{ pos = 320.0    spacing = 2.0  }
}
}

structure{
output_region_index{ boxes = no }
output_material_index{ boxes = no }
output_alloy_composition{ boxes = no }
output_impurities{ boxes = no }

region{
  everywhere{}
  binary{ name = "GaAs" }
}
region{
  line{
    x = [0.0, 10.0]
  }
  binary{
    name = "GaAs"
  }
  contact { name = source }
}
region{
  line{
    x = [10.0, 310.0]
  }
  binary{
    name = "GaAs"
  }
}
region{
  line{
    x = [310.0, 320.0]
  }
  binary{

```

(continues on next page)

(continued from previous page)

```

        name = "GaAs"
    }
    contact { name = drain }
}

region{
    line{
        x = [ 0.0, 160.0]
        # x = [10.0, 160.0]      # doping must not start at 10.0
    }
    doping{
        constant{
            name = "p-type"
            conc = 0.5e18
        }
    }
}

region{
    line{
        # x = [160.0, 310.0]      # doping must not end at 310.0
        x = [160.0, 320.0]
    }
    doping{
        constant{
            name = "n-type"
            conc = 2.0e18
        }
    }
}

impurities{
    # donor{ name = "n-type" energy = 0.027   degeneracy = 2 }
    # acceptor{ name = "p-type" energy = 0.0058   degeneracy = 4 }
    donor{ name = "n-type" energy = -1000.0 degeneracy = 2 }      # '-
    ↪ 1000.0' eV = all ionized
    acceptor{ name = "p-type" energy = -1000.0 degeneracy = 4 }      # '-1000.0' ↪
    ↪ eV = all ionized
}

contacts{
    ohmic{ name = "source" bias = 0.0 }
    ohmic{ name = "drain" bias = 0.0 }
}

classical{
    Gamma{}
    HH{}
    LH{}
    SO{}

    output_bandedges{ averaged = no}
    output_carrier_densities{}
    output_ionized_dopant_densities{}
    output_intrinsic_density{}

```

(continues on next page)

(continued from previous page)

```
}  
  
poisson{  
  output_potential{}  
  output_electric_field{}  
}  
  
run{  
  solve_poisson{ }  
}
```

**Input Files for *nextnano*<sup>3</sup>:**

- pn\_junction\_GaAs\_1D\_nn3.in
- pn\_junction\_GaAs\_1D\_QM\_nn3.in
- pn\_junction\_GaAs\_2D\_nn3.in
- pn\_junction\_GaAs\_3D\_nn3.in

---

*Last update: nnnn/nn/nn*

### 3.20.12 Quantum-Cascade Lasers

**Input files:**

- *examples\quantum\_cascade\_lasers\1DQuantumCascadeLaser\_nnp.in*
- *examples\quantum\_cascade\_lasers\1DQuantumCascadeLaserSiGe\_nnp.in*
- *examples\quantum\_cascade\_lasers\1DQCL\_AlGaAs\_Sirtori\_APL73\_1998\_nnp.in*
- *examples\quantum\_cascade\_lasers\1DQCL\_Andrea\_Friedrich\_NoInjector\_InGaAs\_APL86\_2005\_kp\_nnp.in*
- *examples\quantum\_cascade\_lasers\1DQCL\_Andrea\_Friedrich\_NoInjector\_InGaAs\_APL86\_2005\_sg\_nnp.in*
- *examples\quantum\_cascade\_lasers\1DQCL\_Rochat\_APL81\_2002\_nnp.in*
- *examples\quantum\_cascade\_lasers\1DQCL\_THz\_MIT\_Sandia\_SemicScTech20\_2005\_nnp.in*
- *examples\quantum\_cascade\_lasers\THzQCL\_Andrews\_Vienna\_MatSciEng2008\_nnp.in*
- *examples\quantum\_cascade\_lasers\THzQCL\_Andrews\_Vienna\_MatSciEng2008\_nnp\_electric\_field.in*
- *examples\quantum\_cascade\_lasers\THzQCL\_Andrews\_Vienna\_MatSciEng2008\_nnp\_no\_repeat.in*

** Note**

If you want to obtain the input files that are used within this tutorial, please check if you can find them in the installation directory. If you cannot find them, please submit a Support Ticket.

**Scope:**

This tutorial aims to simulate different quantum-cascade structures proposed in the literature.

## GaAs/ AlGaAs Quantum-Cascade Laser

This tutorial is based on the quantum-cascade structure that has been presented in [Page2001]. Here, we are trying to reproduce fig. 1 of this paper. The corresponding input file is `IDQuantumCascadeLaser.in`.

The quantum-cascade structure consists of a sequence of *GaAs* wells and *Al<sub>0.45</sub>Ga<sub>0.55</sub>As* barriers. The sequence is as follows (from 0 nm to 45 nm; it is repeated outside this region):

|    | Layer                                       | Thickness [nm] |
|----|---------------------------------------------|----------------|
| 1  | <i>Al<sub>0.45</sub>Ga<sub>0.55</sub>As</i> | 4.6            |
| 2  | <i>GaAs</i>                                 | 1.9            |
| 3  | <i>Al<sub>0.45</sub>Ga<sub>0.55</sub>As</i> | 1.1            |
| 4  | <i>GaAs</i>                                 | 5.4            |
| 5  | <i>Al<sub>0.45</sub>Ga<sub>0.55</sub>As</i> | 1.1            |
| 6  | <i>GaAs</i>                                 | 4.8            |
| 7  | <i>Al<sub>0.45</sub>Ga<sub>0.55</sub>As</i> | 2.8            |
| 8  | <i>GaAs</i>                                 | 3.4            |
| 9  | <i>Al<sub>0.45</sub>Ga<sub>0.55</sub>As</i> | 1.7            |
| 10 | <i>GaAs</i>                                 | 3.0            |
| 11 | <i>Al<sub>0.45</sub>Ga<sub>0.55</sub>As</i> | 1.8            |
| 12 | <i>GaAs</i>                                 | 2.8            |
| 13 | <i>Al<sub>0.45</sub>Ga<sub>0.55</sub>As</i> | 2.0            |
| 14 | <i>GaAs</i>                                 | 3.0            |
| 15 | <i>Al<sub>0.45</sub>Ga<sub>0.55</sub>As</i> | 2.6            |
| 16 | <i>GaAs</i>                                 | 3.0            |

In [Page2001], a conduction band offset of 390 meV was used. Consequently, we modify our default band offset by shifting the AlGaAs ternary to get a 390 meV offset. We also apply an electric field of -48 kV/cm.

```
$ElectricField      = 48e5      # Electric field in units of [V/m] - Here: 48 kV/cm
$ReferencePotential = 0.092     # Set the potential at the leftmost point of the grid.
                             ↪ to the same value as in nextnano3
```

For simplicity, in contrast to [Page2001], we do not include doping here. In the original paper, the areas between 15.2 nm and -5.6 nm (9.8 nm) and 29.8 nm and 39.4 nm (9.8 nm), corresponding to layer 11 - 14, were n-type doped with silicon, with a sheet density of  $n_{\text{Si}} = 3.8 \cdot 10^{11} \text{ cm}^{-2}$ . In this example, we do not have to calculate the strain, because piezo and any pyroelectric fields do not exist. We use single-band (effective-mass) rather than 8-band k.p model.

## Bandedge profile

Figure 3.20.12.1 shows the conduction band energy of the Gamma conduction band edge and the probability densities ( $\Psi^2$ ) of the ground state 4 (red), the lower state 6 (blue), the excited state 10 (pink) and the injector state 8 (green). The above shown structure of the conduction band edge and the wave functions is in excellent agreement with fig. 1 of [Page2001].

Note that periodic boundary conditions for the Schrödinger and Poisson equation do not make sense because of the application of an electric field. Thus, we used Dirichlet boundary conditions. However, this will lead to some artificial, wrong wave functions at the boundaries because the wave function is forced to be zero at the boundaries. For the states in the middle of the device where the wave function decays to zero in any case at the boundaries, the boundary conditions do not have any influence at all and so these states are fine. So the suggestion is to calculate 3 or 5 periods, and then take the energy levels and wave functions of the center period. In this way, boundary effects should not be very severe.

```
global{
  periodic{ x = yes } # apply period boundary conditions along the x-direction
}
```

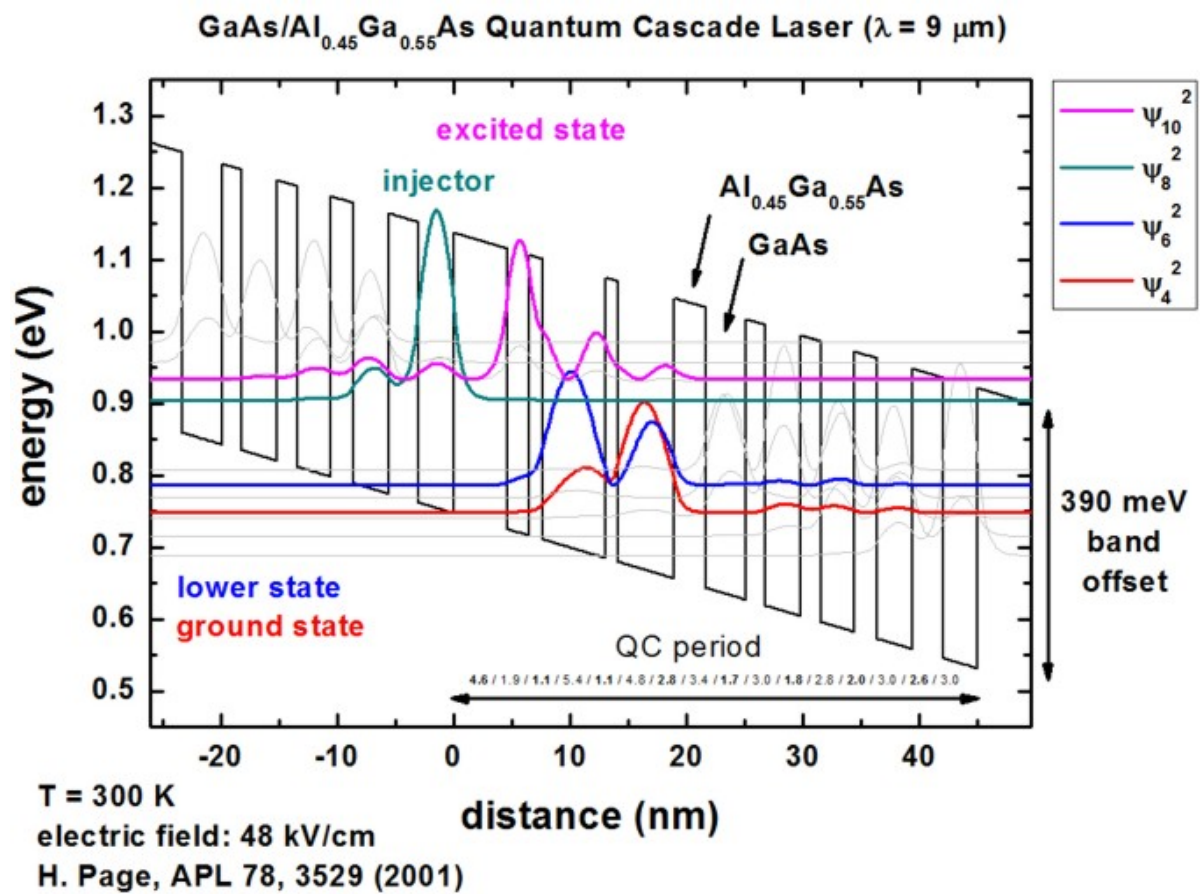


Figure 3.20.12.1: Calculated conduction band edge (black line) of the quantum-cascade structure with electric field of strength 48 kV/cm applied. Also shown are the probability densities ( $\Psi^2$ ) of four electron states, which are shifted by their corresponding eigenenergies.



## Dipole matrix elements

The files:

- *bias\_00000\Quantum\overlap\_integrals\_qr1\_Gamma\_100.txt*
- *bias\_00000\Quantum\dipole\_moment\_matrix\_elements\_qr1\_Gamma\_100.txt*

contain the  $p_x$  and  $z$  matrix elements for all transitions. Our result for the  $z$  matrix element for the transition between the excited state to lower state is in excellent agreement with the result of [Page2001]:

|                                          | <i>nextnano<sup>3</sup></i>          | [Page2001]                 |
|------------------------------------------|--------------------------------------|----------------------------|
| $\langle \Psi_{10}   z   \Psi_6 \rangle$ | $z_{10,6} = 1.6655138016 \text{ nm}$ | $z_{3,2} = 1.7 \text{ nm}$ |
| $\Delta E_{\text{transition}}$           | 147.7 meV                            | 160 meV                    |

## QCL examples

### Note

Please submit a support ticket if you want to obtain the input files for the following structures.

### 1. $\lambda = 9 \text{ }\mu\text{m}$ , i.e. 33 THz or 138 meV

The simulated QCL structure is taken from [Page2001], see Figure 3.20.12.1. The corresponding input is *1DQuantumCascadeLaser.in*.

### 2. $\lambda = 9.4 \text{ }\mu\text{m}$ or 132 meV

The simulated quantum-cascade structure, shown in Figure 3.20.12.2, is based on [Sirtori1998]. The corresponding input file is *1DQCL\_AlGaAs\_Sirtori\_APL73\_1998.in*.

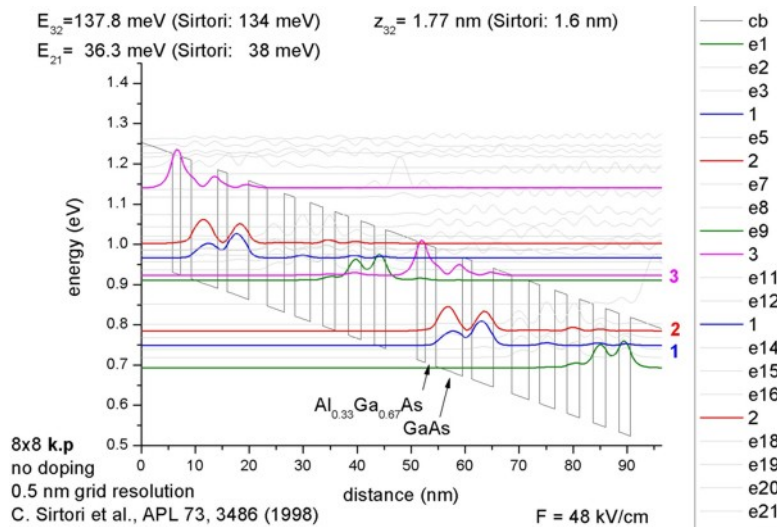


Figure 3.20.12.2: Calculated conduction band edge (black line) of the quantum-cascade structure with electric field of strength 48 kV/cm applied. Also shown are the probability densities ( $\Psi^2$ ) of several electron states, which are shifted by their corresponding eigenenergies.

### 3. $\lambda = 10 \text{ }\mu\text{m}$ or 124 meV (77 K)

The simulated quantum-cascade structure, shown in Figure 3.20.12.3 and Figure 3.20.12.4, is based on [Friedrich2005]. The corresponding input file is *1DQCL\_Andrea\_Friedrich\_NoInjector\_InGaAs\_APL86\_2005\_kp.in*.

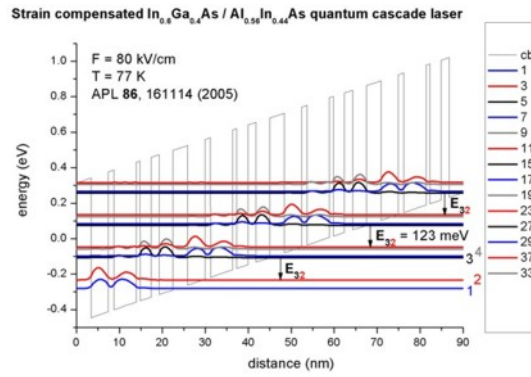


Figure 3.20.12.3: Calculated conduction band edge (black line) of the quantum-cascade structure with electric field of strength 80 kV/cm applied ( $T = 77 \text{ K}$ ). Also shown are the probability densities ( $\Psi^2$ ) of several electron states, which are shifted by their corresponding eigenenergies.

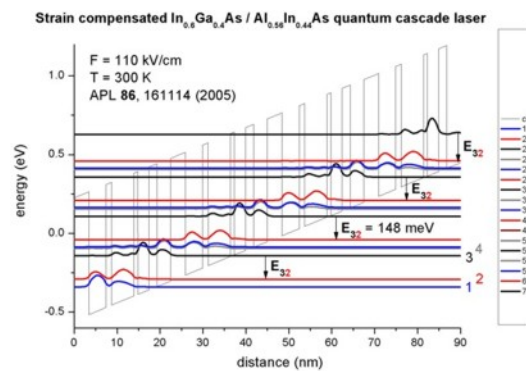


Figure 3.20.12.4: Calculated conduction band edge (black line) of the quantum-cascade structure with electric field of strength 110 kV/cm applied ( $T = 300 \text{ K}$ ). Also shown are the probability densities ( $\Psi^2$ ) of several electron states, which are shifted by their corresponding eigenenergies.

#### 4. $\lambda = 66 \mu\text{m}$ , i.e. 4.54 THz or 18.8 meV

The simulated quantum-cascade structure, shown in Figure 3.20.12.5, is based on [Rochat2002]. The corresponding input file is `1DQCL_Rochat_APL81_2002.in`.

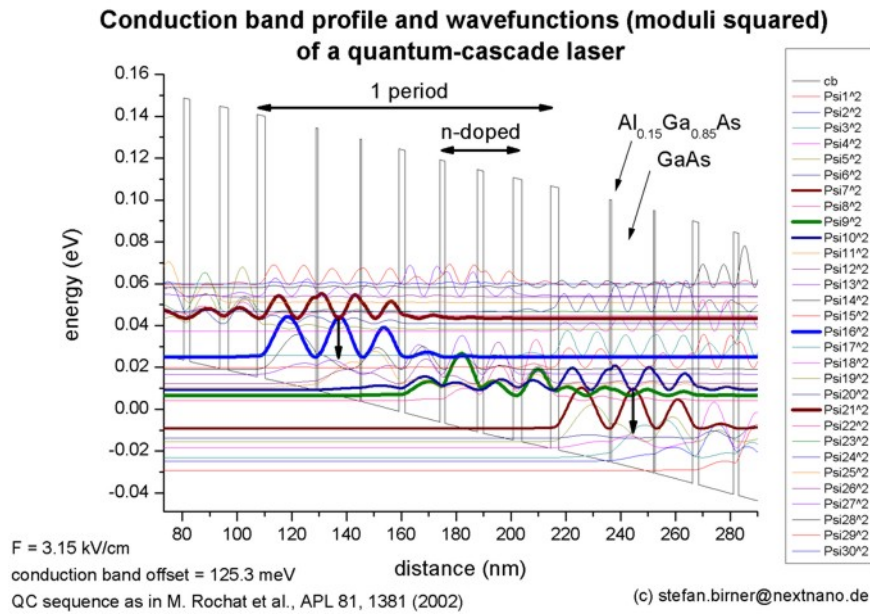


Figure 3.20.12.5: Calculated conduction band edge (black line) of the quantum-cascade structure with electric field of strength 3.15 kV/cm applied. Also shown are the probability densities ( $\Psi^2$ ) of several electron states, which are shifted by their corresponding eigenenergies.

#### 5. $\lambda = 89.2 \mu\text{m}$ , i.e. 3.4 THz or 13.9 meV

The simulated quantum-cascade structure, shown in Figure 3.20.12.6, is based on [Hu2005]. The corresponding input file is `1DQCL_THz_MIT_Sandia_SemicScTech20_2005.in`.

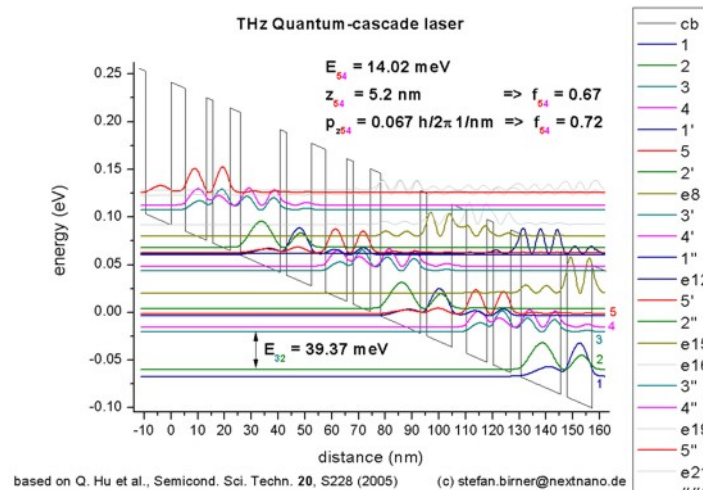


Figure 3.20.12.6: Calculated conduction band edge (black line) of the quantum-cascade structure with electric field of strength 12.2 kV/cm applied. Also shown are the probability densities ( $\Psi^2$ ) of several electron states, which are shifted by their corresponding eigenenergies.

### 6. $\lambda = 107 \mu\text{m}$ , i.e. 2.8 THz or 11 meV

The simulated quantum-cascade structure, shown in Figure 3.20.12.7, is based on [Andrews2008]. The corresponding input file is *THzQCL\_Andrews\_Vienna\_MatSciEng2008\_nnp.in*.

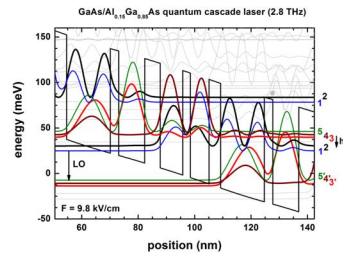


Figure 3.20.12.7: Calculated conduction band edge (black line) of the quantum-cascade structure with electric field of strength 9.8 kV/cm applied. Also shown are the probability densities ( $\Psi^2$ ) of several electron states, which are shifted by their corresponding eigenenergies.

### 7. $\lambda = 9.9 \mu\text{m}$ , i.e. 30.2 THz or 125 meV

The simulated quantum-cascade structure, shown in Figure 3.20.12.8, is based on [Dehlinger2000]. This corresponding input file is *IDQuantumCascadeLaserSiGe\_nnp.in*.

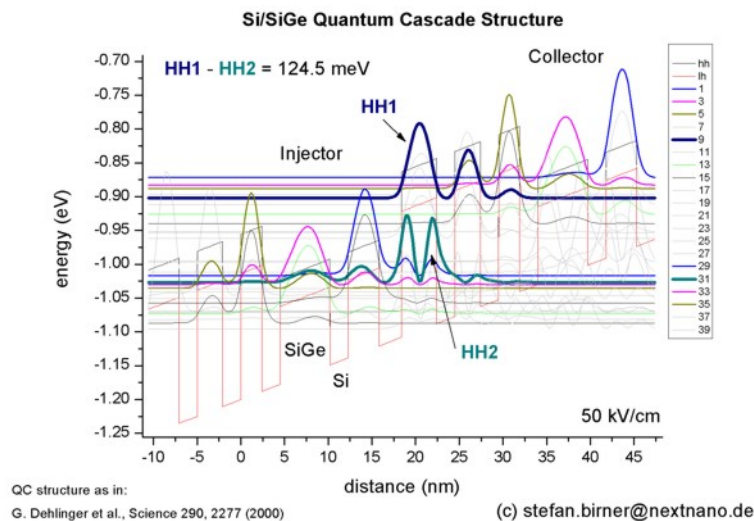


Figure 3.20.12.8: Calculated valance band edge (black line) of the quantum-cascade structure with electric field of strength 50 kV/cm applied. Also shown are the probability densities ( $\Psi^2$ ) of several hole states, which are shifted by their corresponding eigenenergies.

Last update: 27/05/2025

## 3.20.13 Simple quantum cascade structure

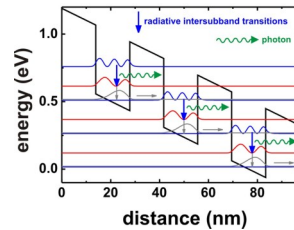
### Input Files:

- *IDQCL\_simple\_nnp.in*

In this tutorial we simulate a simple quantum cascade structure that has been presented in an article by Capasso et al. (Figures 12 (b) and 16 (b) of [CapassoIEEE1986]).

We can generate the following picture that is based on Fig. 3 of [BirnerPhotonikInt2008] and [BirnerPhotonik2008].

It shows the conduction band edge profile of an  $\text{Al}_{0.48}\text{In}_{0.52}\text{As}/\text{In}_{0.53}\text{Ga}_{0.47}\text{As}$  superlattice at an electric field of -89 kV/cm. The single-band effective-mass Schrödinger equation is solved for this band profile. The wave functions ( $\psi^2$ ) of this quantum cascade structure are shown.



The basic idea of such a structure is to depopulate the lowest eigenstate of each quantum well efficiently by bringing it into resonance with the third eigenstate of the next quantum well (resonant tunneling).

The transition second eigenstate  $\rightarrow$  lowest eigenstate should be a nonradiative intersubband transition.

On the other hand, the transition third eigenstate  $\rightarrow$  second eigenstate should be a radiative intersubband transition, i.e. a photon is emitted.

Another important condition for a quantum cascade laser is **population inversion**, i.e. the occupation of the third eigenstate must be much higher than the occupation of the second eigenstate and lowest eigenstate.

- The input file *IDQCL\_simple\_nnp/\*nn3.in* should be rather intuitive and self-explanatory.
- In the *nextnano++* sample file, the electric field is applied by specifying the keyword `contacts` as follows:

```
contacts{
  charge_neutral{
    name = "leftgate"
    bias = 0.0
  }
  charge_neutral{
    name = "rightgate"
    bias = 1.36081          # corresponds to electric field of F = -89
    ↪ kV/cm
  }
}
```

In the keyword `structure`, “leftgate” is defined at  $x = [-1, 0]$  and “rightgate” is at  $x = [152.9, 153.9]$ . Thus the electric field applied by this specification is  $-1.36081 \text{ [V]} / 152.9 \text{ [nm]} = -89 \text{ [kV/cm]}$

- Alternatively, we can apply a constant electric field by providing a value for the field.

```
poisson{
  electric_field{ strength = -89e5 } # [V/m]
  output_potential{}
  output_electric_field{}
}
```

## Output

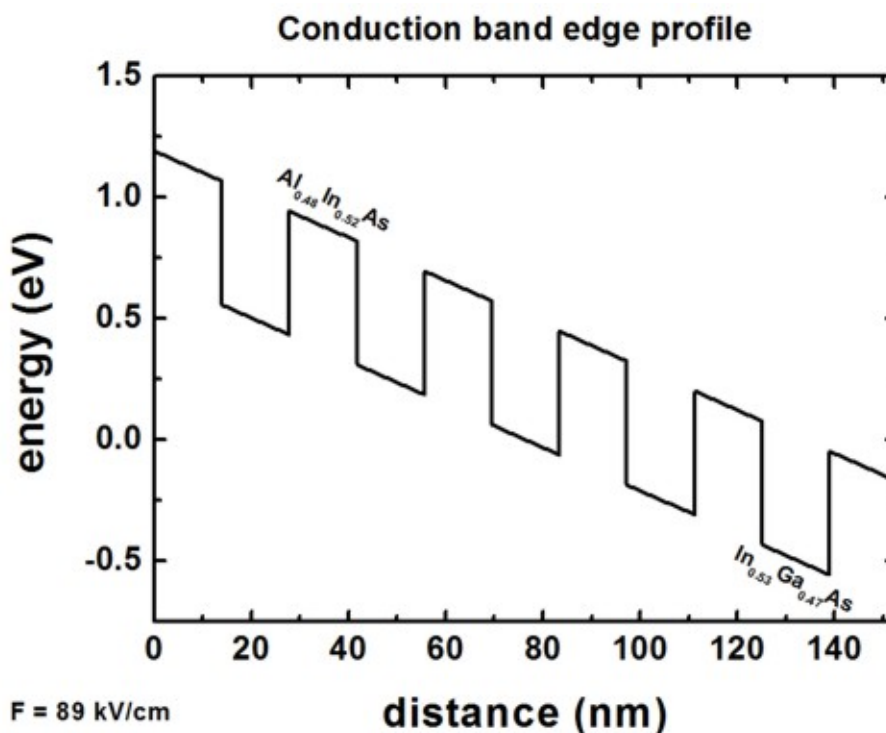
The output files are ASCII files.

## Bandedges

The conduction and valence band edges can be found in the following file:

- *bias\_0000/Quantum/bandedges.dat* (nextnano++)
- *band\_profile/cb\_Gamma.dat* (nextnano<sup>3</sup>, conduction band edge only)

If one plots the conduction band profile, one gets the following figure.



There are six  $\text{Al}_{0.48}\text{In}_{0.52}\text{As}$  barriers and five  $\text{In}_{0.53}\text{Ga}_{0.47}\text{As}$  wells. The conduction band offset is 0.51 eV.

## Eigenvalues

The 40 eigenvalues that were calculated can be found in these files. The units are [eV].

- *bias\_0000/Quantum/wf\_energy\_spectrum\_quantum\_region\_Gamma\_0000.dat* (nextnano++)
- *wavefunctions/ev\_cb1\_sg1\_deg1.dat* (nextnano<sup>3</sup>)

The eigenvalues are also contained in these files, i.e. the eigenvalues for each grid point

- *bias\_0000/Quantum/wf\_probabilities\_shift\_quantum\_region\_Gamma\_0000.dat* (nextnano++)
- *wavefunctions/cb1\_sg1\_deg1\_psi\_squared\_shift.dat* (nextnano<sup>3</sup>)

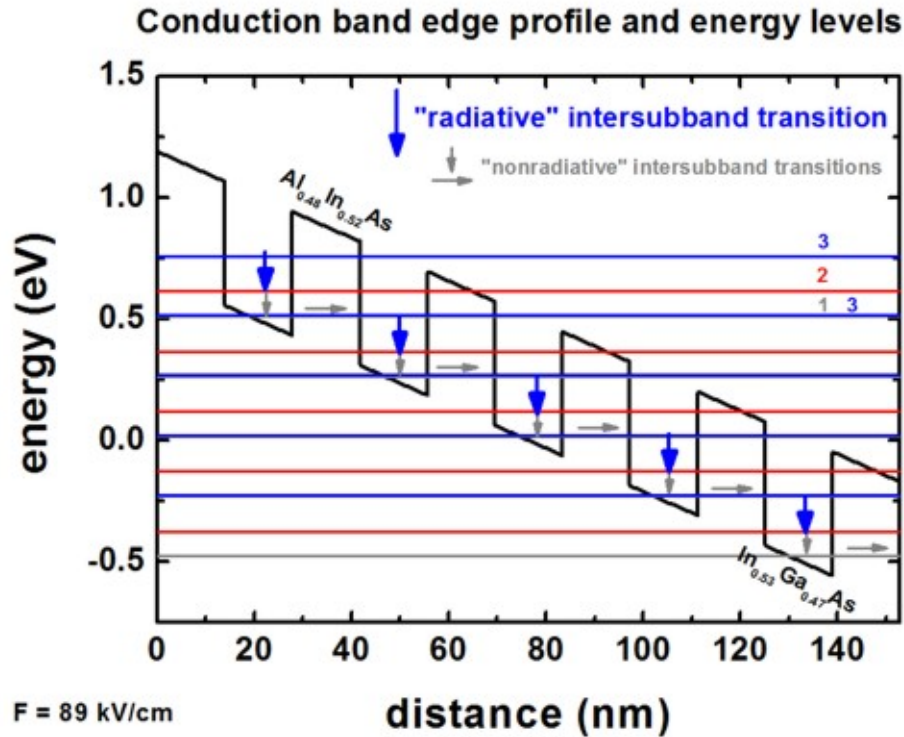
| 1st column                   | 2nd column                      | 3rd column                      | ... | 41st column                      |
|------------------------------|---------------------------------|---------------------------------|-----|----------------------------------|
| grid points in units of [nm] | 1st eigenvalue in units of [eV] | 2nd eigenvalue in units of [eV] | ... | 40th eigenvalue in units of [eV] |

If one plots these columns (together with the conduction band edge) one obtains the following picture:

### Note

The figure shows only the following energy levels: 1,2,3,4,5,9,10,12,16,18,20,26,27,30,37





### Wave Functions

The square of the wave functions ( $\psi^2$ ) of the 40 eigenstates can be found in these files.

- *bias\_0000/Quantum/wf\_probabilities\_shift\_quantum\_region\_Gamma\_0000.dat* (nextnano++)
- *wavefunctions/cb1\_qc1\_sg1\_deg1\_psi\_squared\_shift.dat* (nextnano<sup>3</sup>)

|                              |     |                            |                            |     |                             |
|------------------------------|-----|----------------------------|----------------------------|-----|-----------------------------|
| 1st column                   | ... | 42nd column                | 43rd column                | ... | 81st column                 |
| grid points in units of [nm] | ... | $\psi^2$ of 1st eigenstate | $\psi^2$ of 2nd eigenstate | ... | $\psi^2$ of 40st eigenstate |

#### **Note**

In order to be able to plot the wave functions nicely into the conduction band edge profile, we shift the square of the wave function by its corresponding energy.

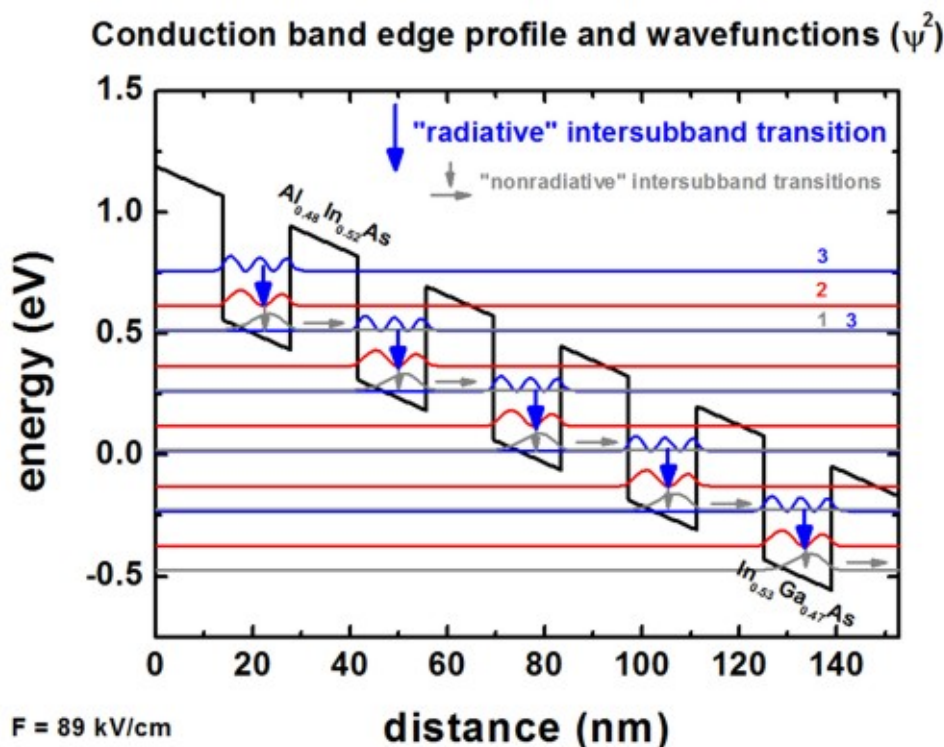
If one plots these columns (together with the conduction band edge) one obtains the following picture:

#### **Note**

The figure shows only the following wave functions: 1,2,3,4,5,9,10,12,16,18,20,26,27,30,37

Now the lowest eigenstate of each quantum well is in resonance with the third eigenstate of the next quantum well. This leads to the depopulation of the lowest eigenstate of each quantum well.

Photon should be emitted with the radiative intersubband transition  $3 \rightarrow 2$  whereas  $2 \rightarrow 1$  should be nonradiative intersubband transition.



### Effective masses

The effective masses that were used for each grid point can be found in these files.

- *Structure/charge\_carrier\_masses.dat* (nextnano++)
- *material\_parameters/conduction\_band\_masses.dat* (nextnano<sup>3</sup>)

#### Note

We need to add the following option into the sample file for nextnano++.

```
output{
  material_parameters{
    charge_carrier_masses{ boxes = yes }
  }
}
```

- 1st column: grid points in units of [nm]
- other columns:
  - nextnano++: effective mass tensor components of Gamma and HH valley in units of [m<sub>0</sub>]. When we use other valleys for the simulation, then these columns shows the effective mass tensor components in that valleys.
  - nextnano<sup>3</sup>: effective mass tensor components of Gamma, L and X valleys in units of [m<sub>0</sub>].

These masses have been calculated from the binaries InAs, GaAs and AlAs for the relevant ternaries, including bowing parameters.



## Intersubband matrix elements

Experienced users might be interested in having a look at the intersubband matrix elements.

We can find the intersubband (or intraband) matrix elements  $p_z$ , the oscillator strengths and the transition energies by adding the followings into `quantum{ }` in `IDQCL_simple_nnp.in`:

```
intraband_matrix_elems{
  Gamma{}
  output_matrix_elements = yes
  output_transition_energies = yes
  output_oscillator_strengths = yes
}
```

The relevant output files are

- `bias_0000/Quantum/momentum_matrix_elements_quantum_region_Gamma_100.txt` (nextnano++)
- `bias_0000/Quantum/transition_energies_quantum_region_Gamma_Gamma.txt` (nextnano++)

In the output file of the nextnano<sup>3</sup> sample file, we can already have them here:

- `wavefunctions/intraband_pz_cb1_sg1_deg1.txt` (nextnano<sup>3</sup>)

### Input Files for nextnano<sup>3</sup>:

- `IDQCL_simple_nn3.in`

Last update: 27/05/2025

## 3.20.14 InAs / In<sub>0.4</sub>Ga<sub>0.6</sub>Sb superlattice dispersion with 8-band k.p (type-II band alignment)

Authors: Stefan Birner, Michael Povolotskyi

### Input Files:

- `T2SL_InAs-GaInSb_Grein_JAP_1995_ID_nnp`

This tutorial aims to reproduce Fig. 2(a) of “*Long wavelength InAs/InGaSb infrared detectors: Optimization of carrier lifetimes*” by Grein and Young.

### Conduction and valence band edges

The heterostructure is a **superlattice** with 3.98 nm InAs and 1.5 nm In<sub>0.4</sub>Ga<sub>0.6</sub>Sb, where both constituents are strained with respect to the GaSb substrate.

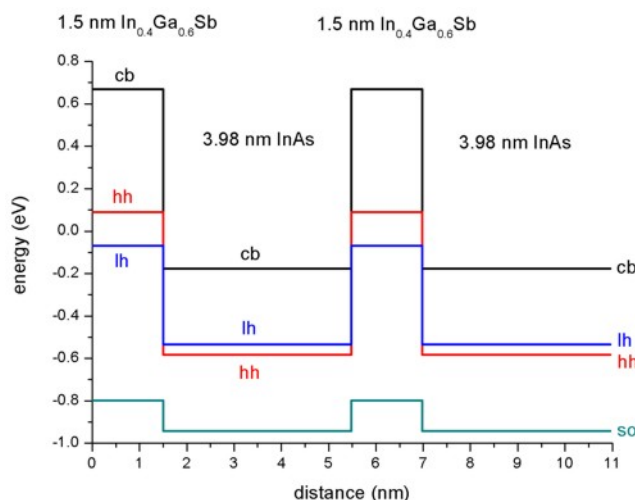
The structure has a **type-II** band alignment, i.e. the electrons are confined in the InAs layer, whereas the holes are confined in the In<sub>0.4</sub>Ga<sub>0.6</sub>Sb layer.

The In<sub>0.4</sub>Ga<sub>0.6</sub>Sb layer is strained pseudomorphically with respect to the GaSb substrate, leading to a **compressive** strain (-2.5%) which splits the degeneracy of the heavy and light hole band edges in this layer. Thus, the heavy hole band edge lies above the light hole band edge.

The InAs layer is also strained pseudomorphically with respect to the GaSb substrate, and is thus under slight biaxial **tension** (+0.6 %). The splitting of the hole band edges is the opposite as in InGaSb, i.e. the light hole band edge is above the heavy hole band edge.

The following figure shows the electron and hole band edges.

Note that the origin of the energy scale is set to the GaSb valence band edge energy.

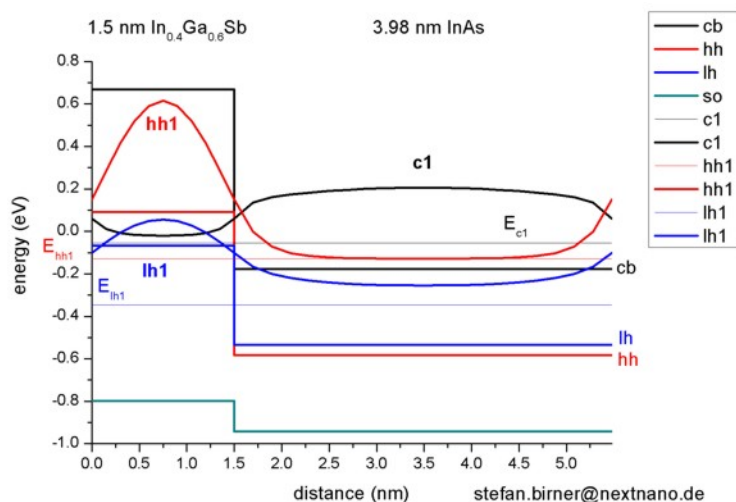


### Electron and hole wave function for $k_{||} = 0$

We simulate **one period only** (i.e. from 0 nm to 5.48 nm) and solve the Schrödinger equation with periodic boundary conditions to mimic an infinite superlattice.

The following figure shows the conduction band edge and the heavy, light and split-off hole valence band edges in this superlattice structure together with the electron (**c1**), heavy hole (hh1) and light hole (lh1) energies and wave functions ( $\psi^2$ ), calculated within 8-band k.p theory.

One can clearly see that the electron state (**c1**) is confined in the InAs layer (right part of the figure), whereas the heavy (hh1) and light hole (lh1) states are confined in the In<sub>0.4</sub>Ga<sub>0.6</sub>Sb layer (left part of the figure).

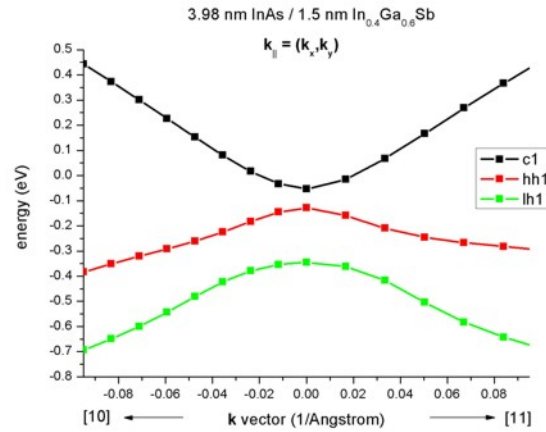


We used the same material parameters as given in the above cited paper by Grein *et al.*, apart from the  $\mathbf{k} \cdot \mathbf{p}$  parameters.

### Electron and hole energies for $k_{||} \neq 0$

The following figure shows the  $E(\mathbf{k}_{||})$  dispersion of the electron ground state and the two highest hole states along two different directions in  $(k_x, k_y)$  space.

This data is contained in this file: `Schroedinger_kp/par1D_disp_01_00_11_hl_8x8kp_ev_min001_ev_max010.dat`. Note that the band gap is not determined by the band gap of one individual layer. It is determined by the electron ground state in the InAs layer, and the hole ground state in the InGaSb layer. This means more freedom for band gap engineering.



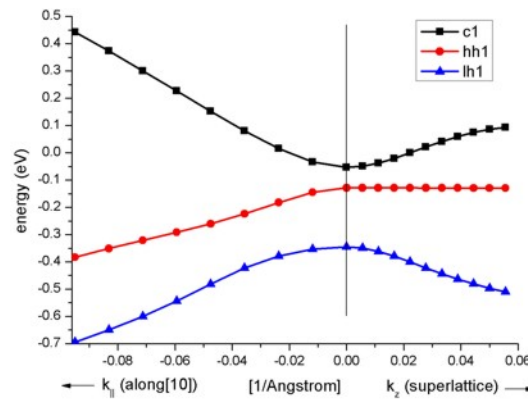
### Electron and hole energies for $k_z \neq 0$

The input file used is `1DInAs_InGaSb_k_superlattice_nnp.in`

The right part of the following figure shows the  $E(k_z)$  superlattice dispersion of the electron ground state and the two highest hole states.  $k_z$  is the superlattice vector between 0 and  $1 \frac{\pi}{L}$  where  $L = 5.48$  nm is the length of one superlattice period. ( $1 \frac{\pi}{L} = 0.05731/\text{\AA}$ )

This data is contained in this file: `Schroedinger_kp/8x8kp_dispSL_hl_qc001_evmin001_evmax016.dat`

The left part of the figure shows the  $E(k_{||})$  dispersion along [10], i.e. from  $(k_x, k_y) = (0,0)$  to  $(k_x, k_y) = (-0.1,0)$  which is shown in the figure above already.



One can clearly see that these heterostructure bands are highly nonparabolic.

### nextnano3

Using `1DInAs_InGaSb_SL_k_parallel_superlattice.in`, one can generate a 3D plot of the energy dispersion  $E(k_x, k_y, k_{SL})$  for each eigenvalue. The files are called `dispersion_k_parallel_k_SL_ev001.fld` where `ev001` indicates eigenvalue number 1.

Last update: nnnn/nn/nn

## 3.20.15 Triangular well

In this tutorial we calculate the Schrödinger equation for a triangular well and compare the results with the analytic solution.

The related input files are followings:

- `1DGaAs_triangular_well_nn*.in`

## Structure

A triangular well consists of a potential with a constant slope that is bound at one side by an infinite barrier.

For  $x < 0$  nm we have an infinite barrier. In our case it is represented by a huge conduction band offset of 100 eV to avoid any wave function penetration into the barrier.

For  $x > 0$  nm we have a linear potential of  $V(x) = eFx$ .

$V(x)$  describes a charge  $e$  in an electric field  $F$  where the product  $eF$  is assumed to be positive.

## Comparison of nextnano++ and the analytic solution

The Schrödinger equation for the transverse component of the electronic wave function has the following form inside the well:

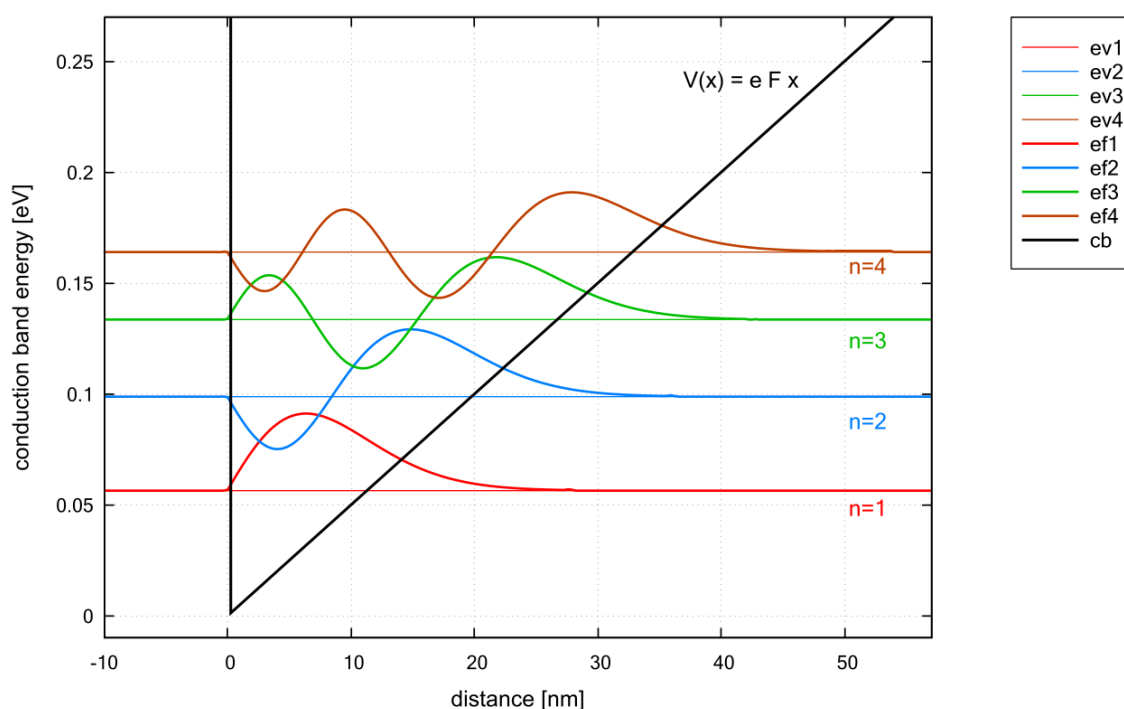
$$\left[ -\frac{\hbar^2}{2m^*} \frac{d^2}{dx^2} + eFx \right] \psi(x) = E\psi(x)$$

Usually one applies Dirichlet boundary conditions at  $x = 0$  nm so that  $\psi(x = 0) = 0$  in order to represent an infinite barrier, i.e. the high barrier prevents significant penetration of electrons into the barrier region.

In our case, we apply Neumann (or Dirichlet) boundary conditions at  $x = -10$  nm and  $x = 150$  nm and let the infinite barrier be represented by the huge conduction band offset of 100 eV. Then, both boundary conditions lead to the same eigenenergies for the lowest eigenstates.

The Schrödinger equation can be simplified by introducing suitable new variables and thus reduces to the Stokes or Airy equation. Its solutions, the so-called Airy functions, are discussed in most textbooks, see for example:

- *The Physics of Low-Dimensional Semiconductors - An Introduction*, John H. Davies, Cambridge University Press (1998)



The figure shows the conduction band edge (black line) which is represented by a triangular potential well  $V(x) = eFx$ . Also shown are the four lowest energy levels and corresponding wave functions. The electric field that has been applied is  $F = 5$  [MV/m], i.e.  $0.05\text{V}/10$  nm. The effective electron mass has the value  $0.067m_0$  (GaAs).

One can see that the distance between the energy levels decreases with increasing  $n$  because the quantum well width gets larger for higher energies. Note that in a parabolic well, the energy levels are equally spaced whereas in an infinitely deep square well, the energy level separation increases with increasing energy.

The eigenvalues of the Airy equation can be calculated using the formula:

$$E_n = c_n \left[ \frac{eF\hbar^2}{2m^*} \right]^{1/3}$$

(The units of  $E_n$  in this equation are [J].)

The lowest eigenvalue has the value  $c_1 = 2.338$ .

For large  $n$ ,  $c_n$  can be approximated by the following equation which can be derived from WKB theory (named after Wentzel, Kramers and Brillouin):

$$c_n \simeq \left[ \frac{3}{2}\pi \left( n - \frac{1}{4} \right) \right]^{2/3}$$

The *nextnano++* and *nextnano<sup>3</sup>* eigenvalues for the lowest four eigenstates are in very good agreement with the analytic results:

|       | <i>nextnano++</i><br>eigen-<br>value | <i>nextnano<sup>3</sup></i><br>eigen-<br>value | calculated<br>value | eigen-<br>$c_n$ (exact) | $c_n$ (approx-<br>imated) |
|-------|--------------------------------------|------------------------------------------------|---------------------|-------------------------|---------------------------|
| n = 1 | 0.05647                              | 0.05644                                        | 0.05664             | $c_1 = 2.338$           | $c_1 = 2.320251$          |
| n = 2 | 0.09887                              | 0.09882                                        | 0.09889             | $c_2$                   | $c_2 = 4.081810$          |
| n = 3 | 0.13358                              | 0.13351                                        | 0.13365             | $c_3$                   | $c_3 = 5.517164$          |
| n = 4 | 0.16426                              | 0.16416                                        | 0.16435             | $c_4$                   | $c_4 = 6.784455$          |

The triangular potential is not symmetric in  $x$ , thus the wave functions lack the even or odd symmetry that one obtains for the infinitely deep square well.

The triangular well model is useful because it can be used to approximate the (idealized) triangularlike shape near a heterojunction formed by the discontinuity of the conduction band and an electrostatic field of electrons or remote ionized impurities.

Last update: nnnn/nn/nn

### 3.20.16 — DEV — Efficient method for the calculation of ballistic quantum transport - The CBR method (2D example)

#### Attention

This tutorial is under construction

- [Header](#)
- [Introduction](#)
- [Simulation setup](#)
- [Transmission](#)
- [Lead modes](#)
- [Notes](#)

## Header

### Input Files:

- *Transmission\_CBR\_Mamaluy\_JAP\_2003\_2D\_nnp.in*
- *Transmission\_CBR\_Mamaluy\_JAP\_2003\_2D\_holes\_nnp.in*

### Scope of the tutorial:

- 

### Main adjustable parameters in the input file:

- parameter

### Relevant output files:

- *bias\_00000\bandedges.fld*
- *bias\_00000\Quantum\probabilities\_shift\_device\_Gamma.fld*
- *bias\_00000\Quantum\probabilities\_shift\_lead\_X\_Gamma.dat*
- *bias\_00000\CBR\transmission\_device\_Gamma.dat*

## Introduction

In this tutorial, we apply the Contact Block Reduction (CBR) method to a Aharonov-Bohm-type structure with a large barrier in the middle of the device. This tutorial is based on [MamaluyCBR2003] and [BirnerCBR2009]. The input file *Transmission\_CBR\_Mamaluy\_JAP\_2003\_2D\_holes\_nnp.in* simulates holes instead of electrons.

## Simulation setup

First, we look into the structure of the device. Figure 3.20.16.1 shows the calculated conduction band edge of the device.

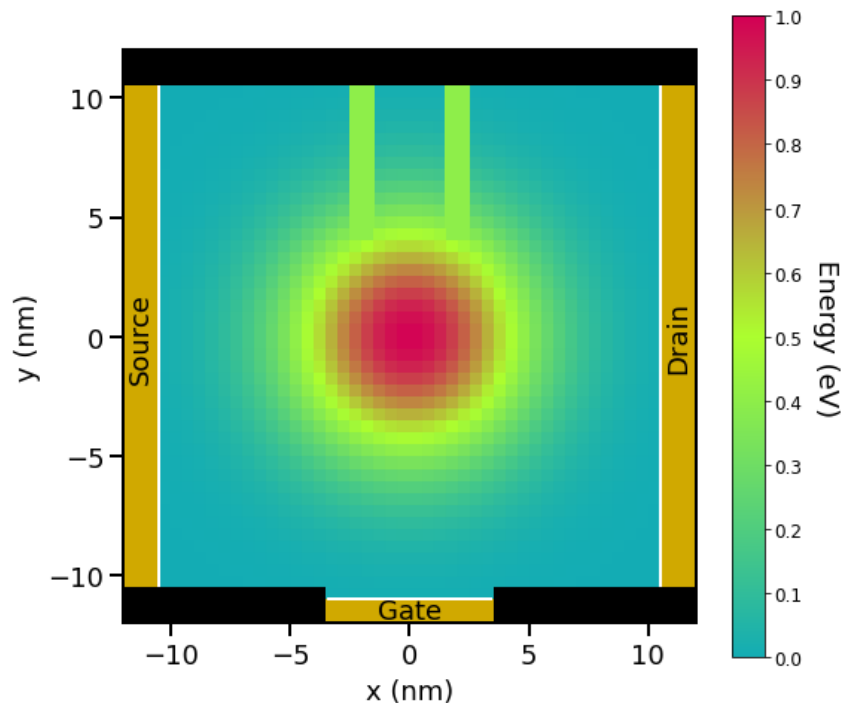


Figure 3.20.16.1: The calculated conduction band edge. The center of the device ( $(x, y) = (0, 0)$  (nm)) is AlAs and the energy is 1.0 (eV). The vicinity of the edges of the device is GaAs and the energy is 0 (eV). The double potential barrier is set so that the energy is equivalent to 0.4 (eV). Note that the blacked out areas are set up with barriers of infinite height. *bias\_00000\bandedges.fld*

The image below shows the 3-dimensional conduction band edge. Note that the height of the infinite potential barriers are set to 2.0 (eV) for convenience.

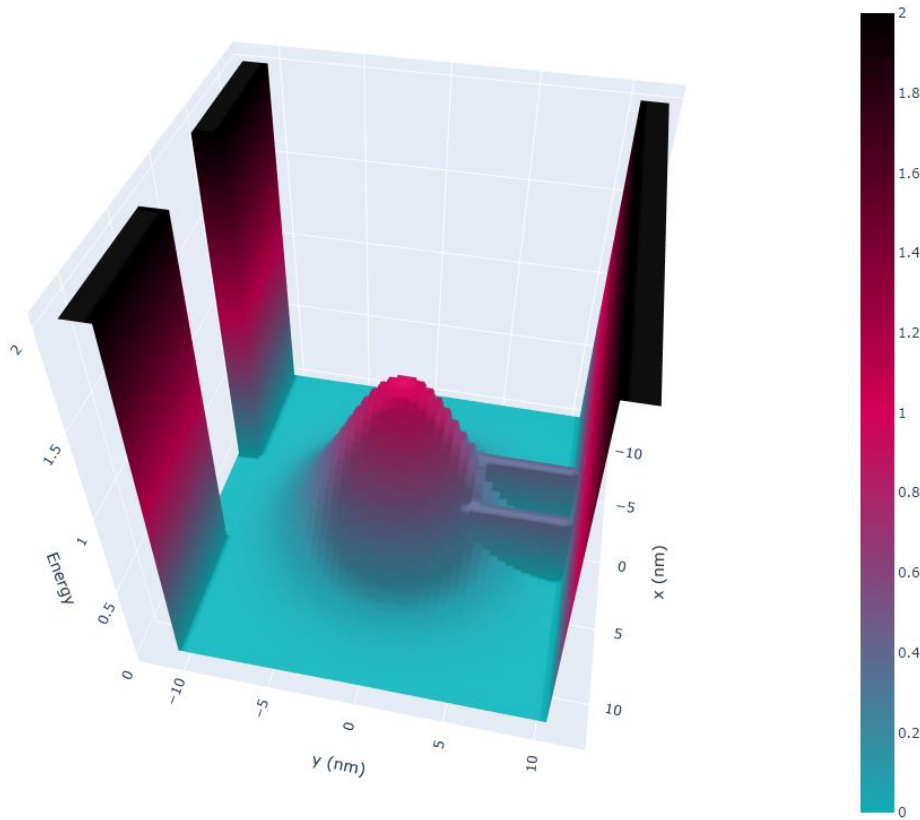


Figure 3.20.16.2: Potential landscape

This device has some features.

The device consists of three contacts that are called ‘source’, ‘gate’ and ‘drain’. They also have leads adjacent to them, indicated by white lines in [Figure 3.20.16.1](#)

In the middle of the device a potential barrier of two-dimensional Gaussian shape effectively expels the electrons from the center. The energy profile is given by

$$E_c = E_{c,0} \exp\left(-\frac{x^2 + y^2}{a^2}\right),$$

where  $E_{c,0} = 1.0$  (eV) so that the maximum height of the Gaussian barrier becomes 1.0 (eV) at the center of the device. In this tutorial,  $a = 5$  (nm).

In the upper part of the device, a thin tunneling double barrier is present and the height is 0.4 (eV).

These conduction band profiles are achieved by adjusting the `database{ }` as below.

```
database{
  binary_zb{
    name = "GaAs"
    conduction_bands{ Gamma{ mass = 0.3 bandgap = 0 } # effective mass 0.3m0
    valence_bands{
      bandoffset = 0.0 # artificially shifted so that (GaAs conduction_
↪bandedge) = 0.0 eV
      delta_S0 = 0.0
    }
  }
}
```

(continues on next page)

(continued from previous page)

```

binary_zb{
    name = "AlAs"
    conduction_bands{ Gamma{ mass = 0.3 bandgap = 0 } # effective mass 0.3m0
    valence_bands{
        bandoffset = 1.00 # artificially shifted so that (AlAs conduction_
↪bandedge) = 1.0 eV

        delta_SO = 0.0
    }
}

bowing_zb{
    name = "AlGaAs_Bowing_x"
    valence = III_V
    conduction_bands{ Gamma{ mass = 0.0 bandgap = 0.000 } } # bowing is_
↪switched off for this simulation
    valence_bands{
        bandoffset = 0.000 # artificially shifted so that (Al0.4Ga0.6As_
↪conduction bandedge) = 0.4 eV
        delta_SO = 0
    }
}

bowing_zb{
    name = "AlGaAs_Bowing_1_x"
    valence = III_V
    conduction_bands{ Gamma{ mass = 0.0 bandgap = 0.000 } } # bowing is_
↪switched off for this simulation
    valence_bands{
        bandoffset = 0.000 # artificially shifted so that (Al0.4Ga0.6As_
↪conduction bandedge) = 0.4 eV
        delta_SO = 0
    }
}

ternary2_zb {
    name      = "Al(x)Ga(1-x)As"
    valence    = III_V
    binary_x   = AlAs
    binary_1_x = GaAs
    bowing_x   = AlGaAs_Bowing_x
    bowing_1_x = AlGaAs_Bowing_1_x
}
}

```

In addition, the infinite potential barriers surround the device as shown as blacked out areas in Figure 3.20.16.1.

The effective electron mass is assumed to be constant throughout the device and equal to  $0.3m_0$ .

We set the boundary conditions as follows:

- If it is at the boundary, and if it is in contact to a lead, a **Neumann** boundary condition is set.
- If it is at the boundary, and if it is **not** in contact to a lead, a **Dirichlet** boundary condition is set.

```

quantum{
    region{

```

(continues on next page)



(continued from previous page)

```

    name          = "device"
    no_density    = yes
    x             = [ $x_contact_left, $x_contact_right ]
    y             = [ $y_contact_bottom, $y_quantum_top ]
    boundary{ x = neumann y = neumann } # boundary condition for CBR = Neumann,
    ↪for propagation direction & Dirichlet for perpendicular direction.
    Gamma{ num_ev = $num_eigenstates_device cutoff = 4.0 }
    output_wavefunctions{
        probabilities = yes
        max_num       = $num_output_wavefunctions_device
        in_one_file    = no
    }
}

# lead 1 is a 1D line (x = $contact_left).
region{
    name          = "lead_1"
    no_density    = yes
    x             = [ $x_contact_left, $x_contact_left ]
    y             = [ $y_inf_barrier_bottom, $y_inf_barrier_top ]
    boundary{ x = neumann y = dirichlet }
    Gamma{ num_ev = $num_eigenstates_lead1 cutoff = 4.0 }
    output_wavefunctions{ probabilities = yes max_num = $num_eigenstates_lead1 }
}

# lead 2 is a 1D line (y = $y_contact_bottom).
region{
    name          = "lead_2"
    no_density    = yes
    x             = [ $bottom_contact_left, $bottom_contact_right ]
    y             = [ $y_contact_bottom, $y_contact_bottom ]
    boundary{ x = dirichlet y = neumann }
    Gamma{ num_ev = $num_eigenstates_lead2 cutoff = 4.0 }
    output_wavefunctions{ probabilities = yes max_num = $num_eigenstates_lead2 }
}

# lead 3 is a 1D line (x = $x_contact_right).
region{
    name          = "lead_3"
    no_density    = yes
    x             = [ $x_contact_right, $x_contact_right ]
    y             = [ $y_inf_barrier_bottom, $y_inf_barrier_top ]
    boundary{ x = neumann y = dirichlet }
    Gamma{ num_ev = $num_eigenstates_lead3 cutoff = 4.0 }
    output_wavefunctions{ probabilities = yes max_num = $num_eigenstates_lead3 }
}

cbr{
    name = "device"
    lead{ name = "lead_1" }
    lead{ name = "lead_2" }
    lead{ name = "lead_3" }
    delta_energy = 0.0005 # energy resolution
    min_energy   = 0.0    # minimum energy
    max_energy   = 0.5    # maximum energy
}

```

(continues on next page)

}

Note the following points.

- To consistent with the results of [MamaluyCBR2003] and [BirnerCBR2009], the quantum region is extended (1 grid point outside along x and y direction), respect to the device dimensions in the papers.

This is attributed to the difference in the way boundary conditions are set in *nextnano++* and *nextnano*<sup>3</sup>. The details are described below in attention.

- To set dirichlet boundary conditions at the top and bottom of the device that are no contact with leads, the quantum region is extended to the infinite potential barrier (1 grid point **further** outside along y direction), respect to the device dimensions in the papers.

The difference in the device dimensions from in [MamaluyCBR2003] and [BirnerCBR2009] arise from the reasons above.

For each energy  $E$  (energy step is equal to 0.0005) where the transmission coefficient  $T(E)$  has to be calculated, a matrix of size  $95 \times 95$  has to be inverted. The size of 95 is determined by the sum of the number of grid points in each lead that are in contact to the device.

- Lead 1 (Source): 41 grid points
- Lead 2 (Gate): 13 grid points
- Lead 3 (Drain): 41 grid points
  - in total: 95 grid points
  - The total CPU time for calculation of the transmission  $T(E)$  in this example is about 5 seconds for 303 eigenstates.

Note that we do not take into account the increase in grid points due to the increase in the gate length.

## Transmission

Figure 3.20.16.3 shows the calculated transmission coefficients of the various lead combinations  $T_{12}$ ,  $T_{23}$ , and  $T_{13}$ . For the orange-dashed lines 100 % (1681 of 1681) of all eigenvectors were used whereas for the light-blue lines only 18 % (303 of 1681) had to be calculated. You can see that reducing the eigenvectors to 18 % or even 7 % (118 of 1681) of the total eigenvectors does not result in significant changes in  $T(E)$ , especially at lower energies. This means that one does not have to calculate all eigenvalues of the device Hamiltonian which grossly reduces CPU time. A small percentage of eigenvalues suffices for  $T(E)$  in relevant energy range of interest.

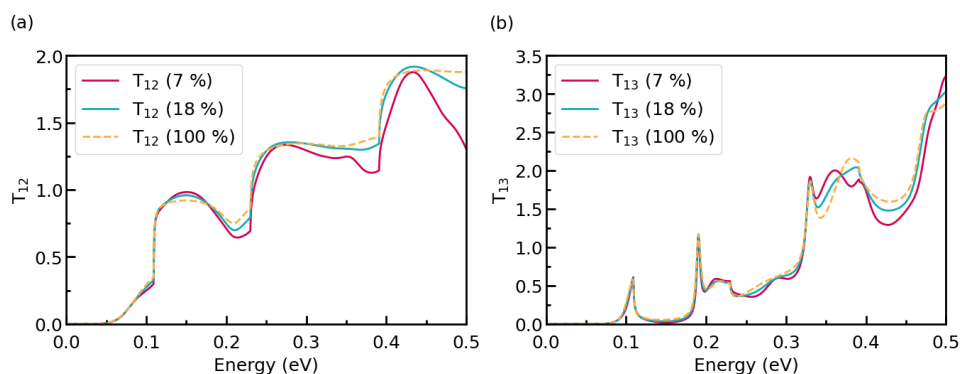


Figure 3.20.16.3: The transmission coefficient  $T(E)$  of a 2D sample with 3 leads.  $T_{12}$  in (a), whereas  $T_{13}$  in (b). *bias\_00000\CBRtransmission\_device\_Gamma.dat*

The *nextnano++* results differ slightly from the [MamaluyCBR2003] and [BirnerCBR2009].

**Reasons:**

- The potential energy profile in the device and in the leads is not identical, as well as the dimensions of the barriers.
- The dimensions of the device are not identical as explained (See the attention below for further information).

Therefore, the eigenenergies and the wave functions in the device, and in the leads differ slightly which explains the small deviations.

The 16th eigenstate is a resonance state of the lower transmission path.

- 1st resonance: the 16th eigenstate: 0.119 (eV)

The square of the 16th wave function with the conduction band is shown below. (*bias\_00000\Quantum\probabilities\_shift\_device\_Gamma.fld*)

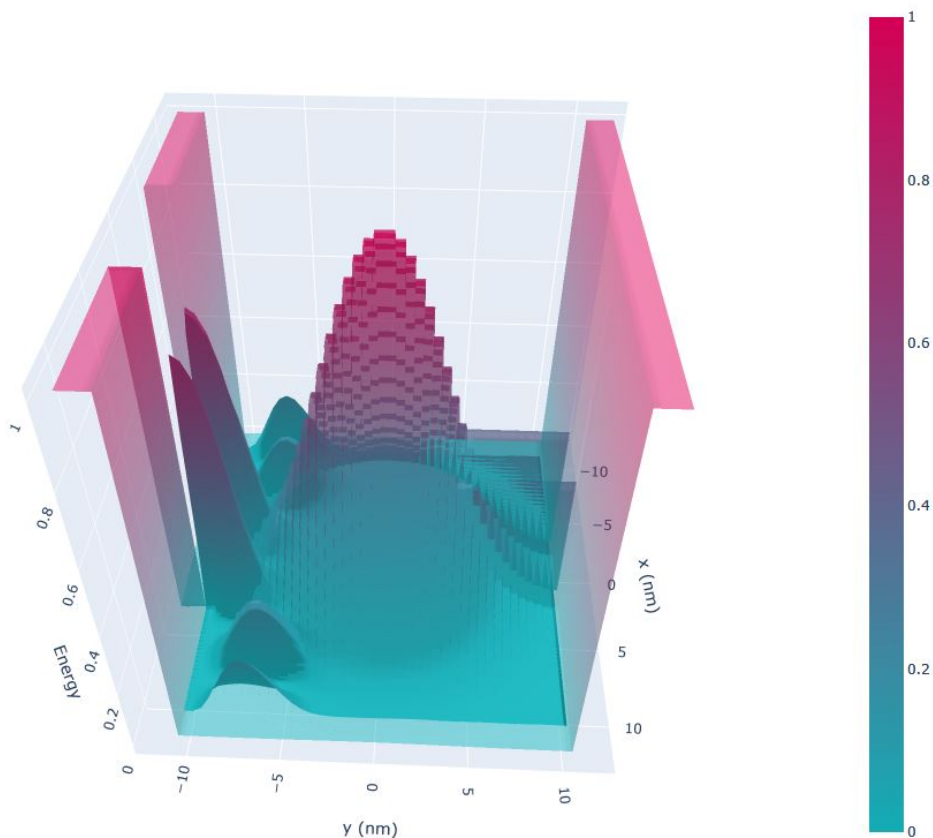


Figure 3.20.16.4: The 16th eigenstate

Note that the square of the wave function is rescaled so that you can see the shape clearly.

The 26th eigenstate and 29th eigenstate are resonance states of the double barrier.

- 1st resonance:
  - the 26th eigenstate: 0.177 (eV) (delocalized)
  - the 29th eigenstate: 0.193 (eV) (more localized)
- 2nd resonance:
  - the 56th eigenstate: 0.311 (eV) (delocalized)
  - the 59th eigenstate: 0.328 (eV) (more localized)
  - the 61th eigenstate: 0.336 (eV) (delocalized)
  - the 63th eigenstate: 0.347 (eV) (delocalized)

- the 64th eigenstate: 0.352 (eV) (more localized)

### TO BE CHECKED

The follow figure shows the square of the wave function of the 26th eigenstate with the conduction band. (*bias\_00000\Quantum\probabilities\_shift\_device\_Gamma.fld*) You can clearly see that it is a resonance state of the double barrier and corresponds to the second peak in the light-blue transmission curve  $T_{13}$  from source to draian around 190 (meV).

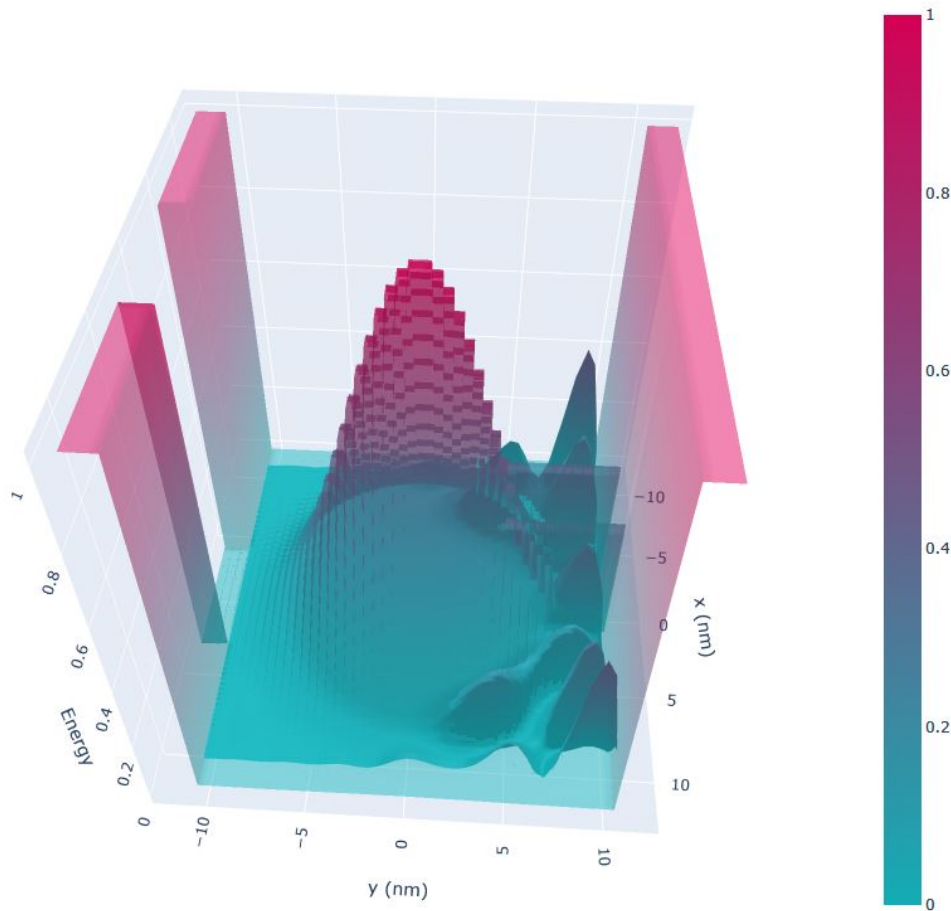


Figure 3.20.16.5: The 26th eigenstate

Note that the square of the wave function is rescaled so that you can see the shape clearly.

### Lead modes

Figure 3.20.16.6 shows the lead modes of the gate, and the source (which is identical to the drain). In the transmission curve  $T_{12}(E) = T_{23}(E)$ , the transmisson shows a step-like behavior which is related to the energies of lead 2 ('gate').

### Notes

#### ⚠ Attention

Here the difference in the way of setting up boundary conditions in *nextnano++* and *nextnano<sup>3</sup>* will be discussed.

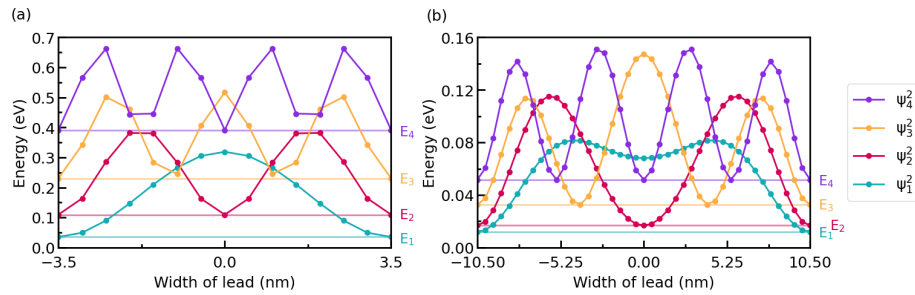


Figure 3.20.16.6: The lead modes of lead 2 (‘gate’) are shown in (a), whereas the lead modes of lead 1, 3 (‘source’, ‘drain’) are shown in (b). `bias_00000\Quantum\probabilities_shift_lead_X_Gamma.dat`

Last update: nnnn/nn/nn

### 3.20.17 Hexagonal GaAs/AlGaAs nanowires

#### Input files:

- `2DGaAs_AlGaAs_circle_nnp.in`
- `2DGaAs_AlGaAs_hexagon_nnp.in`
- `2D_Hexagonal_Nanowire_2DEG_nnp.in`

#### Scope:

In this tutorial we simulate a circular and a hexagonal  $GaAs/Al_{0.33}Ga_{0.67}As$  core-shell structure (Part A) and a hexagonal  $GaAs/AlGaAs$  nanowire structure (Part B).

#### Output files:

- `bias_00000\Quantum\probabilities_quantum_region_Gamma_vtr`

#### Part A: Schrödinger equation of a two-dimensional core-shell structure

In this part of the tutorial, we solve the two-dimensional Schrödinger equation of a circular and a hexagonal  $GaAs/Al_{0.33}Ga_{0.67}As$  core-shell structure.

##### Circular core-shell structure

Input file: `2DGaAs_AlGaAs_circle_nnp.in`

Figure 3.20.17.1 shows the probability density of the 6th eigenstate of the circular  $GaAs/AlGaAs$  structure. The data is contained in the file `bias_00000\Quantum\probabilities_quantum_region_Gamma_vtr`. Its energy level is higher than the  $AlGaAs$  barrier energy, i.e. this state is not confined in the circular shaped  $GaAs$  quantum well. The horizontal and vertical slices are through the center and show the square of the probability amplitude of this eigenstate.

The  $GaAs$  core has a radius of 5 nm. (It cannot be recognized on this plot.) The  $AlGaAs$  shell has a radius of 15 nm. It is surrounded by an infinite barrier which comes from the “band offset” due to the surrounding material “air”.

##### Hexagonal core-shell structure

Input file: `2DGaAs_AlGaAs_hexagon_nnp.in`

Figure 3.20.17.2 shows the conduction band edge of the hexagonal  $GaAs/AlGaAs$  structure. The  $GaAs$  region is indicated in black, the  $AlGaAs$  region in blue. Horizontal and vertical slices through the center show the energy of the conduction band edge profile. The data is contained in the file `bias_00000\bandedges.fld`

The diameter of the hexagonal shaped  $GaAs$  core is ~8.66 nm (corresponding to an outer radius of the core of 5 nm), and the diameter of the hexagonal shaped  $AlGaAs$  shell is ~26 nm (corresponding to an outer radius of the shell of 15 nm).

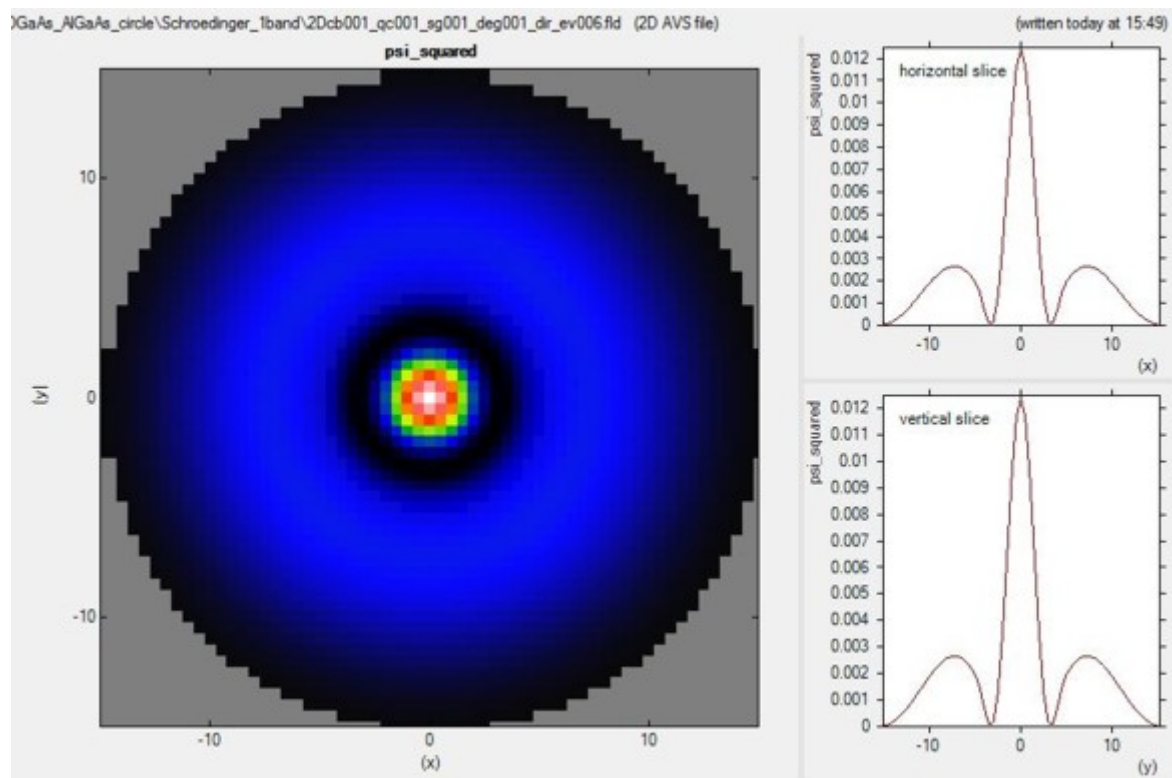
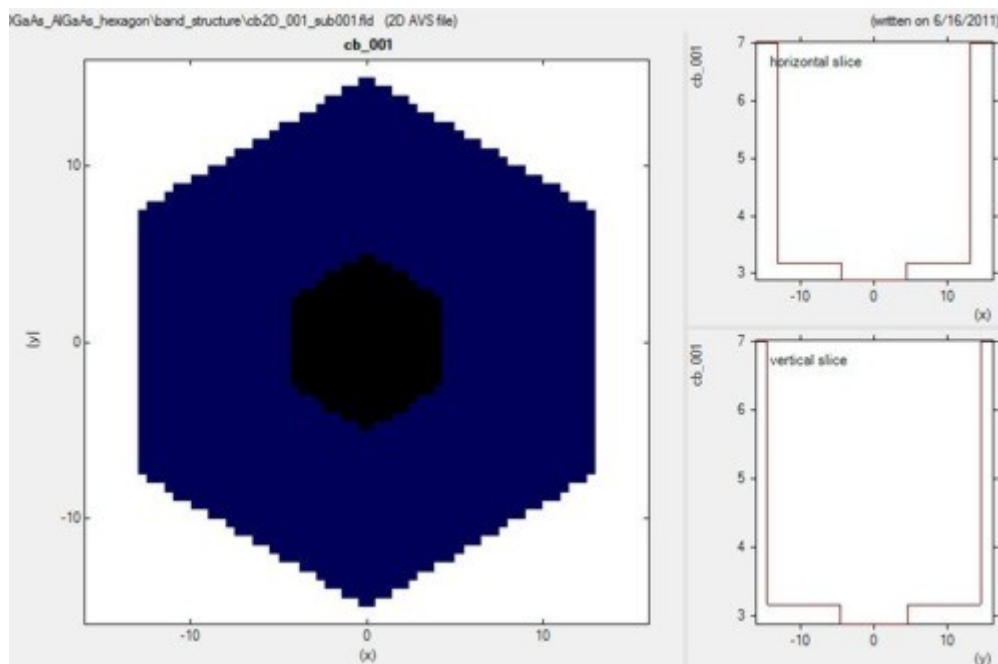
Figure 3.20.17.1:  $\Psi^2$  of the 6th electron eigenstate.

Figure 3.20.17.2: Conduction band edge profile of the hexagonal core-shell structure.



Figure 3.20.17.3 shows the probability density of the 10th eigenstate of the circular *GaAs*/*AlGaAs* structure. The data is contained in the file `bias_00000\Quantum\probabilities_quantum_region_Gamma_vtr`. It's energy level is higher than the *AlGaAs* barrier energy, i.e. this state is not confined in the hexagonal shaped *GaAs* quantum well. The horizontal and vertical slices are through the center and show the square of the probability amplitude of this eigenstate.

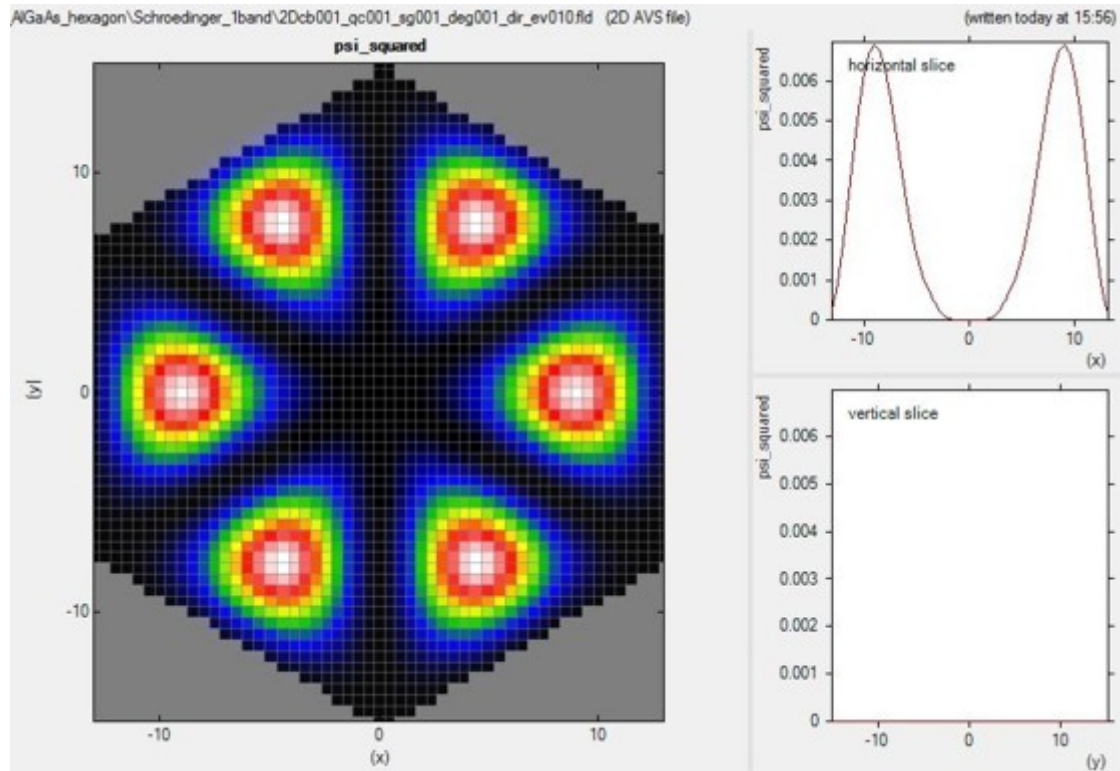


Figure 3.20.17.3:  $\Psi^2$  of 10th electron eigenstate.

The hexagonal *GaAs* core has an outer radius of 5 nm. It cannot be seen on this plot. The *AlGaAs* shell has a diameter of 26 nm. It is surrounded by an infinite barrier which case comes from the “band offset” due to the surrounding material “air”.

### Alloy sweep

#### **Note**

In comparison with *nextnano*<sup>3</sup>, *nextnano++* does not support an alloy sweep in the input file. However, you can use *nextnanomat*'s Template feature to perform an alloy sweep.

In the following, we vary the alloy content  $x$  of the ternary  $Al_xGa_{1-x}As$  from 0 to 0.33 in 11 steps. For  $x = 0$ , we have pure *GaAs*. For  $x = 0.33$  we have an *AlGaAs*/*GaAs* conduction band offset of 0.285 eV, and a valence band offset of -0.168 eV. In the latter case, the quantum confinement is stronger. Even for  $x = 0$  we have “quantum confinement” due to the Dirichlet boundary conditions (corresponding to infinite barriers) at the shell surface that we use for the Schrödinger equation. Consequently, even for  $x = 0$ , we get an e1 - h1 transition energy from the lowest electron state (e1) to the highest heavy hole state (h1) that is larger than the band gap as shown in Figure 3.20.17.4.

The transition energies (e1 - h1), as well as the spatial overlap integral of the electron and hole ground state wave functions, are contained in this file: `bias_00000\Quantum\momentum_matrix_elements_quantum_region_Gamma_001.txt`

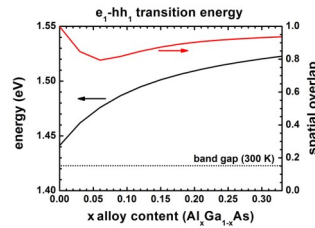


Figure 3.20.17.4: Transition energy and spatial overlap of e1 to h1 transition as a function of alloy content  $x$ .

| alloy                        | type                                                        | e1-h1[eV]   | e1[eV]      | h1[eV]      |    |
|------------------------------|-------------------------------------------------------------|-------------|-------------|-------------|----|
| $\rightarrow$ matrix_element |                                                             |             |             |             |    |
| 0.330000                     | $ \langle \text{psi\_vb001}   \text{psi\_cb001} \rangle ^2$ | 1.522777615 | 2.965889676 | 1.443112062 | 0. |
| $\rightarrow$ 936344908      |                                                             |             |             |             |    |
| 0.300000                     | $ \langle \text{psi\_vb001}   \text{psi\_cb001} \rangle ^2$ | 1.520316794 | 2.963669699 | 1.443352905 | 0. |
| $\rightarrow$ 931291593      |                                                             |             |             |             |    |
| ...                          |                                                             |             |             |             |    |

The spatial overlap of electron and hole wave functions is always very high. When there is only confinement due to the shell boundary, the matrix element is very high (99.8 %). The matrix element must be smaller than 1 for  $x = 0$  because the electron and hole masses are different. The matrix element must be even smaller (94 %) for  $x = 0.33$  (strong confinement) because in addition to the mass difference, the conduction and valence band offsets are not equivalent. The matrix element has a minimum at around  $x = 0.06$  because in this case the electron wave function penetrates into the barrier much stronger than the hole wave function does. Thus the differences in well and barrier masses (as well as band offsets) play an important role for the spatial extension of the wave functions.

## Part B: Hexagonal 2DEG - Two-dimensional electron gas in a delta-doped hexagonal shaped $GaAs/AlGaAs$ nanowire heterostructure

Input file: *2D\_Hexagonal\_Nanowire\_2DEG\_nnp.in*

The following example deals with a delta-doped  $GaAs/AlGaAs$  2DEG (two-dimensional electron gas) structure. In this case, the heterostructure consists of a hexagonal  $GaAs/AlGaAs$  nanowire, see Figure 3.20.17.5.

The self-consistently calculated conduction band edge (*bandedges.fld*) is shown in Figure 3.20.17.6. The horizontal and vertical slices through the center indicate the triangular potential well (conduction band minimum) where the 2DEG is located.

The resulting 2DEG electron density (*bias\_00000density\_electron.fld*) is shown Figure 3.20.17.7. At the corners, the electron density is significantly higher, thus one-dimensional conducting channels are formed. Although the structure itself has a hexagonal symmetry, our rectangular grid breaks this symmetry. Therefore the density in the upper/lower corner are different from the density at the left/right corners.

The 2D Poisson equation and the 2D Schrödinger equation have been solved self-consistently. The dimension of the Schrödinger matrix is 28,625. The CPU time for this calculation was about 18 minutes.

Last update: 27/05/2025

### 3.20.18 Fock-Darwin states of a parabolic, anisotropic (elliptical) potential in a magnetic field

- 1D parabolic confinement along the  $x$  direction with  $\hbar\omega_0 = 4.6$  meV (1D simulation)
- 1D parabolic confinement along the  $y$  direction with  $\hbar\omega_0 = 6.1$  meV (1D simulation)



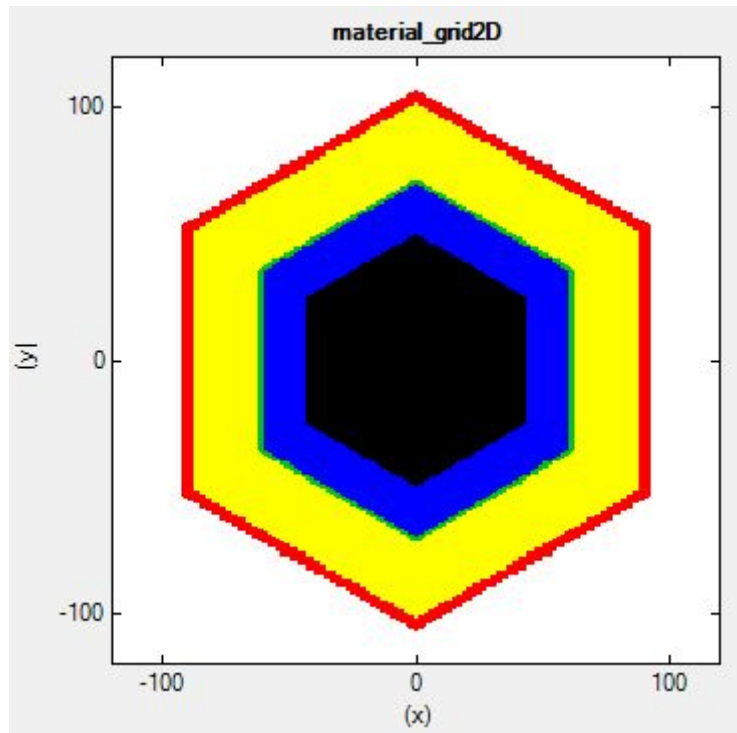


Figure 3.20.17.5: The material layers of the structure: *GaAs* core (black), *AlGaAs* spacer (blue), Si-doped *AlGaAs* (green), *AlGaAs* (yellow), *GaAs* capping layer (red) and Schottky barrier contact (black) are shown. (The white layer itself is not included in the calculation. It only serves as a boundary condition)

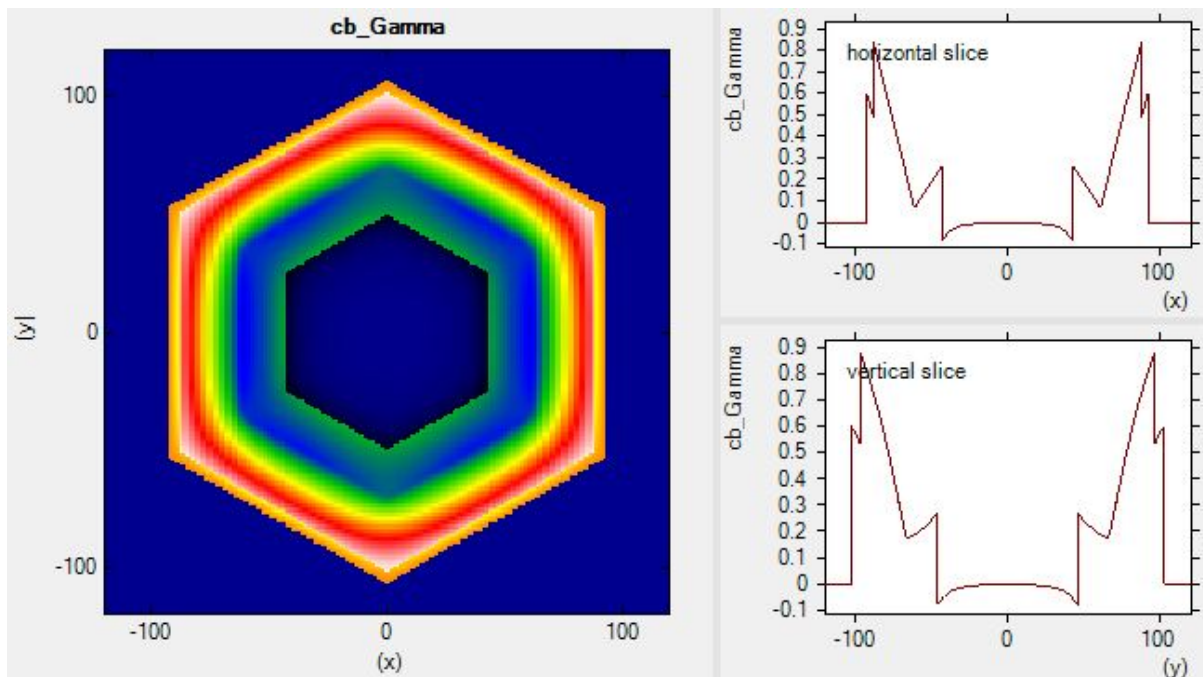


Figure 3.20.17.6: Conduction band edge profile.

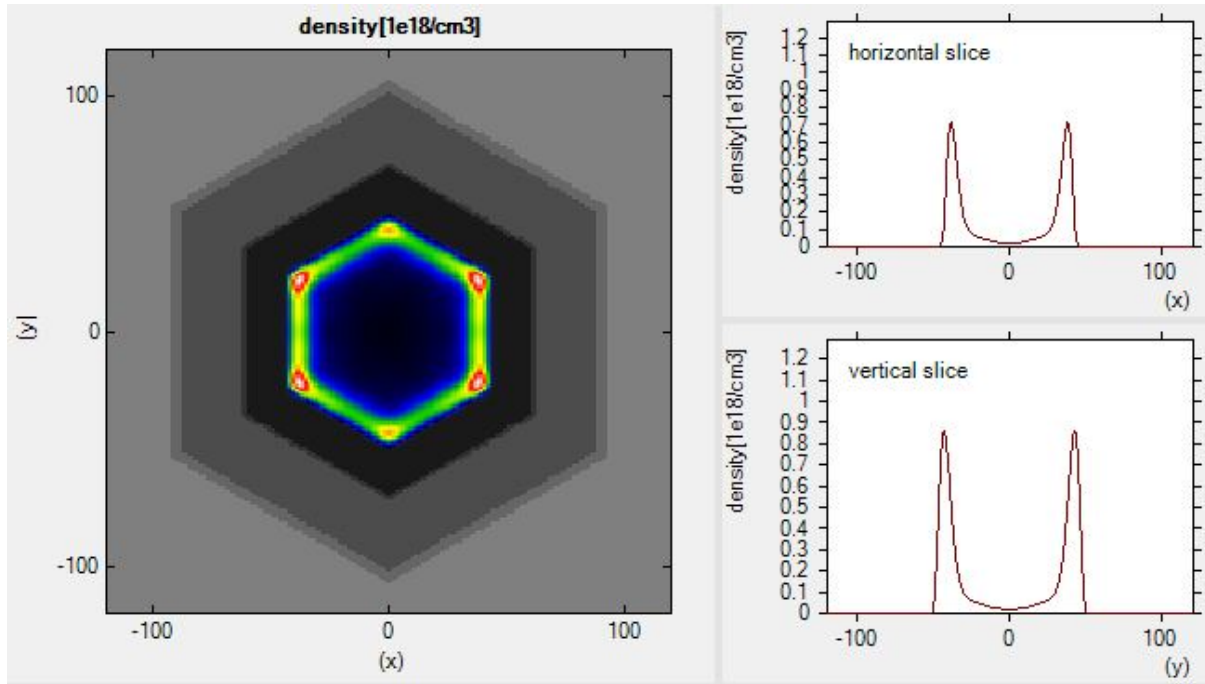


Figure 3.20.17.7: Charge density profile.

- *2D parabolic, anisotropic (elliptical) confinement with  $\hbar\omega_x = 4.6$  meV and  $\hbar\omega_y = 6.1$  meV - Fock-Darwin-like spectrum (2D simulation)*

In this tutorial we study the electron energy levels of a two-dimensional parabolic, anisotropic (elliptical) confinement potential that is subject to a magnetic field. Such a potential can be constructed by surrounding GaAs with an  $\text{Al}_x\text{Ga}_{1-x}\text{As}$  alloy that has a parabolic alloy profile in the (x,y) plane.

It is a good idea to get familiar with the results of a 2D parabolic and isotropic confinement beforehand: *Fock-Darwin states of a 2D parabolic potential in a magnetic field*

The input files used in this tutorial are the followings:

- `1DGaAs_ParabolicQW_infinite_4_6meV.in`
- `1DGaAs_ParabolicQW_infinite_6_1meV.in`
- `2DGaAs_BiParabolicEllipticQD_Austing.in/*_nnp.in`

First, it is necessary to study the energy states of a 1D parabolic confinement.

### 1D parabolic confinement along the x direction with $\hbar\omega_0 = 4.6$ meV (1D simulation)

- `1DGaAs_ParabolicQW_infinite_4_6meV.in`

For similar results and a discussion, we refer to this tutorial: [Parabolic Quantum Well \(GaAs / AlAs\)](#)

**1D parabolic confinement along the y direction with  $\hbar\omega_0 = 6.1$  meV (1D simulation)**

- `1DGaAs_ParabolicQW_infinite_6_1meV.in`

For similar results and a discussion, we refer to this tutorial: [Parabolic Quantum Well \(GaAs / AlAs\)](#)

**2D parabolic, anisotropic (elliptical) confinement with  $\hbar\omega_x = 4.6$  meV and  $\hbar\omega_y = 6.1$  meV - Fock-Darwin-like spectrum (2D simulation)**

- `2DGaAs_BiParabolicEllipticQD_Austing.in/*_nnp.in`

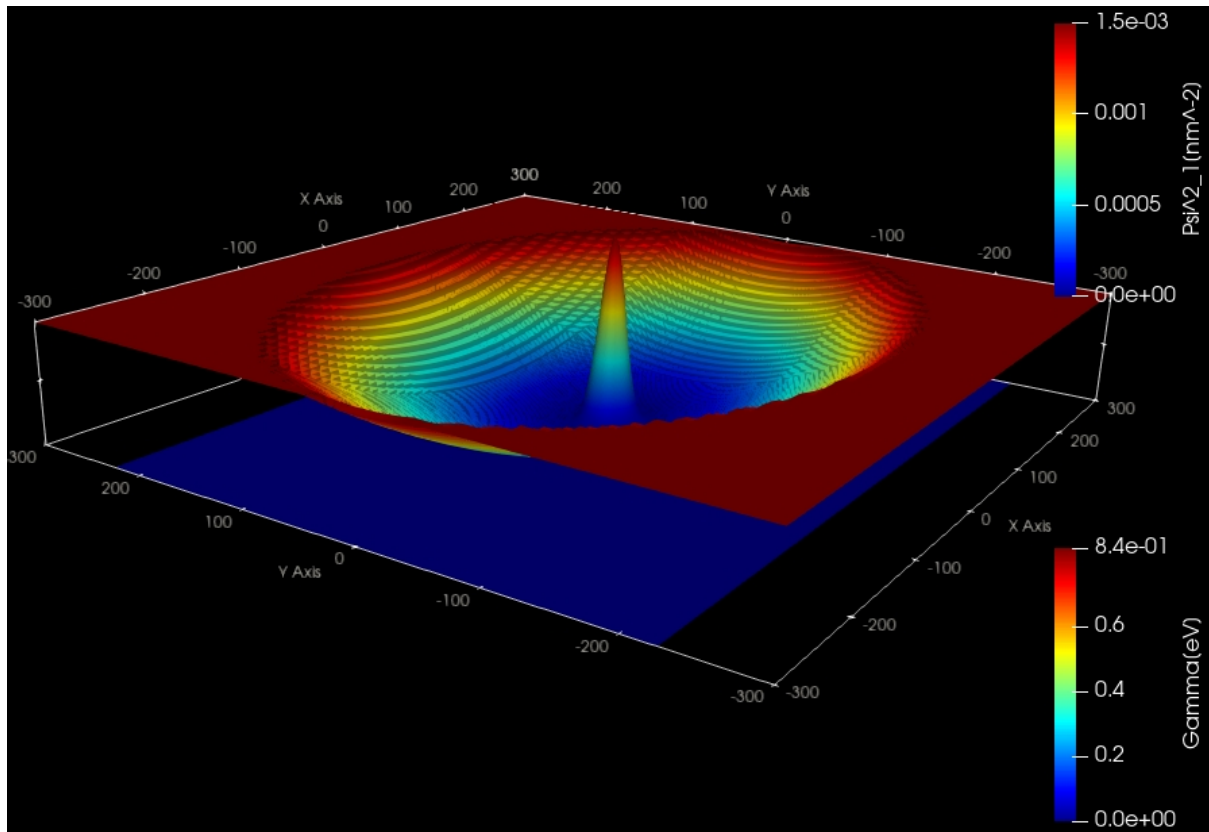
The electron effective mass in GaAs is  $m_e^* = 0.067m_0$ . We assume this value for the effective mass in the whole region (i.e. also inside the AlGaAs alloy).

**Ground state wave function ( $\psi^2$ )**

The following figure shows the parabolic, anisotropic (elliptical) conduction band edge confinement potential, as well as the ground state wave function ( $\psi^2$ ) at  $B = 0$  T calculated in *nextnano++*. In the middle of the sample the conduction band edge is at 0 eV and at the boundary region the conduction band edge has the value 0.84 eV. The radii of the ellipse are 300 nm along the x axis and 226 nm along the y axis. The parabolic confinement along the x direction is:  $\hbar\omega_x = 4.6$  meV

The parabolic confinement along the y direction is:  $\hbar\omega_y = 6.1$  meV

Thus the ellipticity is roughly 4/3.



Fock-Darwin spectrum

At zero magnetic field, the eigenvalues for such a system are given by:

$$E_{n_x, n_y} = (n_x + \frac{1}{2})\hbar\omega_x + (n_y + \frac{1}{2})\hbar\omega_y$$
$$n = n_x + |l|, \quad n_y = n - |l|$$

*for  $n = 0, 1, 2, 3, \dots$   $l = 0, \pm 1, \pm 2, \dots$*

where  $n$  is a radial quantum number,  $l$  an angular momentum quantum number,  $\omega_x$  and  $\omega_y$  oscillator frequencies.

For more details, see A.V. Madhav, T. Chakraborty, Physical Review B 49, 8163 (1994).

The eigenvalue spectrum of a 2D parabolic and **isotropic** potential shows a shell-like structure: [Energy levels of an “artificial atom” - 2D harmonic potential](#) . For the **anisotropic elliptical** potential, this degeneracy at  $B = 0$  T is lifted.

The following figure shows the calculated Fock-Darwin-like spectrum, i.e. the eigenstates as a function of magnetic field magnitude. This is the result of *nextnano<sup>3</sup>* and each of these states is two-fold spin-degenerate. However, a magnetic field lifts this degeneracy (Zeeman splitting) but this effect is not taking into account in this tutorial.

Such a spectrum can be related to experimental transport measurements which give insight into the single-particle energy spectrum of a quantum dot.

The rectangles in the above figure are related to the figures of the following publications:

cyan rectangle: Fig.2 of

Two-level anti-crossings high up in the single-particle energy spectrum of a quantum dot  
C. Payette, D.G. Austing, G. Yu, J.A. Gupta, S.V. Nair, B. Partoens, S. Amaha, S. Tarucha  
[arXiv:0710.1035v1 \[cond-mat.mes-hall\]](#) (2007)

green rectangle and red rectangle: Fig.2(b) and Fig.3(a) of

Probing by transport the single-particle energy spectrum up to high energy of one quantum dot with the ground state of an adjacent weakly coupled quantum dot  
D.G. Austing, G. Yu, C. Payette, J.A. Gupta, M. Korkusinski, G.C. Aers  
physica status solidi (a), 508 (2007)

(Comments on red rectangle: In Fig. 3(a) of the publication by Austing et al., the ground state energy has been subtracted from the excited states. Thus the slope of the energy spectrum look slightly different.)

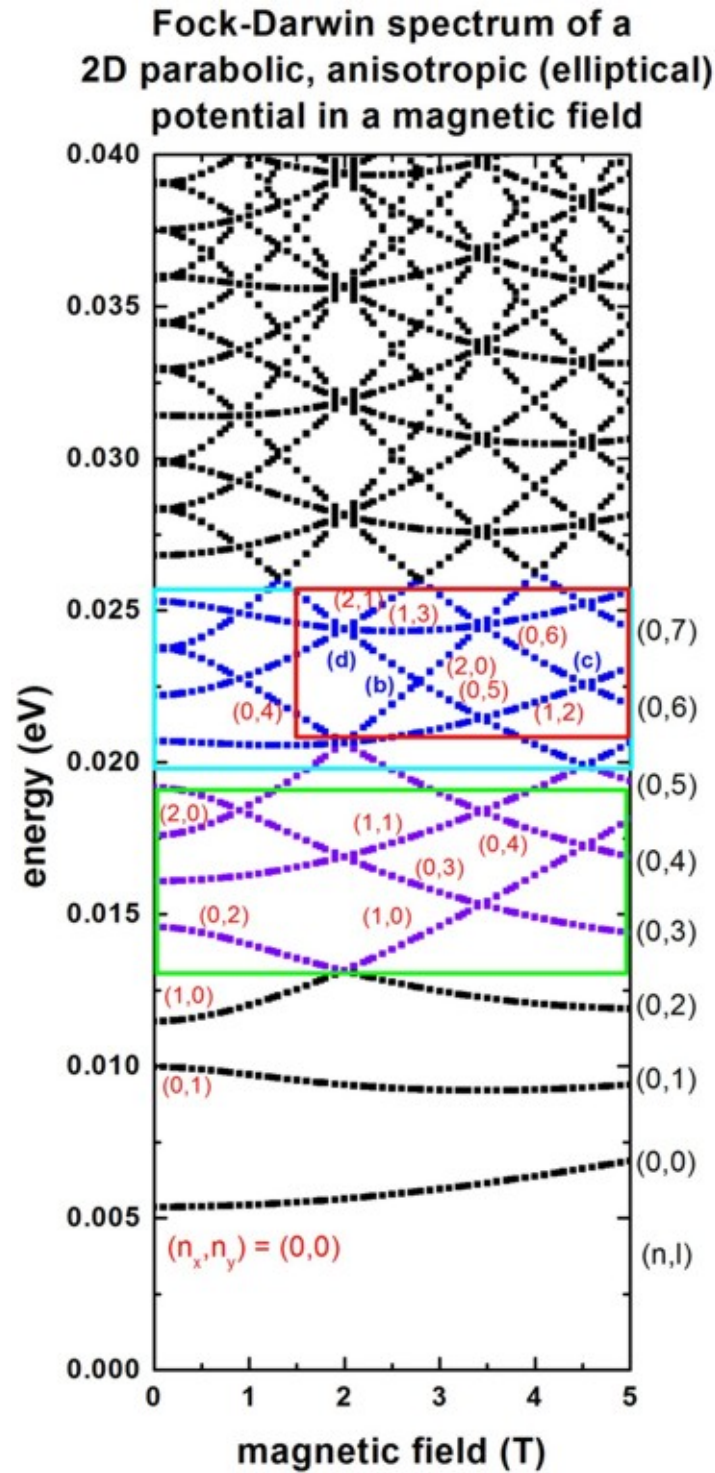
It is interesting to note that there are exact crossings in the calculated spectrum whereas the experiment reveals anti-crossings. In the first reference, this difference on crossings is regarded as a clue to investigate the deviations between the confining potential of realistic dots used in the experiment and the idealistic parabolic potential assumed in the calculation.

---

*Last update: nnnn/nn/nn*

### 3.20.19 Fock-Darwin states of parabolic, isotropic potential in a magnetic field

- *Header*
- *Introduction*
- *2D parabolic confinement with  $\hbar\omega_0 = 4$  meV*



- Results
- 2D parabolic confinement with  $\hbar\omega_0 = 3$  meV - Fock-Darwin spectrum
- Results

## Header

### Input files

- 2DGaAs\_BiParabolicQW\_4meV\_GovernalePRB1998.in/\*\_nnp.in
- 2DGaAs\_BiParabolicQW\_3meV\_FockDarwin.in/\*\_nnp.in

## Introduction

In this tutorial, we study the electron energy levels of a two-dimensional parabolic confinement potential that is subject to a magnetic field. Such a potential can be constructed by surrounding GaAs with an  $\text{Al}_x\text{Ga}_{1-x}\text{As}$  alloy that has a parabolic alloy profile in the (x,y) plane.

The magnetic field  $\mathbf{B}$  is oriented along the z direction.  $\mathbf{B}$  is the rotation of the vector potential  $\mathbf{A}$  so, in this case, we can always take the z-component of the vector potential as 0. Thus the motion in the z direction is not influenced by the magnetic field and that of a free particle with energies and wave functions given by:

$$E_z = \frac{\hbar^2 k_z^2}{2m_e^*}$$

$$\psi(z) = \exp(\pm i k_z z)$$

For that reason, we do not include the z direction into our simulation domain, and thus only simulate in the (x,y) plane (two-dimensional simulation).

This tutorial consists of two parts. First we benchmark the *nextnano++* code to the numerical calculation in [GovernalePRB1998]. Second we reproduce some figures of [KouwenhovenRPP2001].

The figures provided in this tutorial is the results of *nextnano++* input files.

### Note

When `magnetic_field` is specified in a 2D or 3D simulation of *nextnano++*, the Pauli equation, in which both spin eigenfunctions are taken into consideration, is calculated instead of Schrödinger equation. On the other hand, the effect of spin is not considered in *nextnano*<sup>3</sup>. Thus the number of eigenstates calculated in this *nextnano++* input file is two times of the one in the *nextnano*<sup>3</sup> input file. Since the splitting of energy levels due to the spin is small compared to the difference of energy levels, we call the two states split from *i*th eigenstates as “*i*th eigenstates with up-spin” or “*i*th eigenstates with down-spin”.

## 2D parabolic confinement with $\hbar\omega_0 = 4$ meV

We want to benchmark the *nextnano++* code to the numerical calculation in [GovernalePRB1998].

Input file `2DGaAs_BiParabolicQW_4meV_GovernalePRB1998.in/*_nnp.in` aims to reproduce the figures of eigenvalues, ground state and 14th excited state probability densities, and ground state energy as a function of magnetic field magnitude (Fig. 1, 2, 3 and 4 of the paper).

The GaAs sample extends in the x and y directions (i.e. this is a two-dimensional simulation) and has the size of 240 nm x 240 nm. At the domain boundaries we employ Dirichlet boundary conditions to the Schrödinger equation, i.e. infinite barriers. The grid is chosen to be rectangular with a grid spacing of 2.4 nm, in agreement with [GovernalePRB1998].

A two-dimensional parabolic confinement potential is constructed by surrounding GaAs with an  $\text{Al}_x\text{Ga}_{1-x}\text{As}$  alloy that has a parabolic alloy profile in the (x,y) plane. This is chosen so that the electron ground state has the energy:  $E_1 = \hbar\omega_0 = 4$  meV (without magnetic field).



The magnetic field is oriented along the z direction, i.e. it is perpendicular to the simulation plane which is oriented in the (x,y) plane). (In *nextnano++*, the direction is automatically set to the direction perpendicular to the simulation plane.) We calculate the eigenstates for different magnetic field strengths (1 T, 2 T, ..., 20 T), i.e. we make use of the magnetic field sweep. Since *nextnano++* does not have this feature for `magnetic_field` so far, please use the “Template” feature of *nextnanomat* (See the last section of — *FREE — Double Quantum Well* .)

```
global{
  ...
  magnetic_field{
    strength = $STRENGTH
    #direction = [,] # We must not specify this in 1D or 2D simulation
  }
}
```

In *nextnano<sup>3</sup>*, the magnetic field sweeping can be specified in the input file:

```
$magnetic-field
magnetic-field-on           = yes
magnetic-field-strength    = 0.0d0    ! 1 Tesla = 1 Vs/m2
magnetic-field-direction    = 0 0 1    ! [001] direction
magnetic-field-sweep-active = yes      !
magnetic-field-sweep-step-size = 0.5d0 ! 0.5 Tesla = 0.5 Vs/m2
magnetic-field-sweep-number-of-steps = 40 ! 40 magnetic field sweep steps
$end_magnetic-field
```

### Magnetic length and cyclotron frequency

A useful quantity is the magnetic length (or Landau magnetic length) which is defined as:

$$l_B = \left( \frac{\hbar}{m_e^* \omega_c} \right)^{1/2} = \left( \frac{\hbar}{|e|B} \right)^{1/2}$$

It is independent of the mass of the particle and depends only on the magnetic field strength:

- 1 T:  $l_B = 25.6556$  nm
- 2 T:  $l_B = 18.1413$  nm
- 3 T:  $l_B = 14.8123$  nm
- ...
- 20T:  $l_B = 5.7368$  nm

The electron effective mass in GaAs is  $m_e^* = 0.067m_0$ . We assume this value for the effective mass in the whole region (i.e. also inside the AlGaAs alloy). In the above formula,  $\omega_c$  is the cyclotron frequency:

$$\omega_c = \frac{|e|B}{m_e^*}$$

Thus for the electrons in GaAs, where  $m_e^* = 0.067m_0$ , it holds for the different magnetic field strengths:

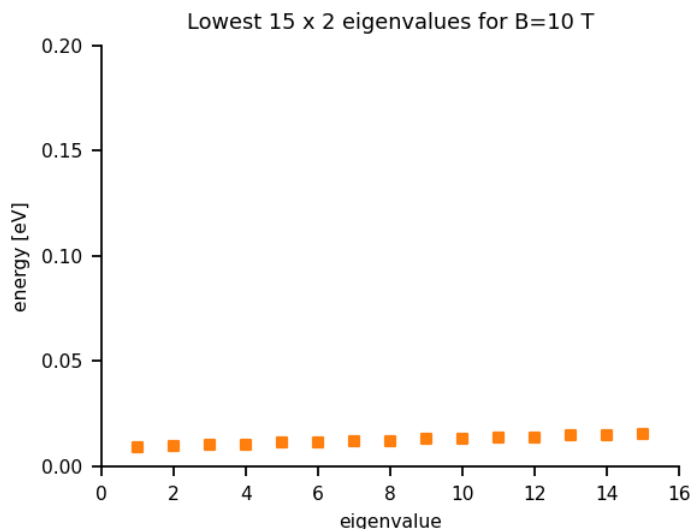
- 1 T:  $\hbar\omega_c = 1.7279$  meV
- 2 T:  $\hbar\omega_c = 3.4558$  meV
- 3 T:  $\hbar\omega_c = 5.1836$  meV
- ...
- 20T:  $\hbar\omega_c = 34.5575$  meV

## Results

### Lowest 15 eigenvalues

The following figure shows the lowest fifteen eigenvalues for a magnetic field magnitude of  $B = 10$  T. It

is in perfect agreement with Fig. 1 of [GovernalePRB1998]. The ground state has the energy  $E_{1\uparrow} = 9.38$  meV and  $E_{1\downarrow} = 9.55$  (at  $B = 10$  T). The spin-split energy,  $\frac{e\hbar B}{2m_e^*}$  is 0.174 meV, is calculated from our result as 0.174 meV which is constant in all of the pair of spin states.



### Probability densities ( $\psi^2$ )

The following figure shows the probability density of the ground state with up-spin ( $\psi^2$ ) for a magnetic field magnitude of  $B = 10$  T. It is in perfect agreement with Fig. 2(a) of [GovernalePRB1998]. The ground states has the energy  $E_{1,\uparrow} = 9.38$  meV and  $E_{1,\downarrow} = 9.55$  (at  $B = 10$  T) in *nextnano++*. The corresponding eigenvalue calculated in *nextnano<sup>3</sup>* is  $E_1 = 9.44$  meV.

The left, vertical axis shows  $\psi^2$  in units of  $\text{nm}^{-2}$  (the peak value is  $0.00267 \text{ nm}^{-2}$ ).

In the same figure, the parabolic conduction band edge confinement potential is also shown. The above axis shows the colormap of the conduction band edge values. In the middle of the sample the conduction band edge is 0 eV, and at the boundary region, the conduction band edge has the value 0.1014 eV.

The following figure shows the probability density ( $\psi^2$ ) of the 14th excited state (up-spin) (i.e.  $E_{15,\uparrow}$ ) for a magnetic field magnitude of  $B = 10$  T. It is in perfect agreement with Fig. 3(a) of [GovernalePRB1998]. 14th excited states have the energy  $E_{15,\uparrow} = 21.71$  and  $E_{15,\downarrow} = 21.88$  meV (at  $B = 10$  T). The corresponding eigenvalue calculated in *nextnano<sup>3</sup>* is  $E_{15} = 21.72$  meV. The left, vertical axis shows  $\psi^2$  in units of  $\text{nm}^{-2}$  (the peak value is  $0.000283 \text{ nm}^{-2}$ ).

In the same figure, parabolic conduction band edge confinement potential is also shown. The above axis shows the colormap of the conduction band edge values. In the middle of the sample the conduction band edge is 0 eV, and at the boundary region, the conduction band edge has the value 0.1014 eV.

### Ground state energy vs. magnetic field magnitude

The following figure shows the ground state energy as a function of magnetic field magnitude. It is in perfect agreement with Fig.4 of [GovernalePRB1998]. The ground state has the energy  $E_1 = 4.04$  meV (spin-degenerated) in *nextnano++* and  $E_1 = 4.01$  meV in *nextnano<sup>3</sup>* at  $B = 0$  T.

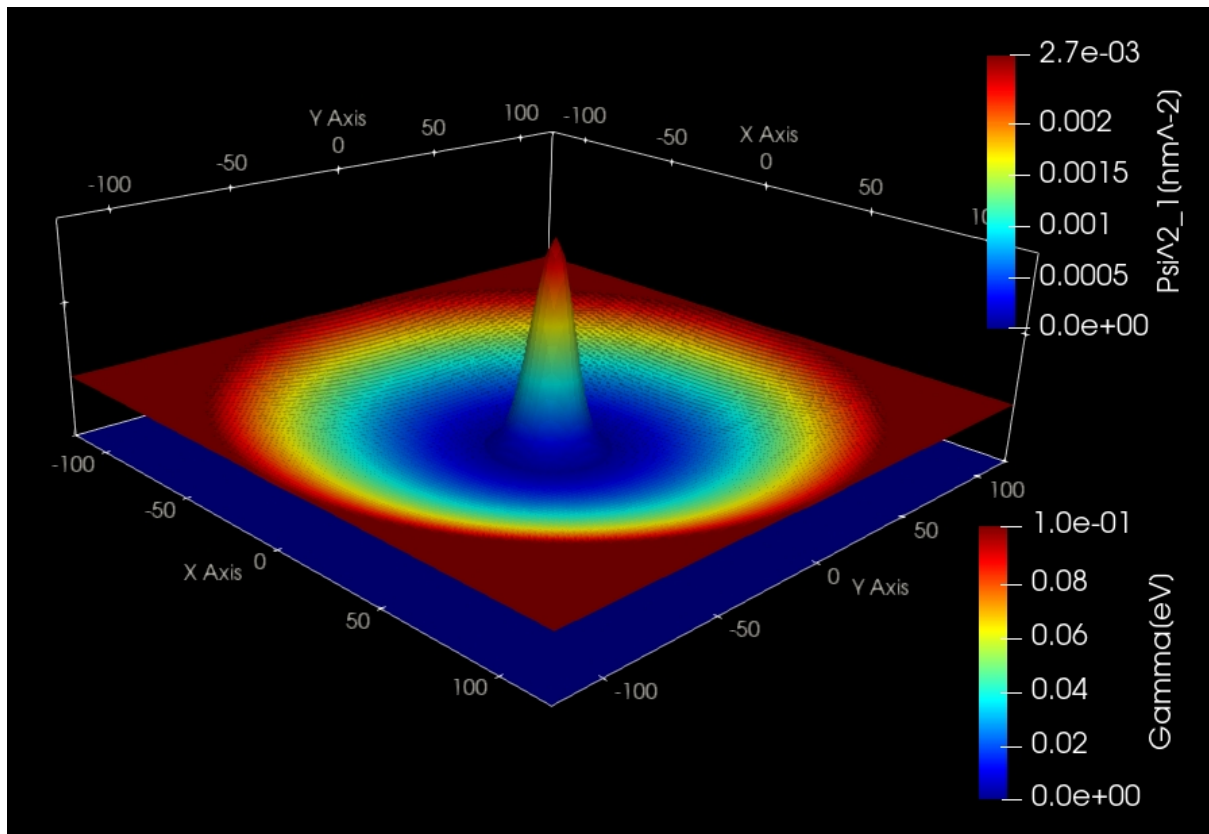
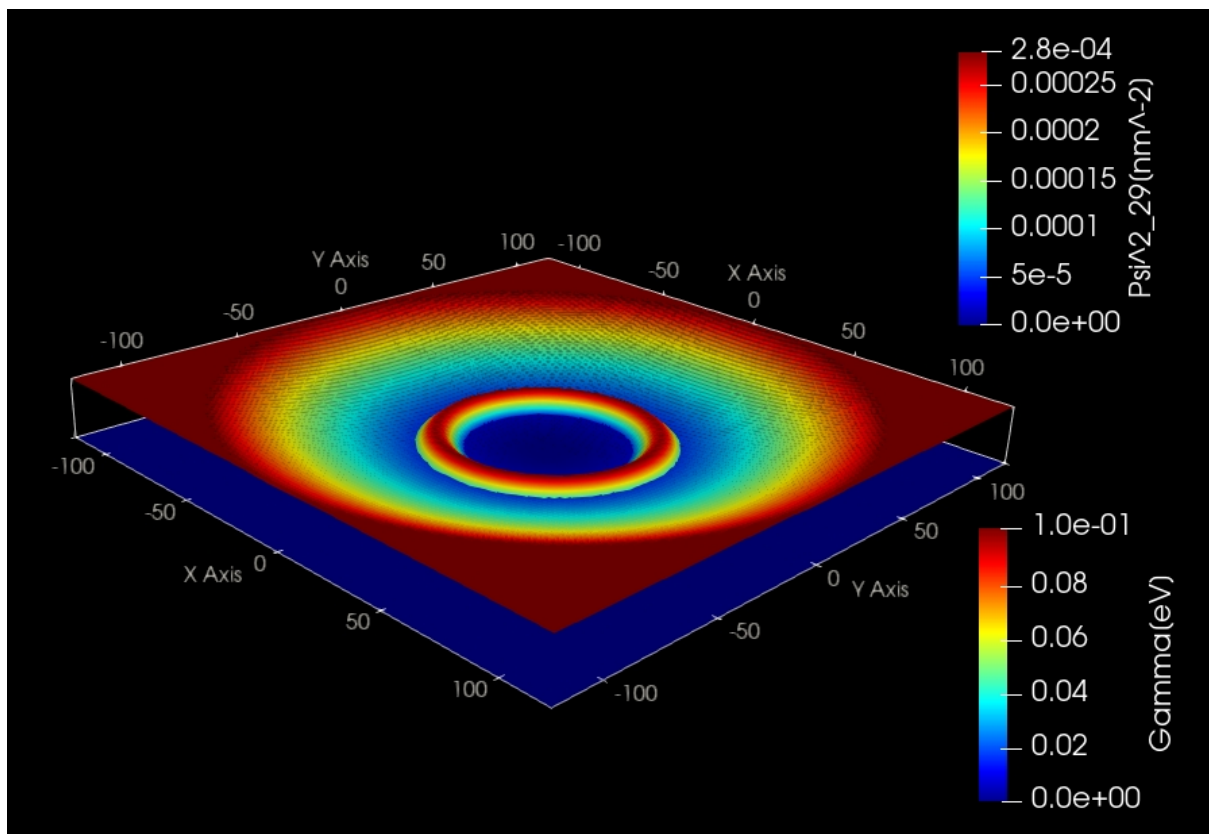
The following figure shows the magnetic field stlength dependence of the spin-split energy ( $E_{1,\downarrow} - E_{1,\uparrow}$ ). The formula of the split energy in the Pauli equation is  $\frac{e\hbar B}{2m_e^*}$ . We can see the proportionality is reproduced in our calculation. The factor is calculated as  $0.0174 \text{ [meV/T]}$ .

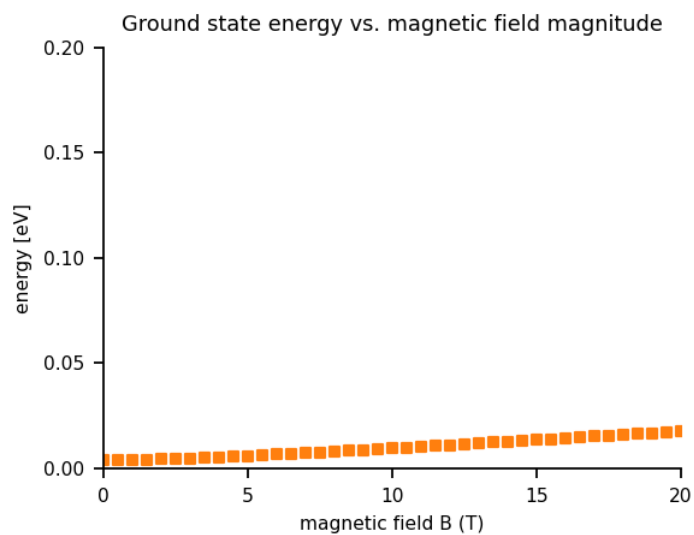
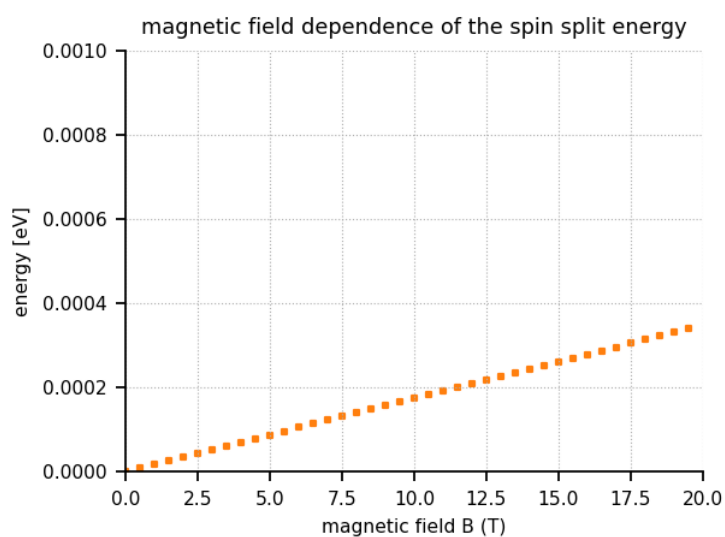
### 2D parabolic confinement with $\hbar\omega_0 = 3$ meV - Fock-Darwin spectrum

Next we reproduce some of the figures of [KouwenhovenRPP2001].

Input file *2DGaAs\_BiParabolicQW\_3meV\_FockDarwin.in/\*\_nnp.in* aims to reproduce the figures of the eigenvalues as a function of magnetic field magnitude and the probability densities of some of eigenstates (Figs. 5(a) and 6(a) (which are analytical results) of the paper).







The GaAs sample extends in the x and y directions (i.e. this is a two-dimensional simulation) and has the size of 600 nm x 600 nm. At the domain boundaries we employ Dirichlet boundary conditions to the Schrödinger equation, i.e. infinite barriers.

A two-dimensional parabolic confinement potential is constructed by surrounding GaAs with an Al<sub>x</sub>Ga<sub>1-x</sub>As alloy that has a parabolic alloy profile in the (x,y) plane. This is chosen so that the electron ground state has the energy:  $E_1 = \hbar\omega_0 = 3$  meV (without magnetic field) in agreement to the paper.

The eigenvalues of a two-dimensional parabolic potential that is subject to a magnetic field can be solved analytically. The spectrum of the resulting eigenstates is known as the **Fock-Darwin states** (1928):

$$E_{n,l} = (2n + |l| + 1)\hbar[w_0^2 + \frac{1}{4}\omega_c^2]^{1/2} - \frac{1}{2}l\hbar\omega_c \quad \text{for } n = 0, 1, 2, 3, \dots \text{ and } l = 0, \pm 1, \pm 2, \dots$$

Note that the last term is  $\omega_c$  and not  $\omega_0$  as in [KouwenhovenRPP2001]. ( $\omega_c = \frac{|e|B}{m_e^*} =$  cyclotron frequency, as described before.)

Each of these states is two-fold spin-degenerate. A magnetic field lifts this degeneracy (Zeeman splitting). This effect is taken into account only in the input file of *nextnano++* but this splitting is small compared to the scale of  $E_{n,l}$ .

The degeneracy of the eigenvalues for zero magnetic field is as follows:

- the ground state is not degenerate
- the second state is two-fold degenerate
- the third state is three-fold degenerate
- the fourth state is four-fold degenerate
- ...

Applying a magnetic field, these degeneracies are lifted as the following figure.

## Results

### Fock-Darwin spectrum

The following figure shows the calculated Fock-Darwin spectrum, i.e. the eigenstates as a function of magnetic field magnitude. The figure is in excellent agreement with Fig. 5(a) of [KouwenhovenRPP2001].

### Probability densities ( $\psi^2$ )

The following figure shows the probability densities ( $\psi^2$ ) of some of these eigenstates for a magnetic field of  $B = 0.05$  T. All of them are the up-spin states. The label of the colorbar shows the actual number of each eigenstates specified in the data file. For example, 5th state in this figure has the label “Psi<sup>2</sup><sub>9[nm<sup>-9</sup>]</sub>”.

The figures are in excellent agreement with Fig. 6(a) of [KouwenhovenRPP2001].

The parabolic conduction band edges are also shown.

### Fock-Darwin spectrum in a very high magnetic fields

The following figure shows the magnetic field dependence of the lowest 30 eigen values (0~4T) and lowest 60 eigenvalues (4~70T). We can see that eventually all states are becoming degenerate Landau levels for very high magnetic fields. The reason is that the electrons are confined only by the magnetic field and not any longer by the parabolic conduction band edge.

The red line shows the fan of the lowest Landau level at  $1/2\hbar\omega_c$ . The higher lying states (not shown) will collect in the second, third, ..., and higher Landau fans (not shown).

The left part of the figure (black region) contains exactly the same Fock-Darwin spectrum that has been shown in the figure further above (from 0 T to 3.5 T).

Last update: nnnn/nn/nn

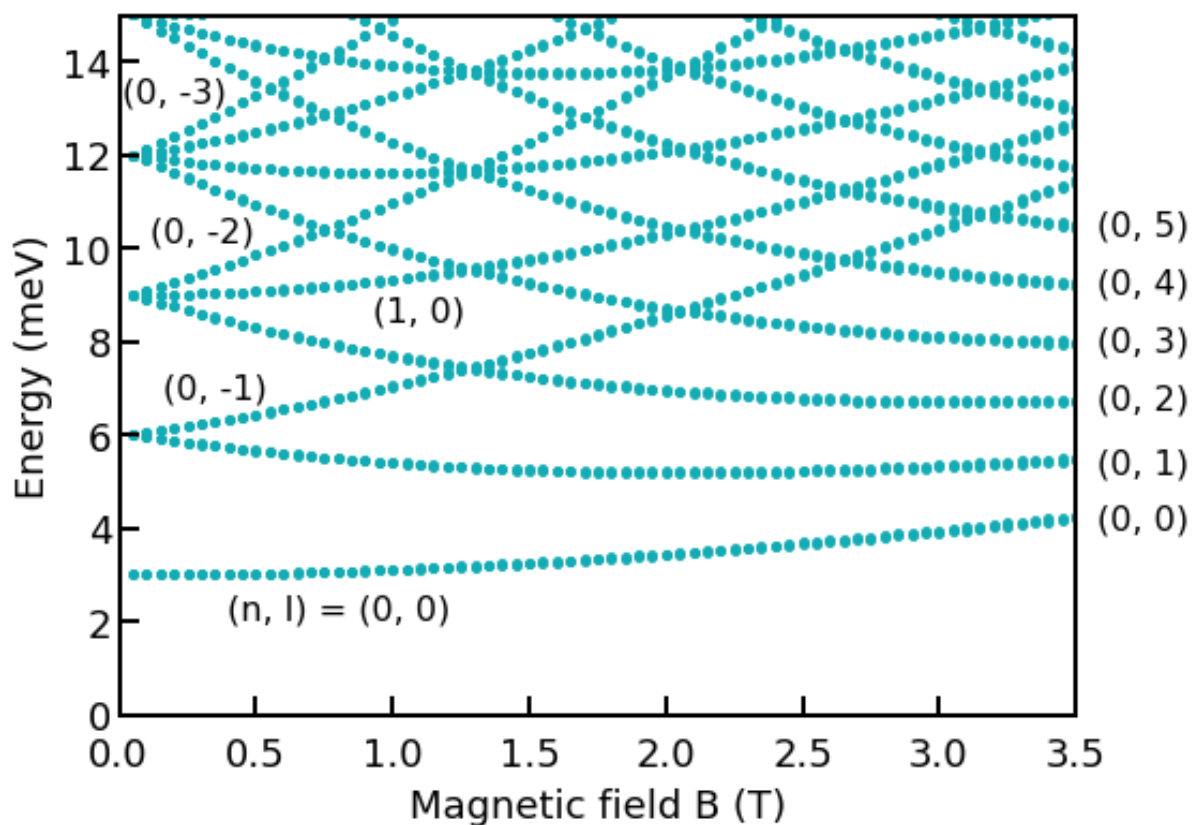
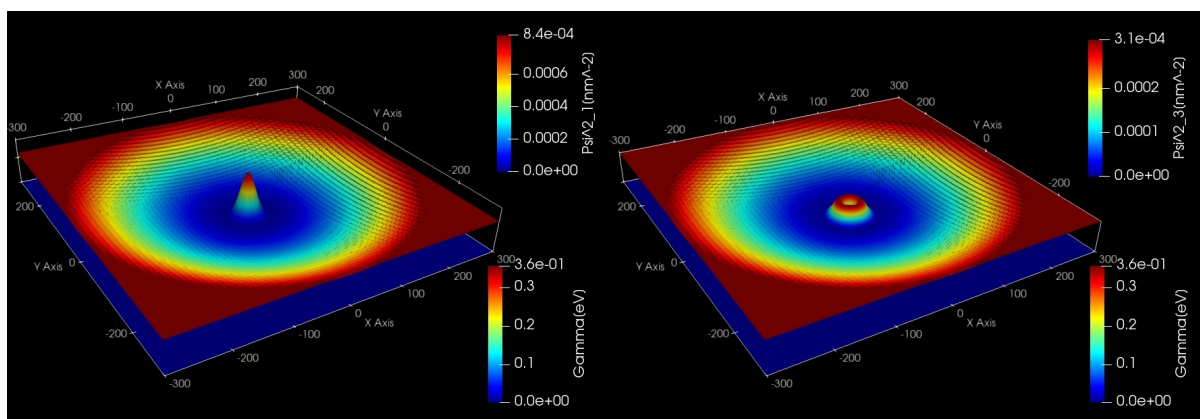


Figure 3.20.19.1: left:  $(n, l) = (0, 0)$  (1st), right:  $(n, l) = (0, 1)$  (2nd)

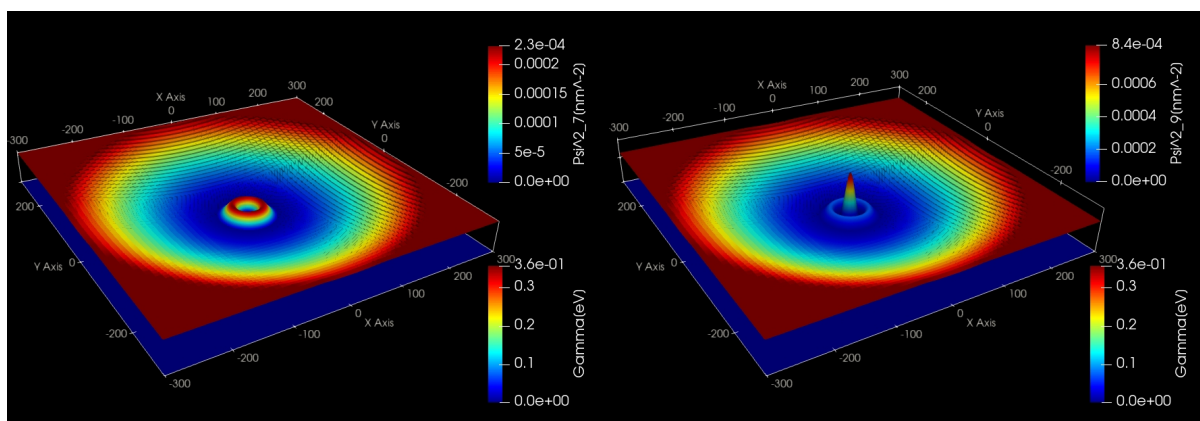


Figure 3.20.19.2: left:  $(n, l) = (0, 2)$  (4th), right:  $(n, l) = (1, 0)$  (5th)

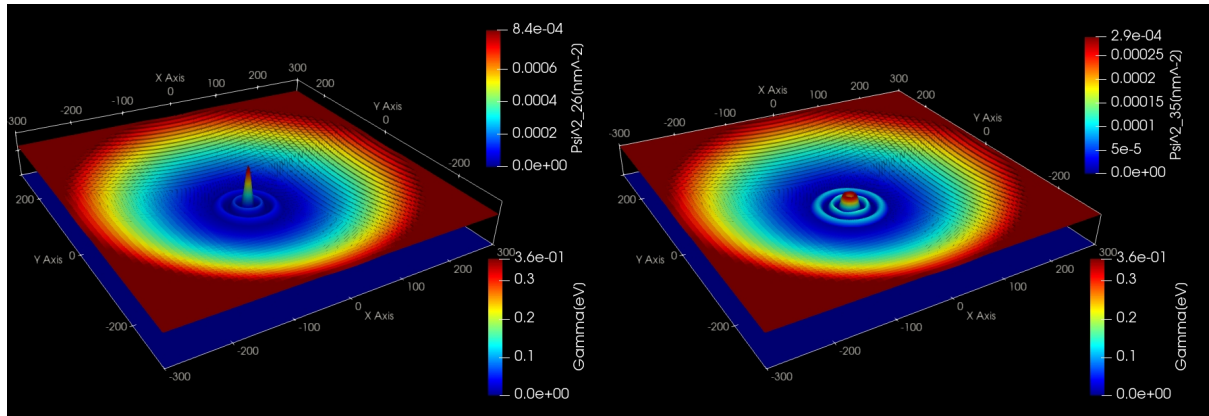
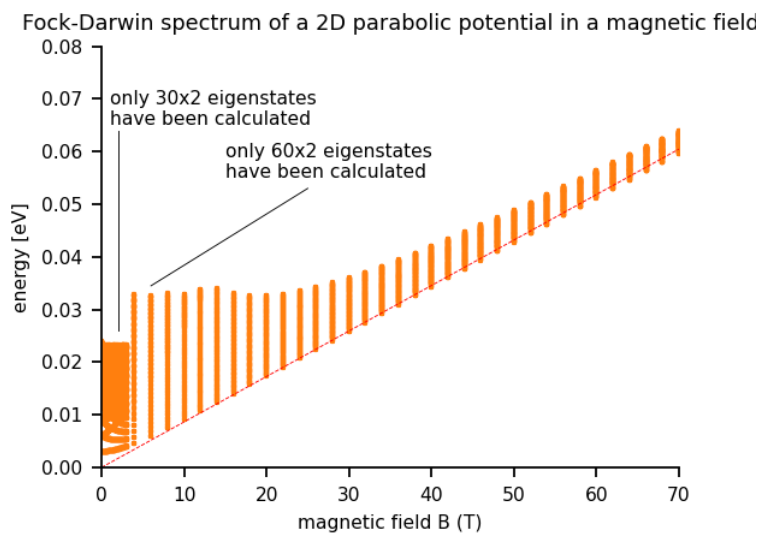


Figure 3.20.19.3: left:  $(n, l) = (2, 0)$  (13th), right:  $(n, l) = (2, 2)$  (18th)



### 3.20.20 Landau levels of a bulk GaAs sample in a magnetic field

In this tutorial, we study the electron energy levels of a bulk GaAs sample that is subject to a magnetic field.

The input files are the followings:

- `2DBulkGaAs_LandauLevels_nn3.in` / `*_nnp.in`

#### Simulation details

The magnetic field is oriented along the z direction. The motion in the z direction is not influenced by the magnetic field and is thus that of a free particle with energies and wave functions given by:

$$E_z = \frac{\hbar^2 k_z^2}{2m^*}$$

$$\psi(z) = \exp(\pm i k_z z)$$

For that reason, we do not include the z direction into our simulation domain, and thus only simulate in the (x,y) plane (two-dimensional simulation).

This plane has the size of 300 nm × 300 nm and consists of GaAs. At the domain boundaries we employ Dirichlet boundary conditions to the Schrödinger equation, i.e. infinite barriers.

We calculate the eigenstates for different magnetic field strengths (1 T, 2 T, 3 T).

```
global{
  ...
  magnetic_field{
    strength = $B # [T]
  }
}
```

```
$magnetic-field

magnetic-field-on           = yes
magnetic-field-strength    = 0.0      ! 0 Tesla = 0 Vs/m2
magnetic-field-direction   = 0 0 1    ! [001] direction

magnetic-field-sweep-active = yes      !
magnetic-field-sweep-step-size = 1.0   ! 1 Tesla = 1 Vs/m2
magnetic-field-sweep-number-of-steps = 3 ! 3 magnetic field sweep steps

output-magnetic-vector-potential = yes ! output of A(x,y,z)
$end_magnetic-field
```

#### Magnetic length and cyclotron frequency

A useful quantity is the magnetic length (or Landau magnetic length) which is defined as:

$$l_B = \left( \frac{\hbar}{m_e^* \omega_c} \right)^{1/2} = \left( \frac{\hbar}{|e| B} \right)^{1/2}$$

It is independent of the mass of the particle and depends only on the magnetic field strength:

- 1 T:  $l_B = 25.6556$  nm
- 2 T:  $l_B = 18.1413$  nm
- 3 T:  $l_B = 14.8123$  nm

In the above formula,  $\omega_c$  is the cyclotron frequency:

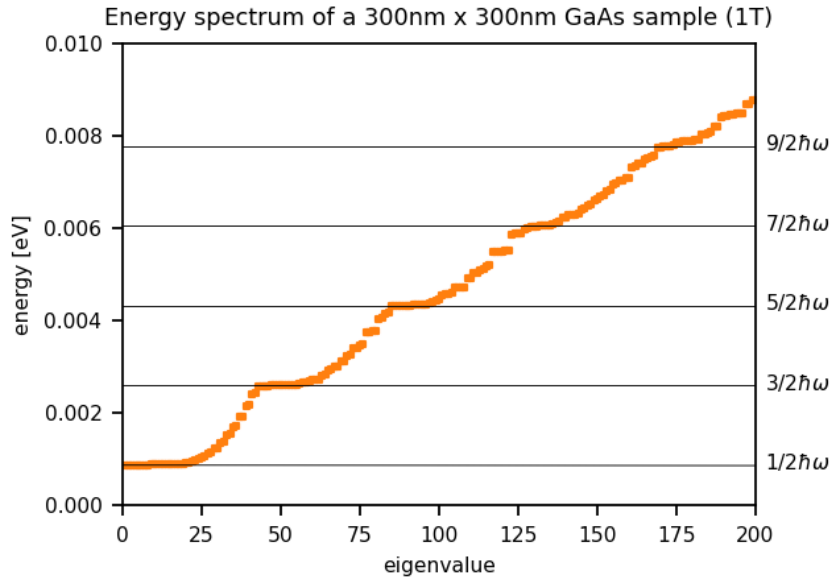
$$\omega_c = \frac{|e|B}{m_e^*}$$

Thus for the electrons in GaAs, where  $m_e^* = 0.067m_0$ , it holds for the different magnetic field strengths:

- 1 T:  $\hbar\omega_c = 1.7279$  meV
- 2 T:  $\hbar\omega_c = 3.4558$  meV
- 3 T:  $\hbar\omega_c = 5.1836$  meV

## Results

The calculated energy spectra for different magnetic fields (1 T, 2 T, 3 T) are as follows:



## Landau levels

The Landau levels are analytically given by

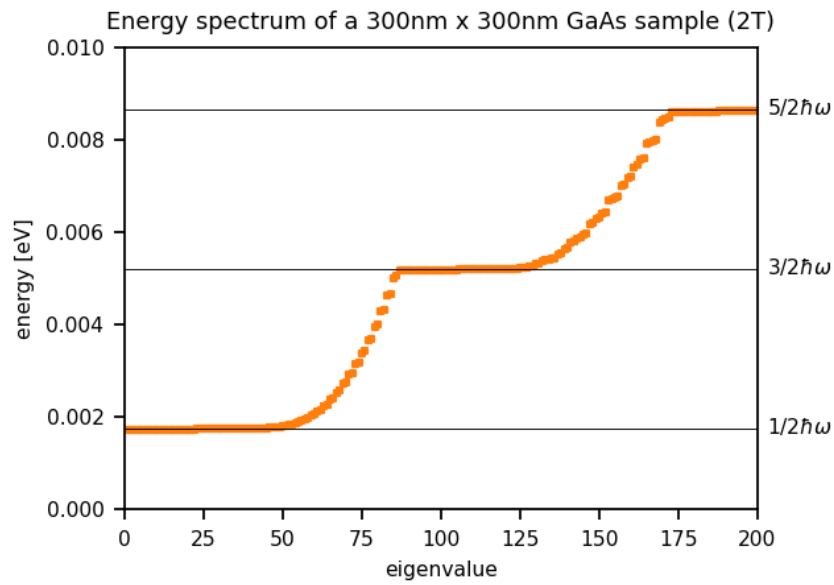
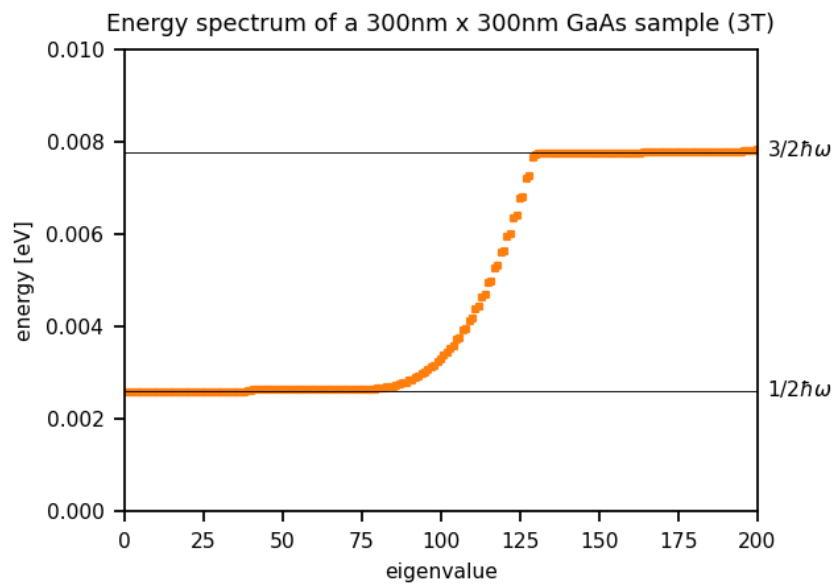
$$E_n = \left(n - \frac{1}{2}\right) \hbar\omega_c$$

where  $n = 1, 2, 3, \dots$

The number of states for each Landau level can be calculated as follows (see P.Y. Yu, M. Cardona, Fundamentals of Semiconductors, p. 536, 3rd ed.):

$$N = L_x L_y \frac{|e|B}{h} = \frac{L_x L_y}{2\pi l_B^2}$$

where  $L_x$  and  $L_y$  are the lengths in the x and y directions (300 nm in this example) and  $l_B$  is the magnetic length. Here we ignore spin.





- $N(1\text{ T}) = 21.76 \sim \mathbf{22}$  states per Landau level (in the figure above: 42)
- $N(2\text{ T}) = 43.52 \sim \mathbf{44}$  states per Landau level (in the figure above: 86)
- $N(3\text{ T}) = 65.29 \sim \mathbf{65}$  states per Landau level (in the figure above: 130)

When `magnetic_field{}` is specified, `nextnano++` calculates the [Schrödinger-Pauli equation](#), which takes into account the spin. Since the interaction energy between the spin and magnetic field is small compared to the separation of Landau levels, the number of states per Landau level calculated by `nextnano++` is almost double of the analytical result that ignores the spin.

### Energy eigenvalues

For the calculations, we used the symmetric gauge  $A = -\frac{1}{2}r \times B = \frac{1}{2}B \times r$  leading to the following energies (see J.H. Davies, The Physics of Low-Dimensional Semiconductors, p. 222):

$$E_{n,l} = \left( n + \frac{1}{2}l + \frac{1}{2}|l| - \frac{1}{2} \right) \hbar\omega_c$$

One can see that all states having a negative value of  $l$  are degenerate with the states with  $l = 0$ , i.e. the allowed energies are independent of  $l$  if  $l < 0$  (for the same  $n$ ). The energies increase if  $l$  increases (for  $l > 0$  and for the same  $n$ ).

---

*Last update: nnnn/nn/nn*

### 3.20.21 Electron wave functions in a cylindrical well (2D Quantum Corral)

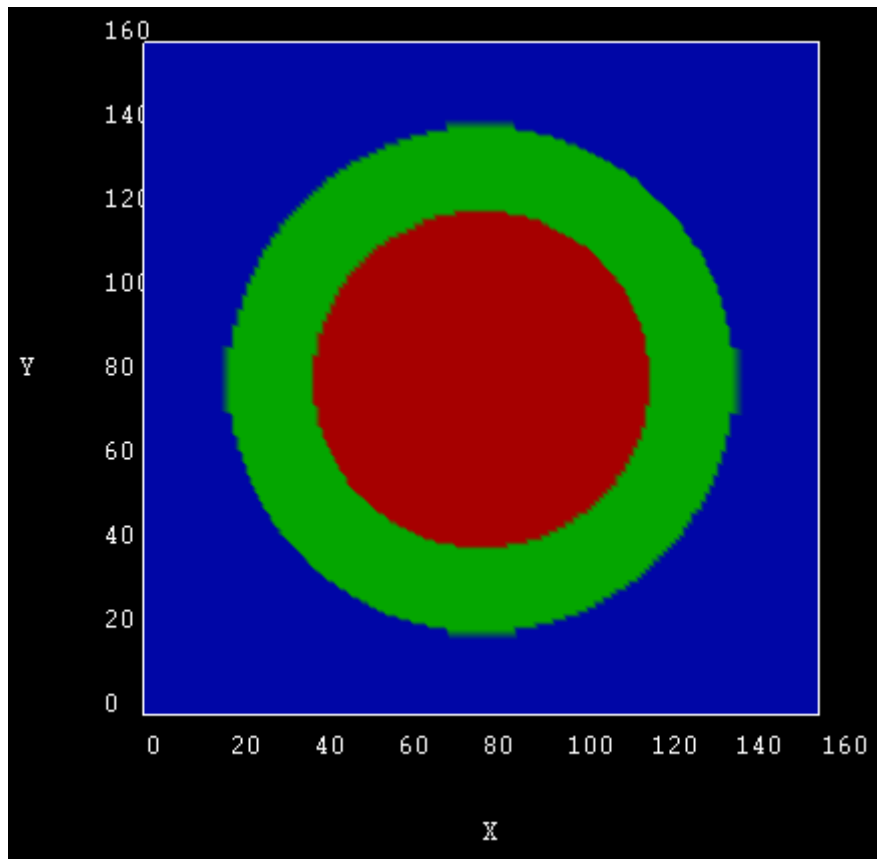
In this tutorial we demonstrate 2D simulation of a cylindrical quantum well. We will see the electron eigenstates and their degeneracy.

Input files used in this tutorial are the followings:

- `2DQuantumCorral_nn3.in` / `*_nnp.in`

### Structure

- A cylindrical InAs quantum well (diameter 80 nm) is surrounded by a cylindrical GaAs barrier (20 nm) which is surrounded by air. The whole sample is 160 nm x 160 nm.
- We assume **infinite** GaAs barriers. This can be achieved by a circular quantum cluster with **Dirichlet** boundary conditions, i.e. the wave function is forced to be zero in the GaAs barrier.
- The electron mass of InAs is assumed to be isotropic and parabolic ( $m_e = 0.026m_0$ ).
- Strain is not taken into account.



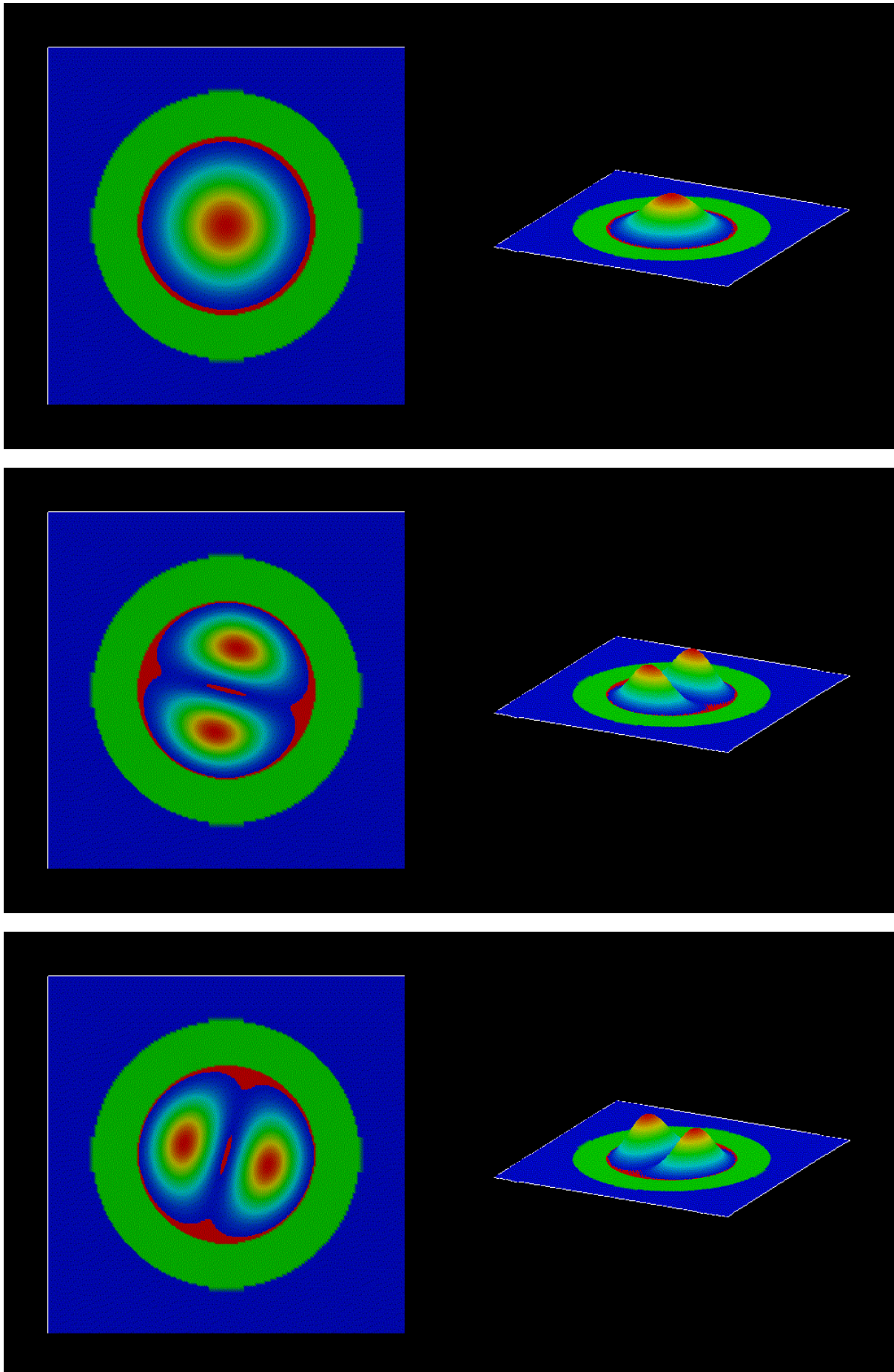
### Simulation outcome

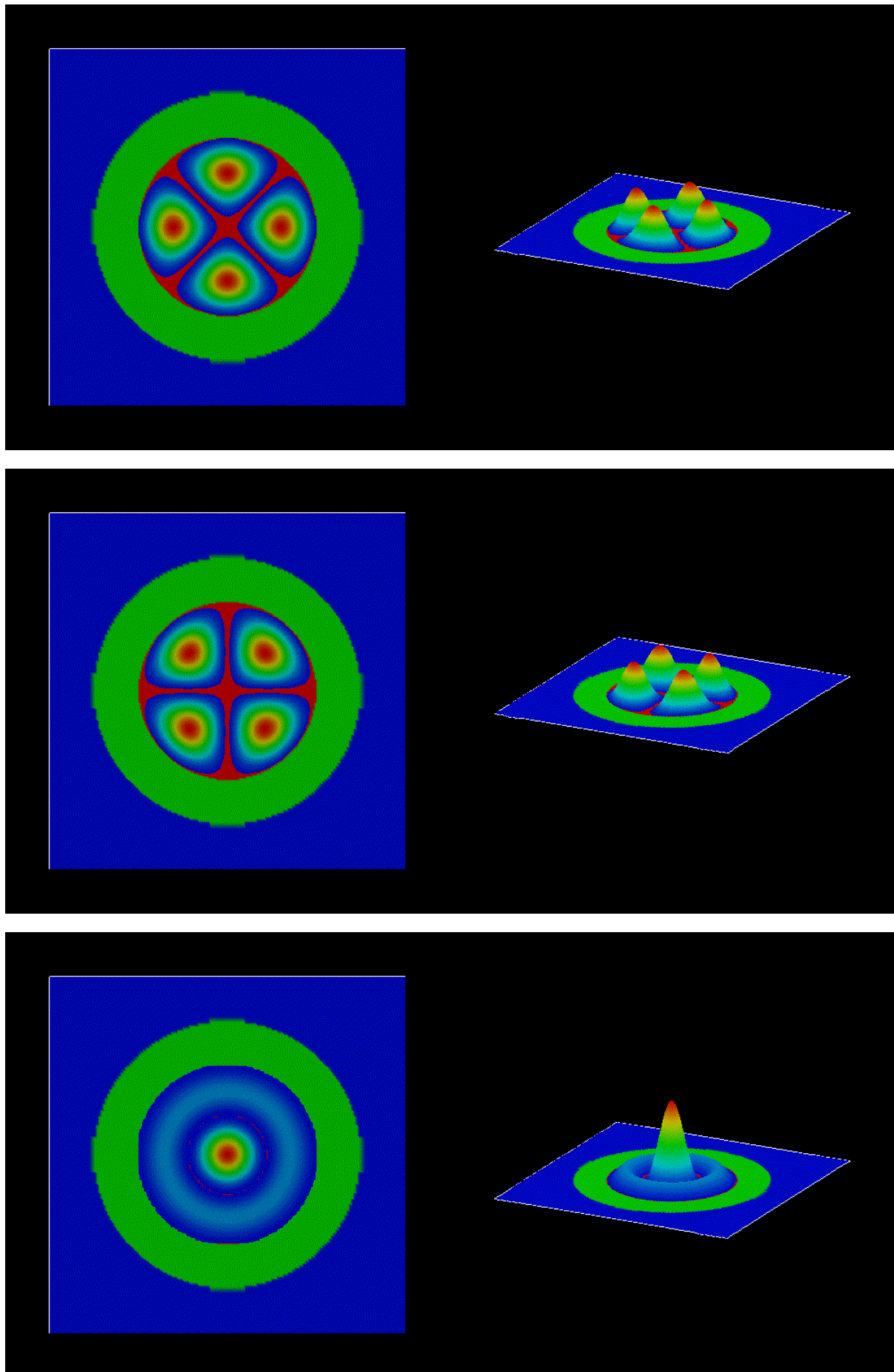
#### Electron wave functions

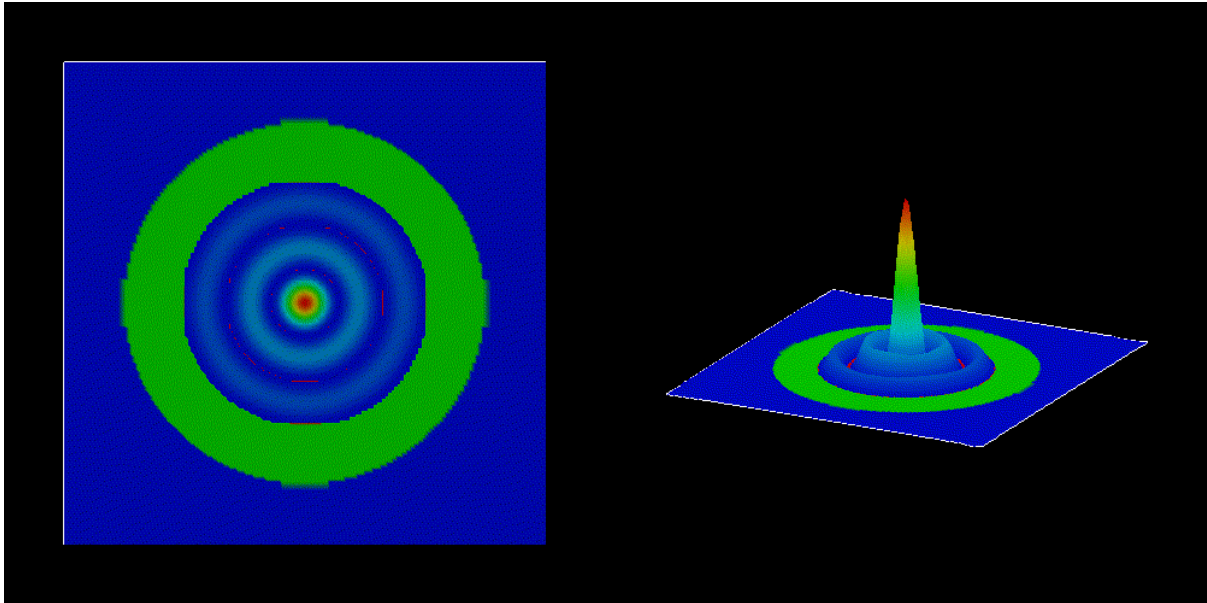
The size of the quantum cluster is a circle of diameter  $2a = 80$  nm.

The following figures shows the square of the electron wave functions (i.e.  $\psi^2$ ) of the corresponding eigenstates. They were calculated within the effective-mass approximation (single-band) on a rectangular finite-differences grid.

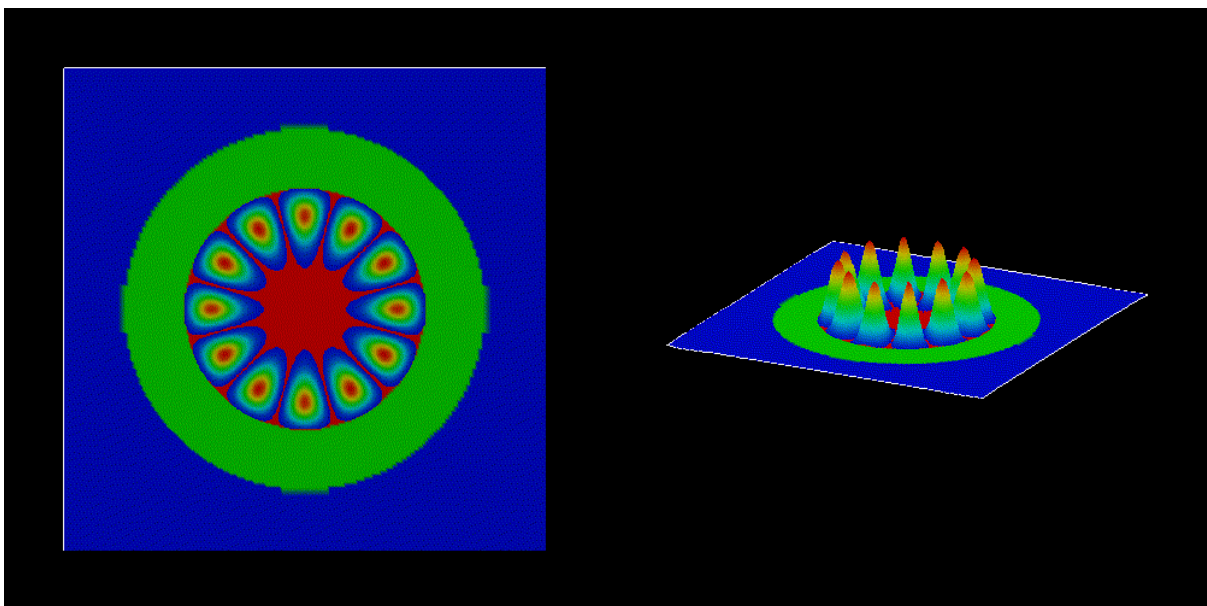
- 1st eigenstate,  $(n, l) = (1, 0)$
- 2nd eigenstate,  $(n, l) = (1, 1)$
- 3rd eigenstate,  $(n, l) = (1, -1)$
- 4th eigenstate,  $(n, l) = (1, 2)$
- 5th eigenstate,  $(n, l) = (1, -2)$
- 6th eigenstate,  $(n, l) = (2, 0)$
- 15th eigenstate,  $(n, l) = (3, 0)$







- 20th eigenstate,  $(n, l) = (1, 6)$

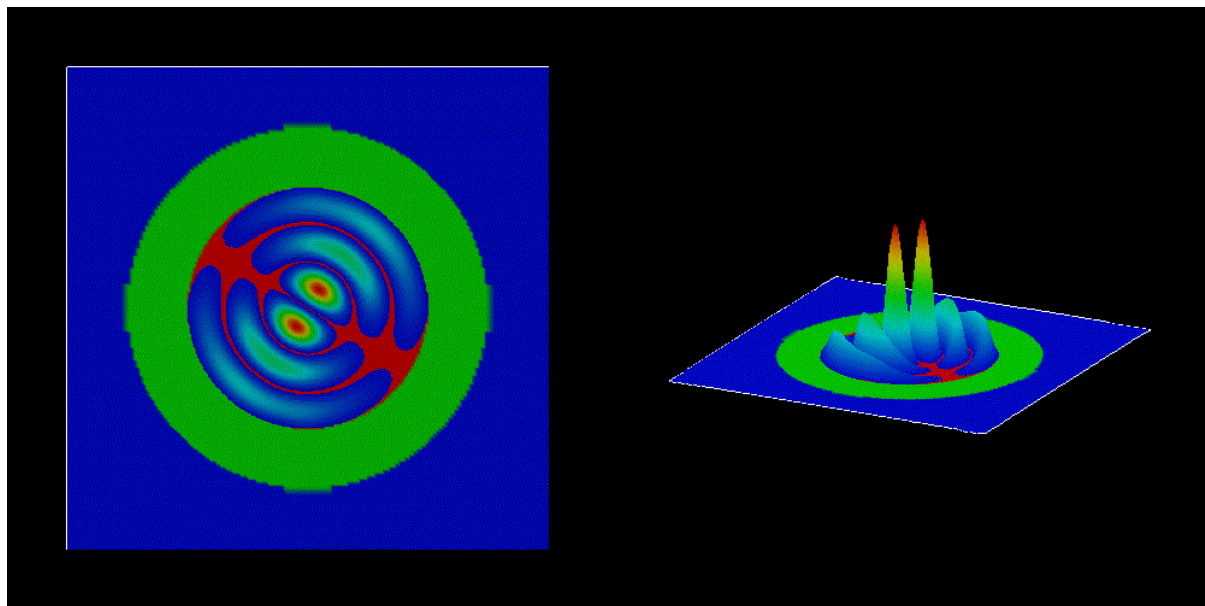


- 22th eigenstate,  $(n, l) = (3, 1)$

The parameters of the quantum corral are the followings:

- radius:  $a = 40$  nm
- $m_e = 0.026m_0$
- $V(r) = 0$  for  $r < a$
- $V(r) = \infty$  for  $r > a$





The analytical solution of the eigenstates of this quantum well is:

$$\psi_{n,l}(r, \theta) \propto J_l\left(\frac{j_{l,n}r}{a}\right) [A \cos(l\theta) + B \sin(l\theta)] \quad (3.20.21.1)$$

where

- $J_l(x)$  is the Bessel function of the first kind (We cite them for  $l = 0, 1, 2$  below.)
- $j_{l,n}$  is its zero point i.e.  $J_l(j_{l,n}) = 0$  and  $n = 1, 2, \dots$
- $A, B$  are constant
- $l = 0, \pm 1, \pm 2, \dots$

The corresponding eigenenergies are:  $E_{nl} = \frac{\hbar^2 j_{l,n}^2}{2m_e a^2}$

The Quantum number  $n$  comes from the boundary condition  $\psi(a, \theta) = 0$ . The requirement that  $\psi$  has the same value at  $\theta = 0$  and  $2\pi$  leads to the quantum number  $l$ . In the above figures of the eigenstates, we can know them through the following relations:

- (the number of zero points in the radial direction) =  $n$
- (the number of zero points in the circumferential direction)/2 =  $|l|$

## Energy spectrum

The following figure shows the energy spectrum of the quantum corral. (The zero of energy corresponds to the InAs conduction band edge.)

The two-fold degeneracies of the states

- (2, 3), (4, 5), (7, 8), (9, 10), (11, 12), (13, 14), (16, 17), (18, 19), (20, 21), (22, 23), (24, 25), (26, 27), (28, 29), (31, 32), (33, 34), (35, 36), (37, 38), (39, 40)

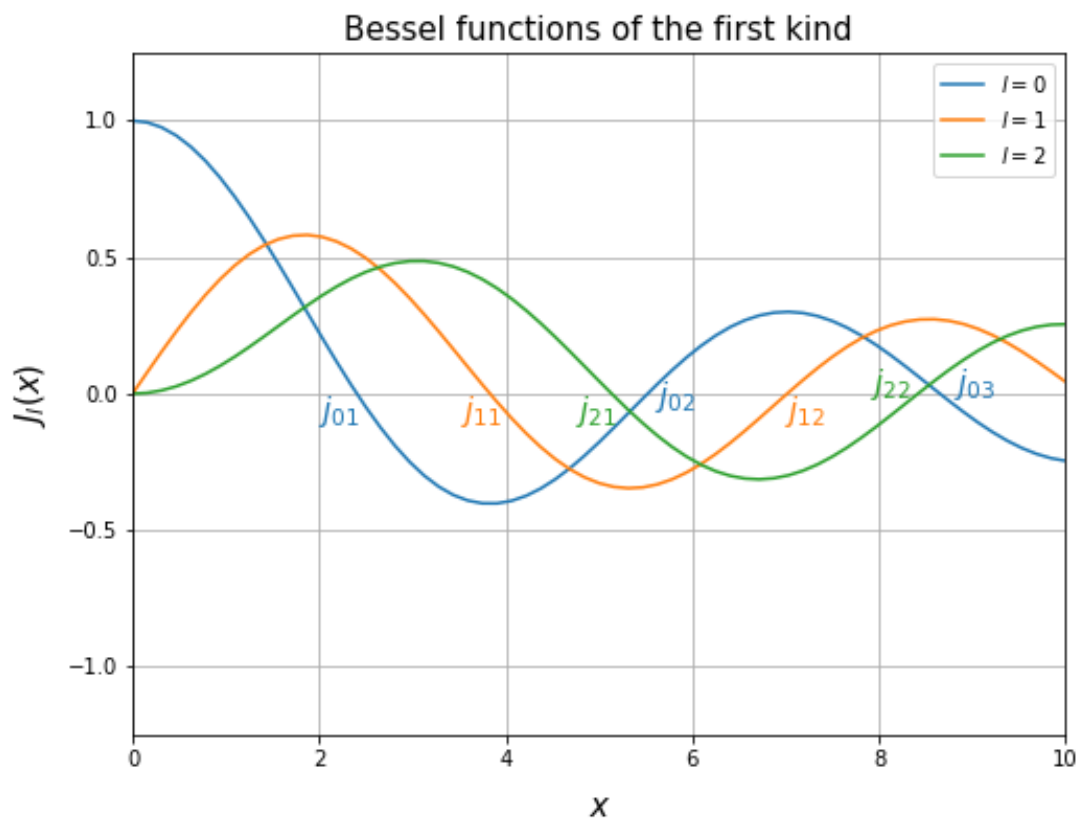
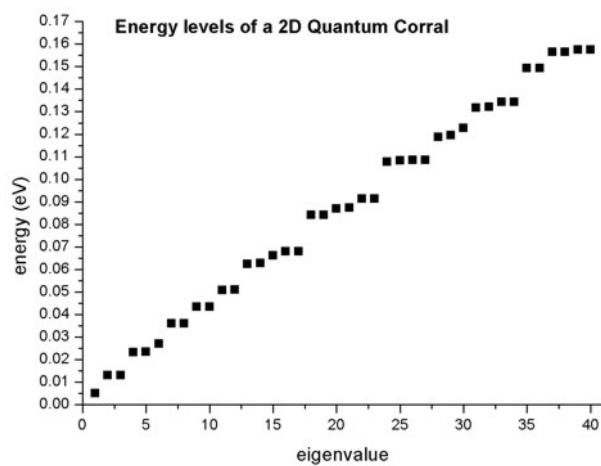


Figure 3.20.21.1: Bessel functions of the first kind for  $l = 0, 1, 2$  generated by scipy.



corresponds to  $|l| \geq 1$ . On the other hand, the non-degenerate energy eigenvalues corresponds to  $l = 0$

The analytical energy values are:  $E_{nl} = \frac{\hbar^2 j_{l,n}^2}{2m_e a^2}$ .

There is a formula to approximate  $j_{l,n}$ :  $j_{l,n} = (n + \frac{1}{2}|l| - \frac{1}{4})\pi$  which is accurate as  $n \rightarrow \infty$ .

Here we describe the comparison between the analytical values, approximate values, *nextnano++* results and *nextnano<sup>3</sup>* results.

|     | $[n, l]$ | $j_{l,n}$ | $j_{l,n}$<br>(ap-<br>prox.) | $E_{n,l}$<br>[eV] | $E_{n,l}$ [eV] (ap-<br>prox.) | $E_{n,l}$<br>( <i>nextnano++</i> )<br>[eV] | $E_{n,l}$<br>( <i>nextnano<sup>3</sup></i> )<br>[eV] |
|-----|----------|-----------|-----------------------------|-------------------|-------------------------------|--------------------------------------------|------------------------------------------------------|
| 1st | [1, 0]   | 2.405     | $0.75\pi \simeq 2.356$      | 0.00530           | 0.00508                       | 0.00510                                    | 0.00511                                              |
| 2nd | [1, 1]   | 3.832     | $1.25\pi \simeq 3.927$      | 0.01345           | 0.01412                       | 0.01294                                    | 0.01298                                              |
| 3rd | [1, -1]  | 3.832     | $1.25\pi \simeq 3.927$      | 0.01345           | 0.01412                       | 0.01294                                    | 0.01298                                              |
| 4th | [1, 2]   | 5.136     | $1.75\pi \simeq 5.498$      | 0.02416           | 0.02768                       | 0.02320                                    | 0.02325                                              |
| 5th | [1, -2]  | 5.136     | $1.75\pi \simeq 5.498$      | 0.02416           | 0.02768                       | 0.02329                                    | 0.02325                                              |
| 6th | [2, 0]   | 5.520     | $1.75\pi \simeq 5.498$      | 0.02791           | 0.02767                       | 0.02685                                    | 0.02693                                              |
| 7th | [2, 1]   | 7.016     | $2.25\pi \simeq 7.069$      | 0.04508           | 0.04574                       | 0.03584                                    | 0.03597                                              |

Further details about the analytical solution of the cylindrical quantum well with infinite barriers can be found in:

**The Physics of Low-Dimensional Semiconductors - An Introduction**

John H. Davies

Cambridge University Press (1998)

---

*Last update: nnnn/nn/nn*

### 3.20.22 Electron wave functions of a 2D slice of a Triple Gate MOSFET

In this tutorial we demonstrate the 2D simulation of a Triple Gate MOSFET. We solve the **two-dimensional Schrödinger and Poisson equations self-consistently** for a 2D slice. We would see the difference between the electron densities calculated quantum mechanically and classically.

The relevant input files are as follows:

- *2DSi\_TGMOS\_2Dcut\_atGate\_cl.in* / *\*\_nnp.in*
- *2DSi\_TGMOS\_2Dcut\_atGate\_qm.in* / *\*\_nnp.in*
- *2DSi\_TGMOS\_2Dcut\_atGate\_qm\_iso.in* / *\*\_nnp.in*
- *3DSi\_TGMOS\_5 nm\_SD0V\_G0V\_qm.in*
- *3DSi\_TGMOS\_5 nm\_SD0V\_G05V\_qm.in*



If you want to obtain the input files that are used within this tutorial, please contact support [at] nextnano.com.

The values and graphs described in this tutorial are the result of *nextnano<sup>3</sup>*.

## 2D Simulation

### Structure

A Triple Gate MOSFET is a **nanowire** if the dimensions along the x and y directions are only a few nanometers, thus quantization effects have to be taken into account. The structure considered is as follows:

- The Si channel has a rectangular shape with a width of 5 nm and a height of 5 nm.
- The Si channel is surrounded by SiO<sub>2</sub> (thickness 1.5 nm).

The Si/SiO<sub>2</sub> nanowire is surrounded by a Gate (at the left and right side, and at the top).

The following schematic shows a 2D slice of a 3D Triple Gate MOSFET.

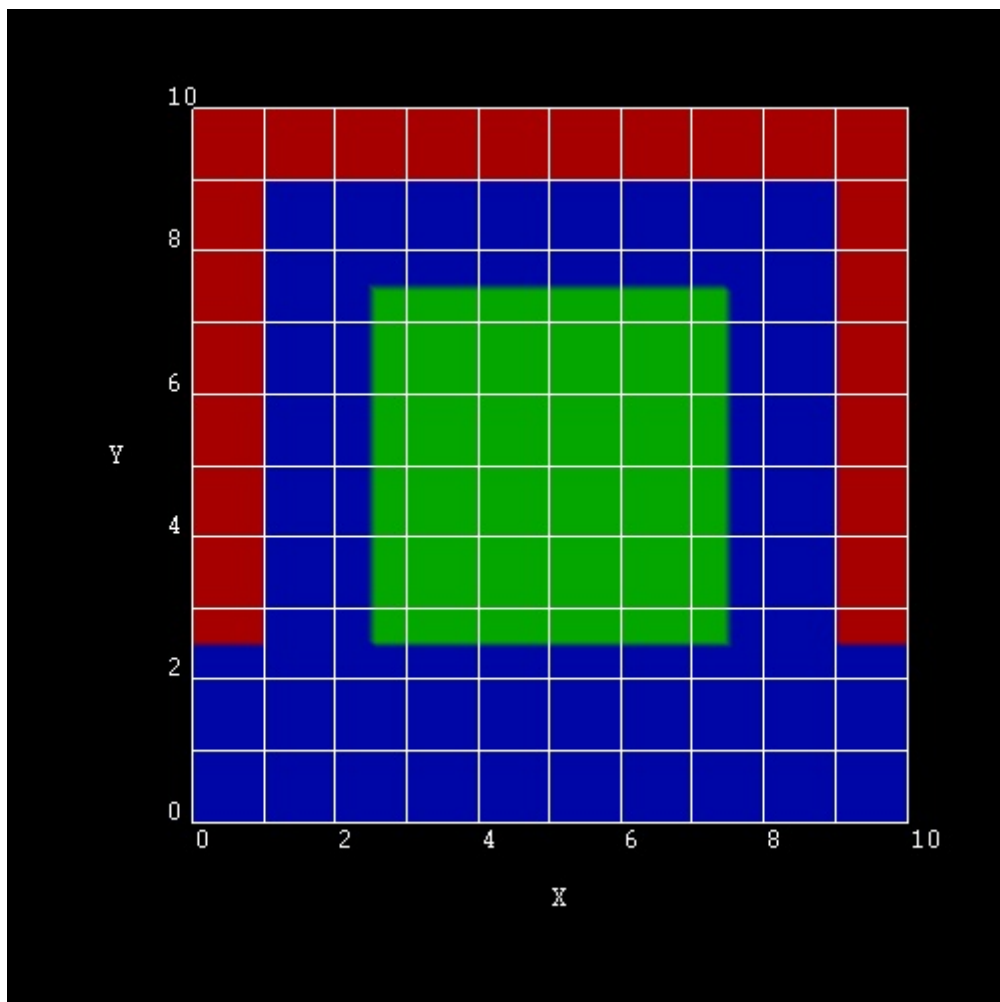


Figure 3.20.22.1: 2D slice of a 3D Triple Gate MOSFET

## Simulation Details

In this tutorial we will only simulate this 2D slice and not the whole 3D structure.

We apply a voltage of 0.5 V to the Gates and solve the **two-dimensional Schrödinger and Poisson equations self-consistently** (including the SiO<sub>2</sub> region).

There are six equivalent conduction band minima in silicon (Delta valleys). Since the constant energy surfaces are ellipsoids, the mass tensor has the following two kinds of effective masses:

- The longitudinal mass is  $0.916m_0$ .
- The transversal mass is  $0.190m_0$  (2 directions).

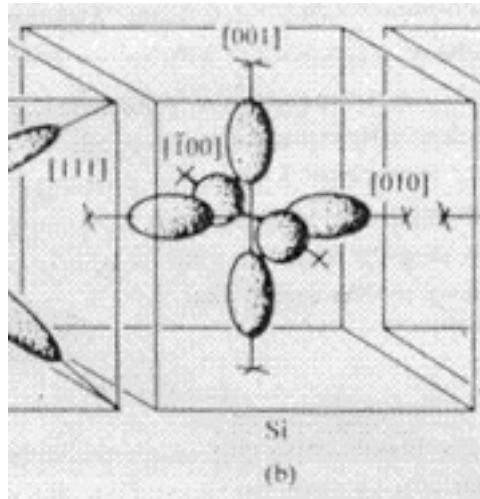


Figure 3.20.22.2: constant energy surface of Si conduction band

Therefore, we need to solve **three** 2D Schrödinger equations with different effective mass tensor orientations.

Our Schrödinger equations are numbered X1, X2, X3 in *nextnano++* or deg1, deg2, deg3 in *nextnano<sup>3</sup>*.

- X1/deg1: a)  $m_{xx} = m_l = 0.916m_0$ ,  $m_{yy} = m_t = 0.190m_0$
- X2/deg2: b)  $m_{xx} = m_t = 0.190m_0$ ,  $m_{yy} = m_l = 0.916m_0$
- X3/deg3: c)  $m_{xx} = m_{yy} = m_t = 0.190m_0$

The potential  $E_c(x, y)$  that enters the Schrödinger equation is the same in these three cases.

### Note

The cases a) and b) are not identical (i.e. degenerate) because the potential is not symmetric with respect to exchanging x and y coordinates.

The following keyword and specifier can be used to output the effective mass tensors ( $1/m_{ij}$ ).

```
# nextnano++
output{
  ...
  material_parameters{
    ...
    charge_carrier_masses{
      boxes = yes
    }
  }
}
```

```
! nextnano3
$output-1-band-schroedinger
...
effective-mass-tensor = yes
```

## Results

### Electron wave functions $|\psi^2|$

- 2DSi\_TGMOS\_2Dcut\_atGate\_qm\_nnp.in, \*\_nn3.in

The lowest eigenstates for the cases a), b) and c) are the following:

- X1/deg1: a)  $m_{xx} = m_l = 0.916m_0$ ,  $m_{yy} = m_t = 0.190m_0$

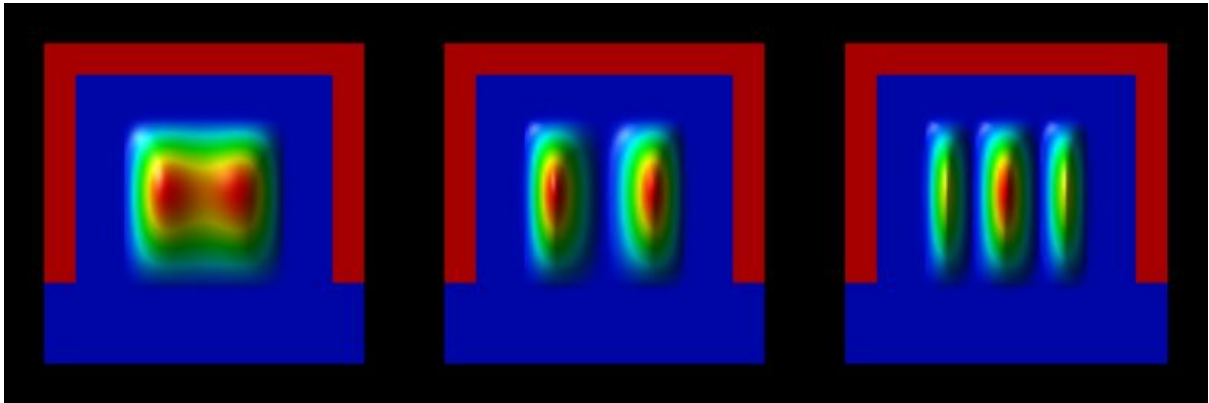


Figure 3.20.22.3:  $E_{1,X1} = -26$  meV,  $E_{2,X1} = -1$  meV,  $E_{3,X1} = 77$  meV

Here, the heavier mass is along the x direction, and the lighter mass along the y direction. The energy spacing between the two lowest subbands is about 24 meV. The eigenvalues are contained in *bias\_00000/Quantum/energy\_spectrum\_quantum\_region\_X1\_00000.dat/Schroedinger\_1band/ev2D\_cb003\_qc001\_sg0*

- X2/deg2: b)  $m_{xx} = m_t = 0.190m_0$ ,  $m_{yy} = m_l = 0.916m_0$

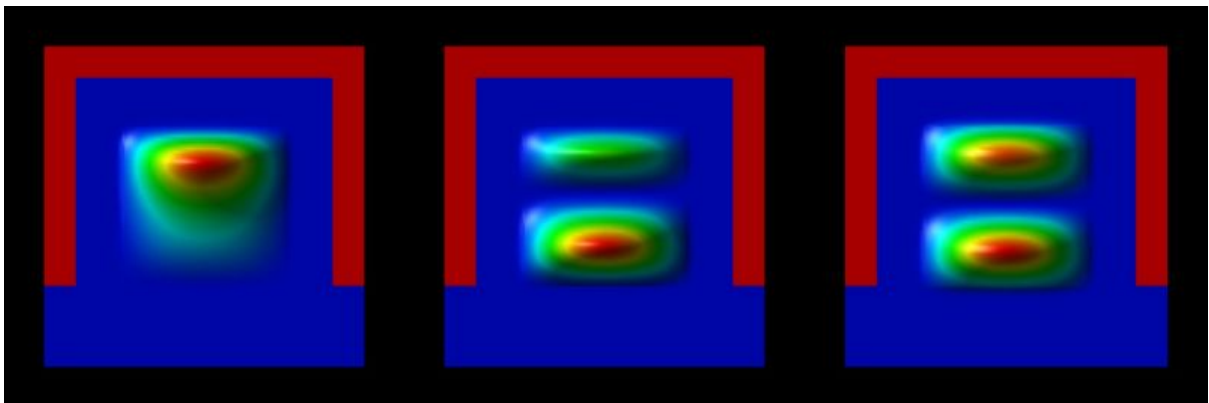


Figure 3.20.22.4:  $E_{1,X2} = -28$  meV,  $E_{2,X2} = 6$  meV,  $E_{3,X2} = 82$  meV

Here, the lighter mass is along the x direction, and the heavier mass along the y direction. The energy spacing between the two lowest subbands is about 35 meV. The eigenvalues are contained in *bias\_00000/Quantum/energy\_spectrum\_quantum\_region\_X2\_00000.dat/Schroedinger\_1band/ev2D\_cb003\_qc001\_sg0*

- X3/deg3: c)  $m_{xx} = m_{yy} = m_t = 0.190m_0$

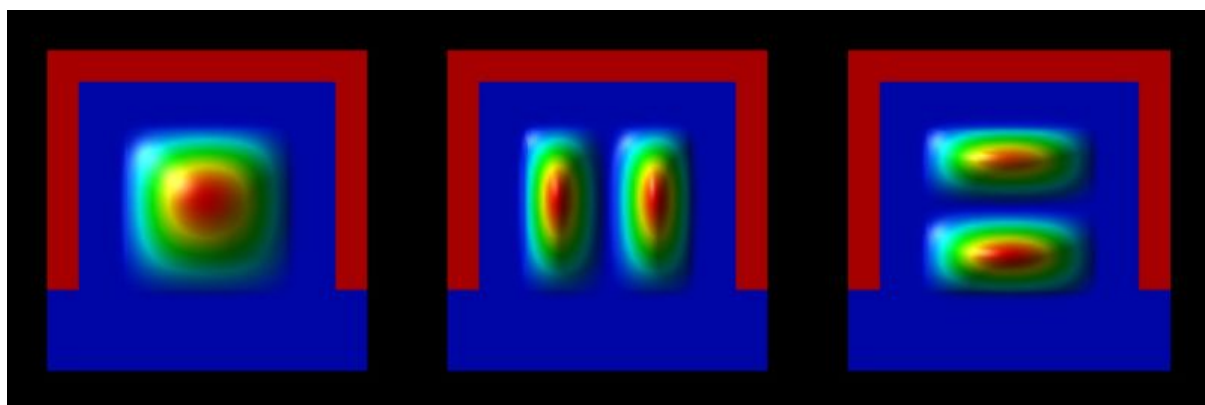


Figure 3.20.22.5:  $E_{1,X3} = 16$  meV,  $E_{2,X3} = 167$  meV,  $E_{3,X3} = 173$  meV

These eigenvalues have the lighter mass in the x and y directions. Consequently, their energies are much higher than in the other two Schrödinger equations. The energy spacings between the lowest subbands is of the order 140-150 meV. The eigenvalues are contained in *bias\_00000/Quantum/energy\_spectrum\_quantum\_region\_X3\_00000.dat/Schroedinger\_1band/ev2D\_cb003\_qc001\_sg0*

(Compare the wave functions and the energies with the isotropic case as discussed further below.)

### Electron density

The resulting electron density has the following shape, see Figure 3.20.22.6:

The units are  $1 \times 10^{18} \text{ cm}^{-3}$ . The density has been calculated by occupying the eigenstates with respect to the Fermi level which is at 0 eV. Note that the quantum mechanical density is close to zero near the Si/SiO<sub>2</sub> interfaces because the wave functions tend to zero at the SiO<sub>2</sub> barriers.

Figure 3.20.22.7 shows the same quantum mechanical electron density together with two slices through the conduction band edges. The units are in eV and the conduction band offset between SiO<sub>2</sub> and Si is 3.1 eV. At the gates, the conduction band edge is set to -0.5 eV, representing the applied bias of 0.5 eV. One can clearly see that **for silicon** in the middle of the nanowire the conduction band has its highest value and its lowest value close at the Si/SiO<sub>2</sub> interface.

If one had neglected the effect of quantum confinement, then the resulting **classical electron density** would have peaks near the Si/SiO<sub>2</sub> interfaces as is shown in Figure 3.20.22.8.

- *2DSi\_TGMOS\_2Dcut\_atGate\_cl\_nnp.in, \*\_mm3.in*

Obviously, a realistic calculation of such transistors cannot be based on classical densities. The full 2D (or better 3D) Schrödinger equations have to be solved. The IV characteristics of such a quantum-mechanically calculated Triple Gate MOSFET transistor will be discussed in another tutorial.

### Isotropic electron masses

Very often, for simplicity, an isotropic electron mass for the Schrödinger equation is assumed. E.g. the DOS (density of states) electron mass of Si in the Delta minima can be calculated as follows:

$$m_{e,DOS}^* = (m_l m_t^2)^{1/3} = (0.9160 \cdot 19^2)^{1/3} m_0 = 0.321 m_0$$

In this case, only **one** Schrödinger equation has to be solved (in contrast to three equations as described above).

The wave functions and energies in this case are:

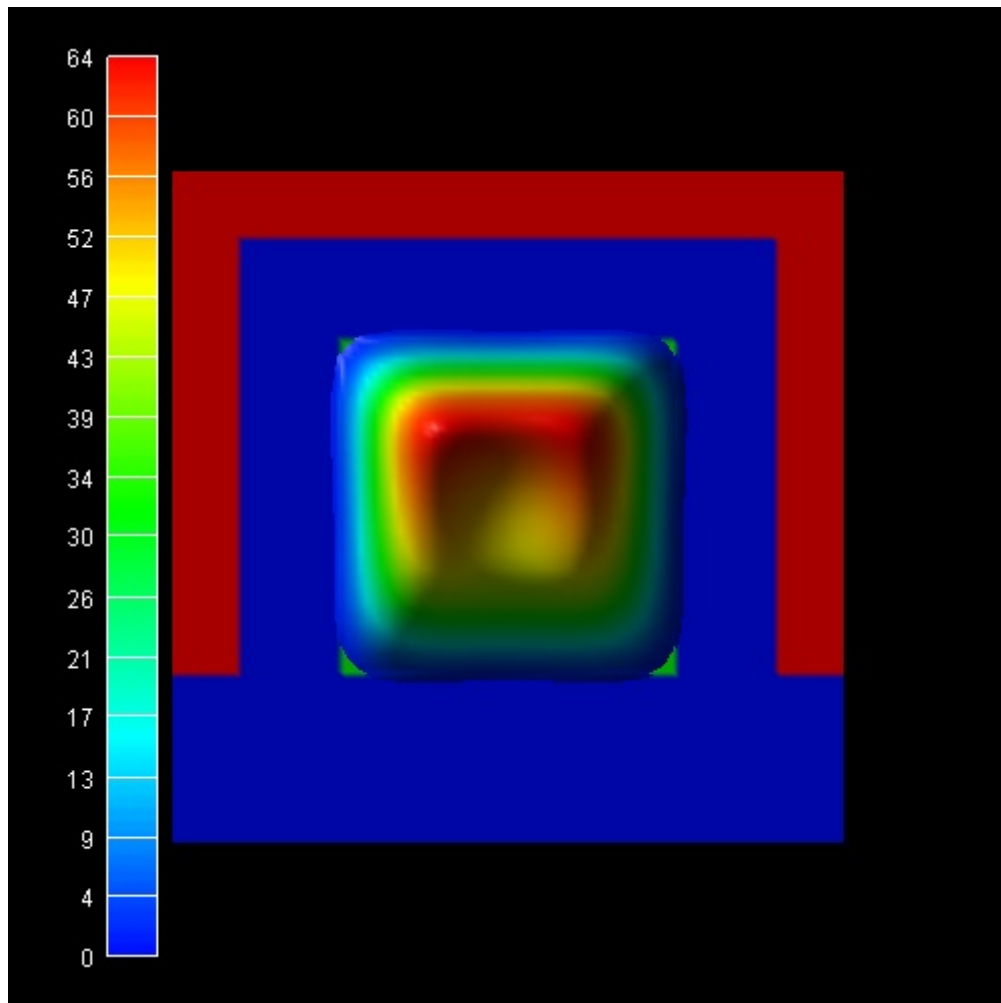


Figure 3.20.22.6: electron density

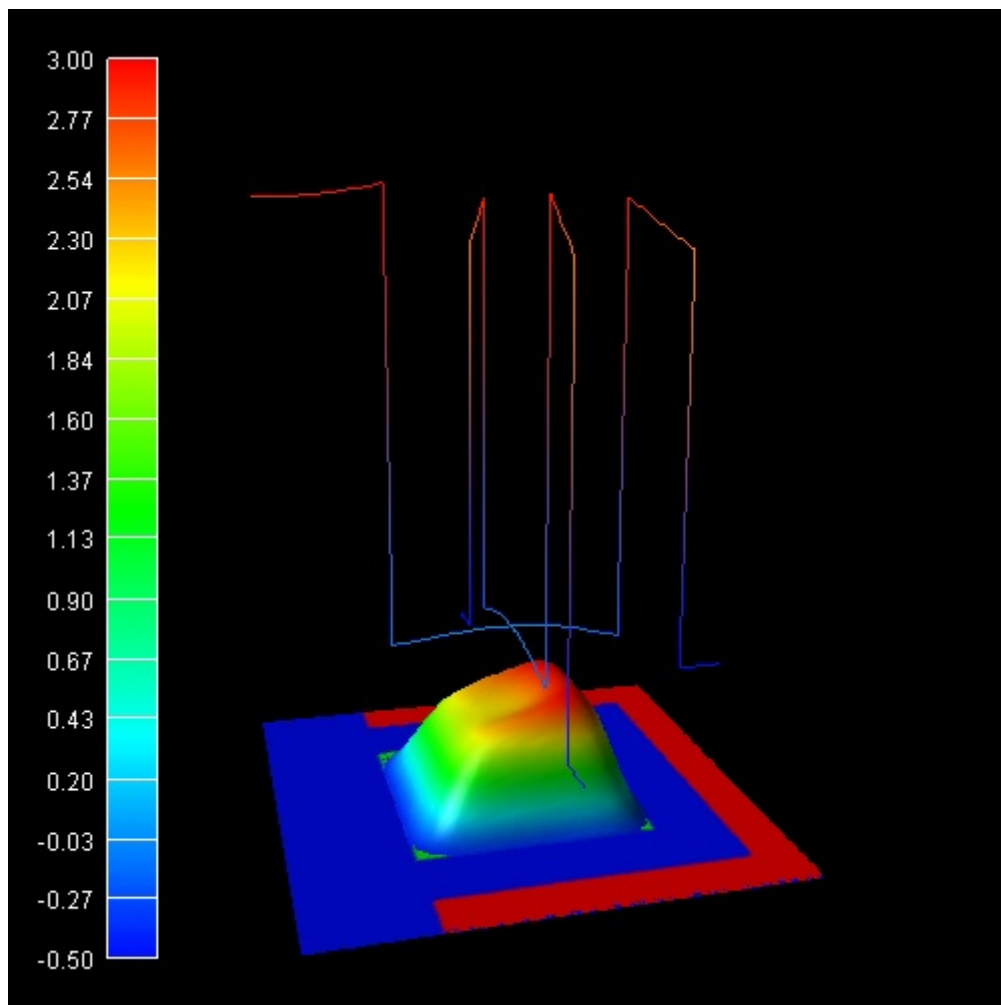


Figure 3.20.22.7: electron density and slices through the conduction band edges

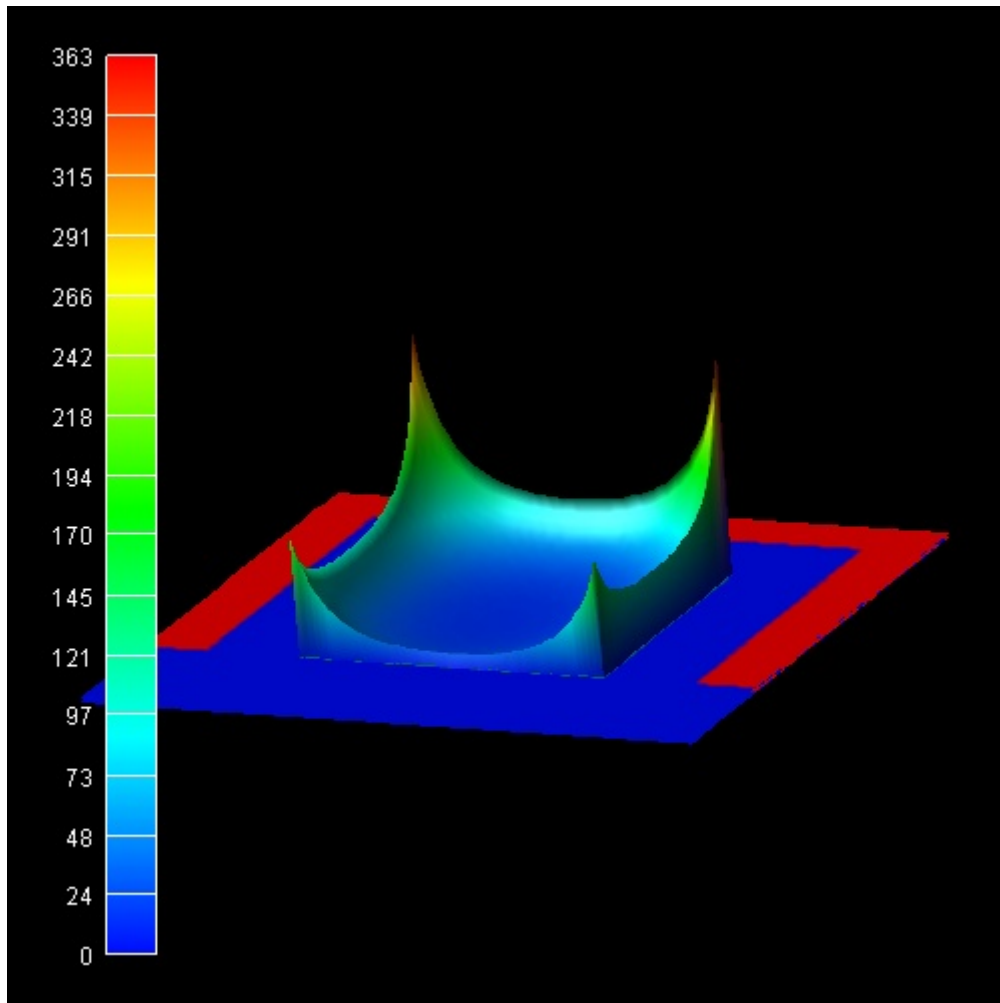


Figure 3.20.22.8: classical electron density calculated by *2DSi\_TGMOS\_2Dcut\_atGate.cl.in*

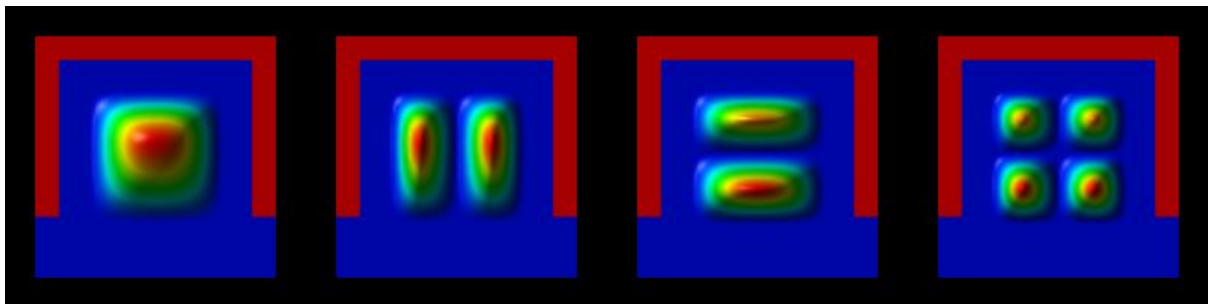


Figure 3.20.22.9:  $E_1 = -21$  meV,  $E_2 = 69$  meV,  $E_3 = 75$  meV,  $E_3 = 166$  meV

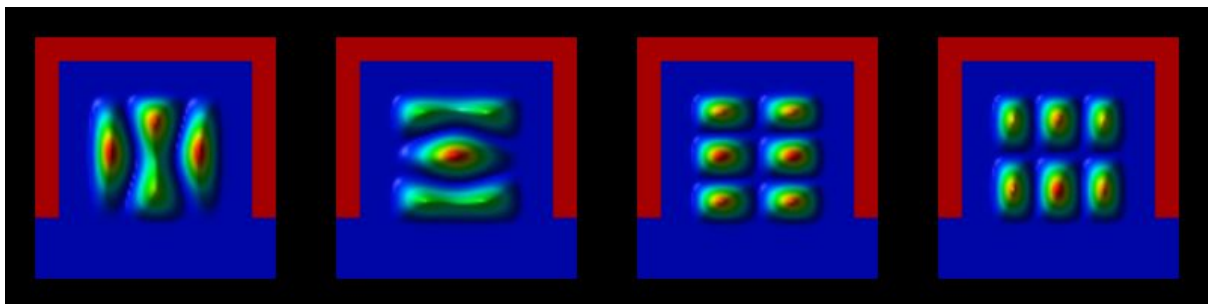


Figure 3.20.22.10:  $E_5 = 262$  meV,  $E_6 = 270$  meV,  $E_7 = 360$  meV,  $E_8 = 360$  meV

- $m_{xx} = m_{yy} = m_{DOS} = 0.321m_0$

The wave functions  $|\psi^2|$  look very similar as in the case of “X3/deg3: c)” (see above) where the masses are isotropic in the (x,y) plane but here, the energy spacings between different subbands are smaller (around 90-100 meV) because the DOS mass is larger than the transversal masses.

The eigenvalues are contained in *bias\_00000/Quantum/energy\_spectrum\_quantum\_region\_X3\_00000.dat/Schroedinger\_1band/ev21*

### 3D simulation of the Triple Gate MOSFET

The following figures show the results of the self-consistent 3D Schrödinger-Poisson solution of this Triple Gate structure (Si channel length = 25 nm, source region length = 10 nm, drain region length = 10 nm, constant doping profile in source and drain region with a doping concentration of  $1 \times 10^{20} \text{ cm}^{-3}$  (fully ionized))

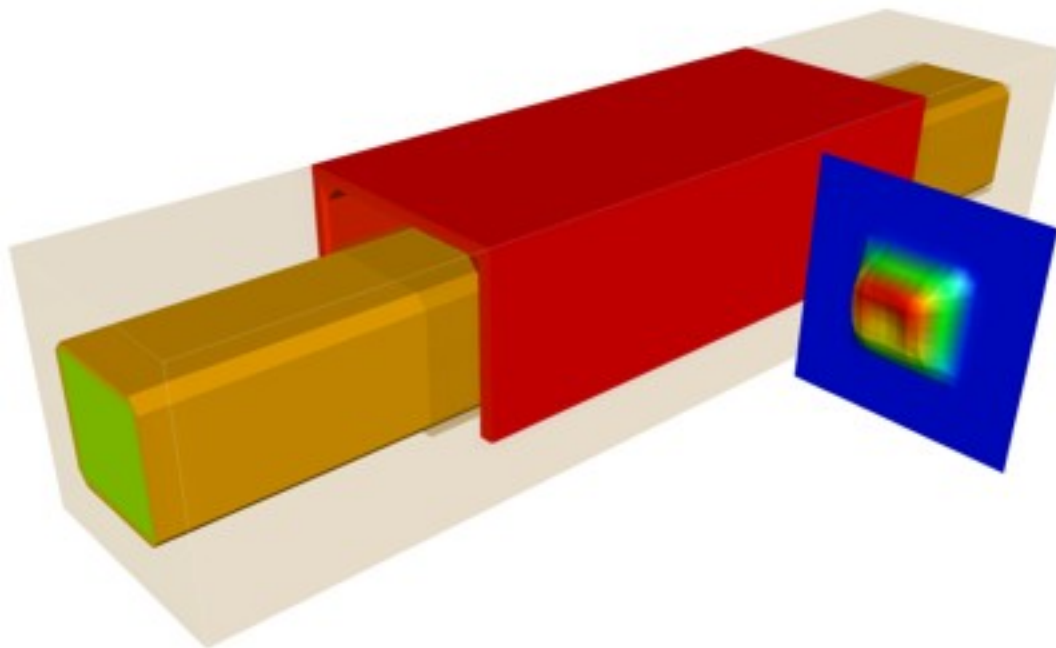


Figure 3.20.22.11: The whole 3D structure of this triple gate MOSFET and electron density through a 2D slice

The plots show the isosurfaces of the electron densities along 2D slices through the Triple Gate MOSFET. [Figure 3.20.22.12](#) and [Figure 3.20.22.13](#) also show 1D slices of the conduction band profiles and 1D slices of the electron densities in the middle of the device.

The classical densities would look similar to the classical densities of the 2D calculations shown above.

---

*Last update: nnnn/nn/nn*

### 3.20.23 Transmission through a nanowire (CBR)

- *Header*
- *System*



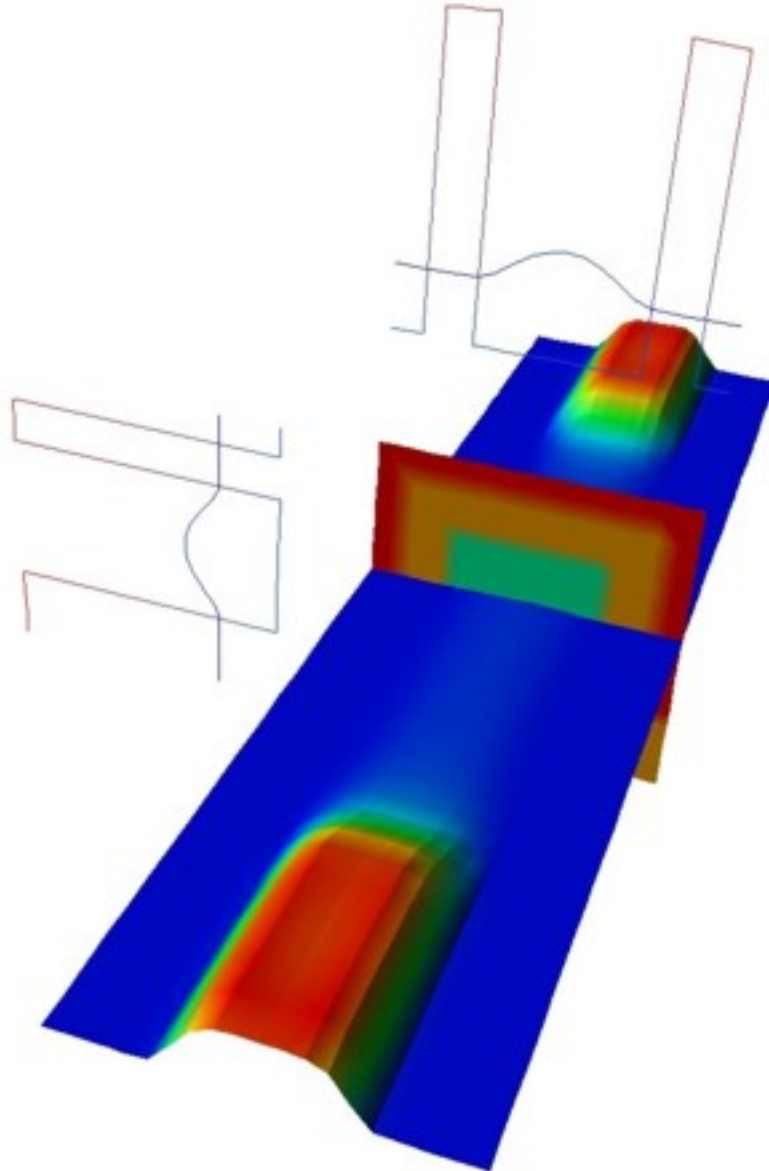


Figure 3.20.22.12: closed channel,  $V_{SD} = 0.0\text{V}$ ,  $V_{SG} = 0.0\text{V}$

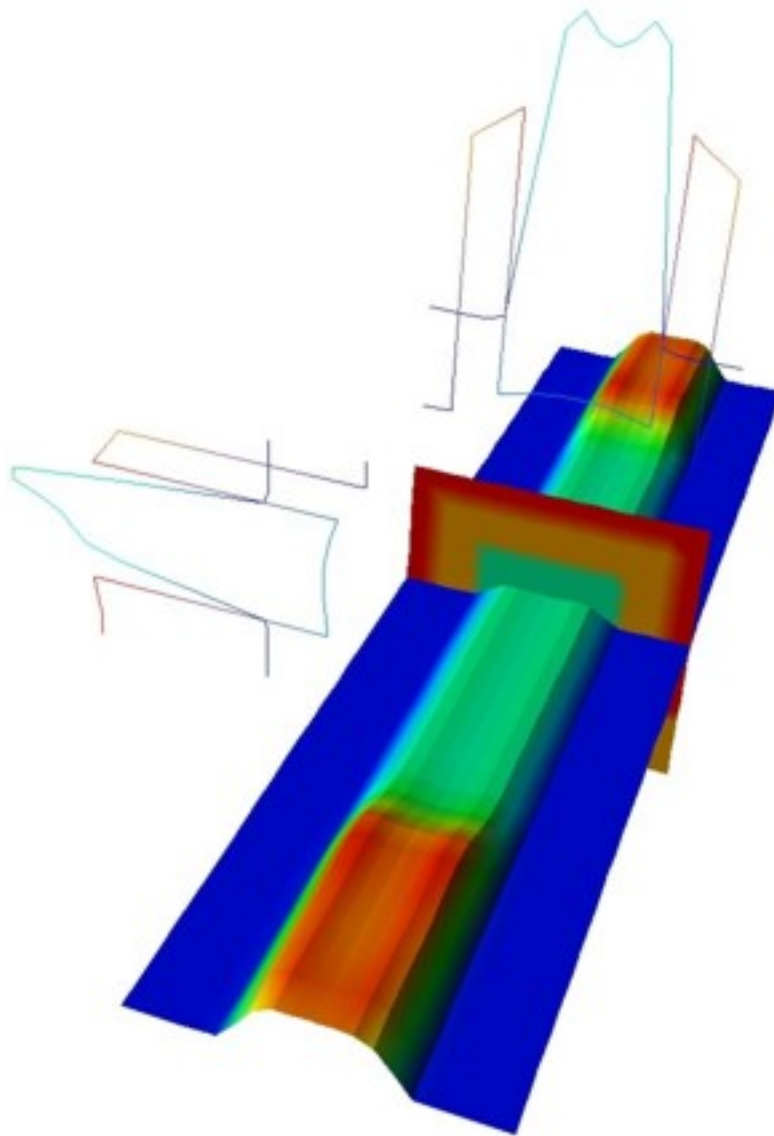


Figure 3.20.22.13: open channel,  $V_{SD} = 0.0\text{V}$ ,  $V_{SG} = 0.5\text{V}$

- *Input file*
- *CBR efficiency assessment*
- *Lead modes*

## Header

### Input Files:

- *transmission-nanowire\_GaAs\_3D\_nnp.in*

### Scope of the tutorial:

- transmission

We apply the Contact Block Reduction (CBR) method to a simple GaAs nanowire of cuboidal shape. The corresponding tutorial for *nextnano<sup>3</sup>* is [here](#).

## System

We consider a GaAs cuboidal tube of dimensions  $10\text{ nm} \times 10\text{ nm} \times 20\text{ nm}$ . Two leads of  $10\text{ nm} \times 10\text{ nm}$  each are attached to the edge of the device. The grid spacing is  $1\text{ nm}$  in all directions. The effective electron mass is assumed to be constant throughout the device and equal to  $0.067\ m_0$ .

## Input file

To simulate 3D (or 2D) system with CBR method in *nextnano++* correctly, The quantum regions have to be appropriately specified in the input file.

```
quantum{
  region{
    name = "lead_1"
    x    = [-6,6]
    y    = [-6,6]
    z    = [-0.1,0.1]
    boundary{ x=dirichlet y=dirichlet z=cbr }
    Gamma{ num_ev = $num_eigenstates_device }
  }
}
```

The perpendicular directions, i.e. x- and y-directions, of the system are elongated by one grid due to the treatment of edge points in *nextnano++*. Since the simulation is three dimensional, the lead region specified here has to be two dimensional. The number  $\pm 0.1$  is chosen to be smaller than the grid spacing, so that the region “lead\_1” becomes a 2D sheet (Note: this is slightly different in *nextnano<sup>3</sup>* input). CBR boundary condition has to be imposed in the propagation direction, i.e. z-direction, whereas Dirichlet boundary condition is set for perpendicular directions.

```
cbr{
  name = "device"
  lead{ name = "lead_1" }
  lead{ name = "lead_2" }
  delta_energy = $delta_energy
  abs_min_energy = $E_min
  abs_max_energy = $E_max
}
```

Here we specify the device region and leads attached to the device. The program calculates transmission through the region “device”, from “lead\_1” to “lead\_2”. The resolution, minimum and maximum of the energy axis can be also tuned here.

## CBR efficiency assessment

The biggest advantage of the CBR method is that it can correctly predict the spectrum without calculating all eigenmodes of the 3D device. That means that, for low energies, one can significantly reduce the simulation load for the calculation of transmission spectrum [Birner2009](#). To demonstrate it we perform three different simulations, sweeping the number of modes considered in the calculation. In the input file, the variable \$CBR\_case switches the number of eigenmodes.

```
$CBR_case      = 1                                # (ListOfValues:1,2,3)

$CBR_light     = iszero($CBR_case-1)
$CBR_medium    = iszero($CBR_case-2)
$CBR_heavy     = iszero($CBR_case-3)

#if $CBR_light $num_eigenstates_device = 200      # 5.6% of all device_
↪modes
#if $CBR_light $num_eigenstates_lead   = 30       # 17.8% of all lead modes

#if $CBR_medium $num_eigenstates_device = 400     # 11.3% of all device_
↪modes
#if $CBR_medium $num_eigenstates_lead   = 50      # 30.0% of all lead modes

#if $CBR_heavy $num_eigenstates_device = 600      # 16.9% of all device_
↪modes
#if $CBR_heavy $num_eigenstates_lead   = 80       # 47.3% of all lead modes
```

Figure 3.20.23.1 shows the calculated transmission coefficient as a function of energy. The result of *nextnano*<sup>3</sup> is shown for reference. Arrows indicate the cutoff energies, namely the eigenenergy of the highest device mode considered in each simulation. The transmission coefficient drops when the energy exceeds the cutoff value. In the low energy, however, it is sufficient to calculate only a part of all eigenfunctions of the device Hamiltonian. Lower cutoff energy means lower dimension of matrices and vectors in the simulation, e.g. Eq.(36) in [Birner2009](#), which reduces the calculation load. For example, a simulation performed at nextnano office took

- 42 sec for \$CBR\_case=1 (black)
- 3 min 14 sec for \$CBR\_case=2 (blue)
- 11min 17 sec for \$CBR\_case=3 (red)

## Lead modes

The step-like increase of the transmission coefficient is attributed to the discrete energy levels of the lead modes. Let us have a close look at the first few steps. We can see that  $T(E)$  increases by integers.

The lead mode probability distribution  $|\psi(x, y)|^2$  and corresponding eigenvalues are exported to the following files:

```
~\Quantum\wf_probabilities_lead_1_Gamma_0000.fld      ~\Quantum\
wf_energy_spectrum_lead_1_Gamma_0000.dat
```

To see the energy eigenvalues, it is convenient to switch to **Show Output File as Text** (marked yellow).

Once the energy reaches 76 meV, the first lead mode energy is reached and then this mode transmits perfectly, giving a transmission of 1.

As can be seen from \Quantum\wf\_probabilities\_lead\_1\_Gamma\_0000.fld, the second and third lead mode states are degenerate due to the symmetry of the lead cross-section. Thus they have the same energy 190 meV. Consequently, the spectrum increases by 2 at the energy of 190 meV. In this fashion, the step-like behavior of the transmission coefficient is explained by lead eigenmodes.

---

*Last update: nnnn/nn/nn*

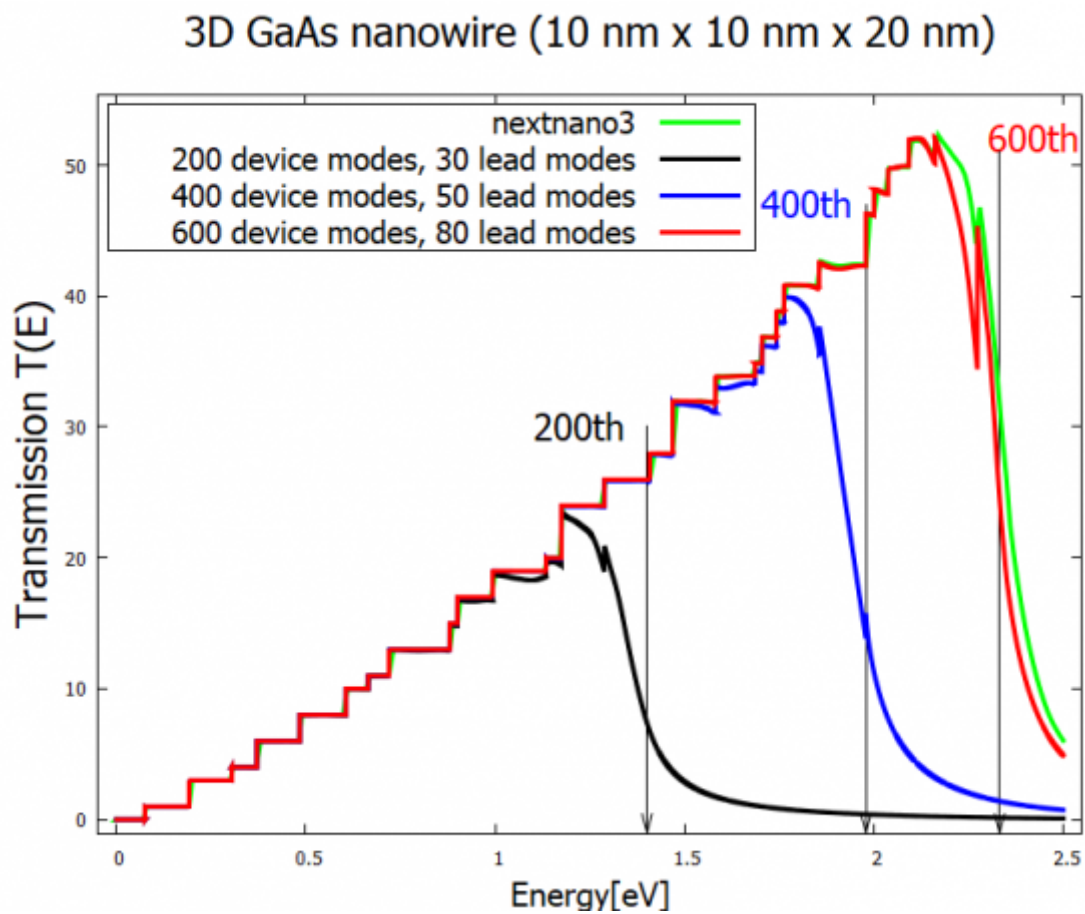


Figure 3.20.23.1: Transmission coefficient of a GaAs 3D nanowire simulated with three different CBR parameters. The *nextnano<sup>3</sup>* result is shown for reference. Arrows indicate the cutoff energies, namely the eigenenergy of the highest device eigenmode considered in each simulation.

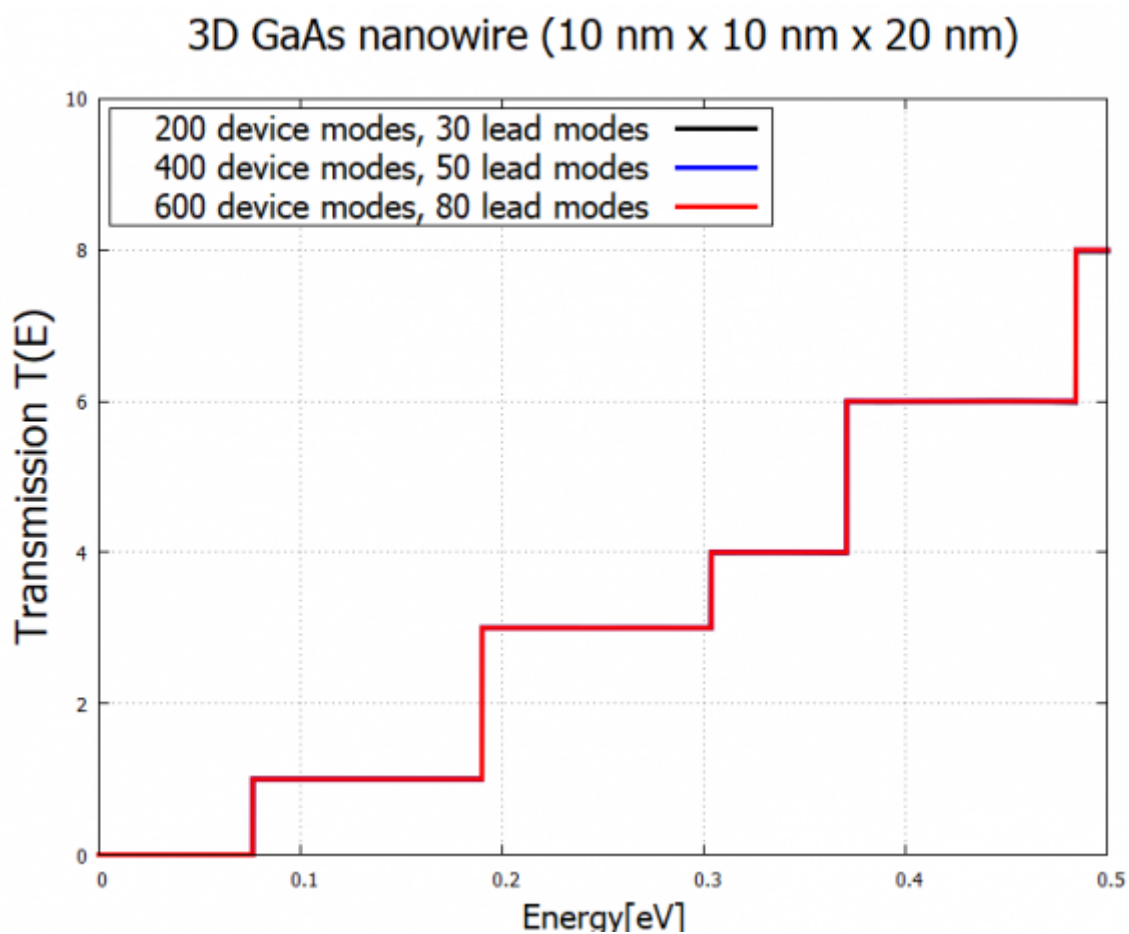
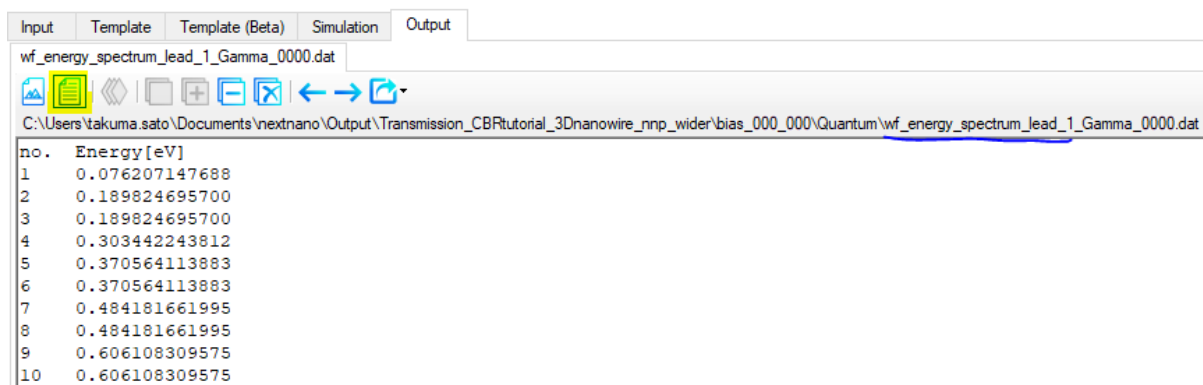


Figure 3.20.23.2: Zoom into the first few steps of  $T(E)$ . The transmission increases by integer at the eigenenergies of the lead.



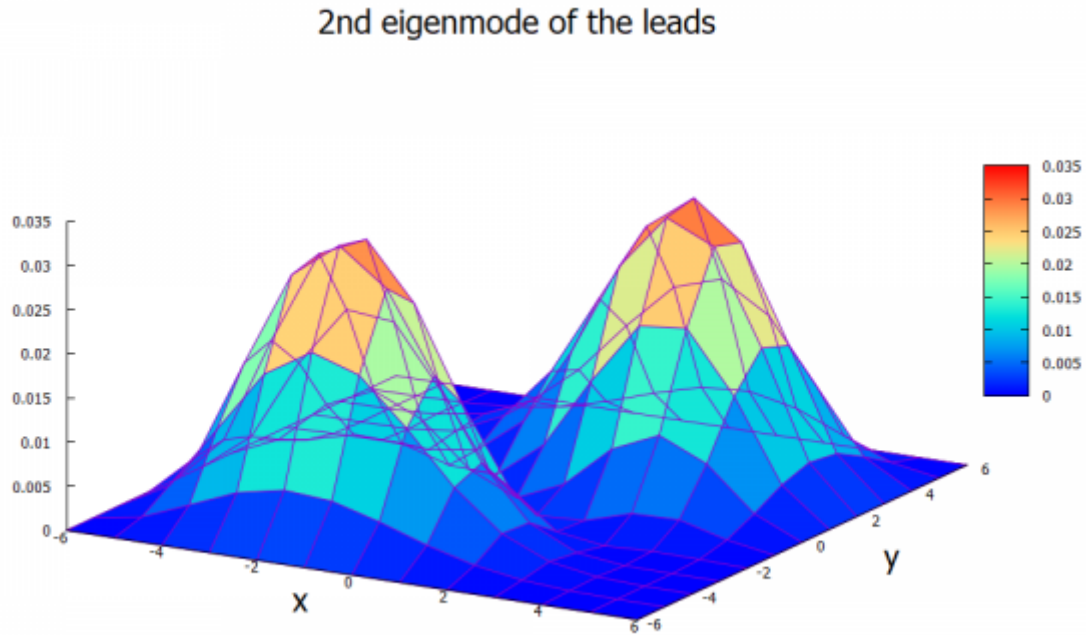


Figure 3.20.23.3: The probability distribution  $|\psi(x, y)|^2$  of the 2nd lead mode.

### 3.20.24 — DEV — Artificial quantum dot crystal - Superlattice dispersion (mini-bands)

#### ⚠ Attention

This tutorial is under construction

#### Input files:

- *QDSL\_Ge-Si\_Lazarenkova\_JAP\_2001\_3D\_cubic\_nnp.in*
- *QDSL\_Ge-Si\_Lazarenkova\_JAP\_2001\_3D\_tetragonal\_nnp.in*

#### Scope:

- In this tutorial, the superlattice energy dispersion for artificial crystals consisting of quantum dots (QDs) are calculated. The QDs are made of Ge embedded in Si. The simulations are performed for cubic and tetragonal QDs.
- This tutorial aims to reproduce figure 2 and 6 in [Lazarenkova2001].

#### Output files:

- *bias\_00000\Quantum\Dispersion\dispersion\_quantum\_region\_HH\_along\_100.dat*
- *bias\_00000\Quantum\Dispersion\dispersion\_quantum\_region\_HH\_along\_110.dat*
- *bias\_00000\Quantum\Dispersion\dispersion\_quantum\_region\_HH\_along\_111.dat*

#### Cubic Quantum Dots

The QDs have a cubic shape with  $L_x = L_y = L_z = 6.5$  nm and are separated by a distance of  $H_x = H_y = H_z = 1.5$  nm. We model only one QD and assume periodic boundary conditions along the x, y and z direction giving a superlattice period of  $d_x = d_y = d_z = 8$  nm ( $d_i = L_i + H_i$ ).

```

global{
  ...
  periodic{
    x = yes
    y = yes
    z = yes
  }
}

```

The single-band Schrödinger equation is solved for the valence band only (heavy hole). The valence band offset is assumed to be  $VBO = 0.45$  eV, i.e. assuming the valence band edge of the QD is at +0.45 eV, the valence band edge of the barrier is at 0 eV. The energy dispersion relation is calculated along the [100], [110] and [111] direction.

```

quantum{
  ...
  HH{
    num_ev = $num_states
    dispersion{
      output_dispersions{}
      path{
        name = "along_100"
        point{ k = [0.0, 0.0, 0.0] }
        point{ k = [1.0, 0.0, 0.0] }
        num_points = $num_k_points
      }
      path{
        name = "along_110"
        point{ k = [0.0, 0.0, 0.0] }
        point{ k = [1.0, 1.0, 0.0] }
        num_points = $num_k_points
      }
      path{
        name = "along_111"
        point{ k = [0.0, 0.0, 0.0] }
        point{ k = [1.0, 1.0, 1.0] }
        num_points = $num_k_points
      }
    }
  }
}

```

## Results

Figure 3.20.24.1 shows the calculated dispersion relation along the [100] direction. The figure agrees very well with Fig. 2(a) of the paper by Lazarenkova et.al. [Lazarenkova2001] in the energy region where the confinement inside the QD is strong. For the higher lying states inside the QD and above the QD, our results differ because we use the correct 3D QD confinement potential whereas Lazarenkova et.al. [Lazarenkova2001] approximated the potential landscape with an analytical ansatz that allows for the separation of the x, y and z variables. (This ansatz is only justified for states confined deep inside the QD.) The right part of the figure shows schematically the valence band edge of the QD with the energy levels of the single, uncoupled QD.

In all figures, the eigenstates are labeled with the quantum numbers  $n_x, n_y, n_z$ , e.g. 111. The superscript refers to their degeneracy. At  $K_{SL} = 0$ , the degeneracy is higher than at nonzero KSL vectors where the symmetry in the Brillouin zone is lower. (The superlattice vector  $K_{SL} = 0$  is often denoted as  $q$ .)

The following figures (Figure 3.20.24.2 and Figure 3.20.24.3) show the calculated dispersion relations along the [110] and [111] directions, respectively. The agreement to Fig. 2(b) and 2(c) of [Lazarenkova2001] is again very good for the states that lie deep inside the QD (see also comments above). Note that the eigenstates along the [111]



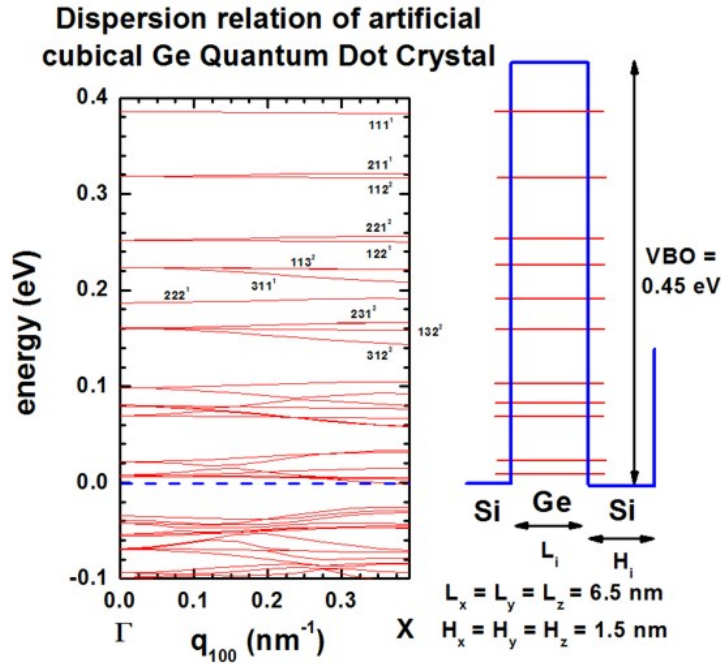


Figure 3.20.24.1: (left) Superlattice dispersion along [100] for heavy-hole states in an artificial cubic Ge quantum dot crystal and (right) valence-band profile through the center of the QD and eigenstates of the lowest heavy-hole states.

direction show a higher degree of degeneracy throughout the superlattice Brillouin zone as compared to [100] and [110].

Both, the QD itself and the QD superlattice have the same cubic symmetry in this example. Thus the degeneracy of the 123 (incl. permutations) energy band is sixfold throughout the Brillouin zone along the [111] directions (as shown in Figure 3.20.24.3).

### Tetragonal Quantum Dot

The QD has a tetragonal shape with  $L_x = L_y = 5$  nm and  $L_z = 2.5$  nm and are separated by a distance of  $H_x = H_y = 2.5$  nm and  $H_z = 1.25$  nm. This gives a superlattice period ( $d_i = L_i + H_i$ ) of  $d_x = d_y = 7.5$  nm and  $d_z = 3.75$  nm. The grid spacing was chosen to 0.25 nm in  $x$ ,  $y$  and  $z$  direction, i.e. 30 grid points in  $x$  and  $y$  direction, 15 grid points in  $z$  direction. Therefore, the size of Schrödinger matrix to be solved is  $30 \cdot 30 \cdot 15 = 13500$ . All other assumptions are the same as for the cubic QD example above.

### Results

Figures show the dispersion along the [100], [110] and [111] directions, respectively. All results are in very good agreement to Fig. 6 of [Lazarenkova2001].

The version of this tutorial for nextnano<sup>3</sup> can be found [here](#).

Last update: 17/07/2024

## 3.20.25 — DEV — Energy levels of an “artificial atom” - Spherical and ellipsoidal CdSe Quantum Dot

### ⚠ Attention

This tutorial is under construction

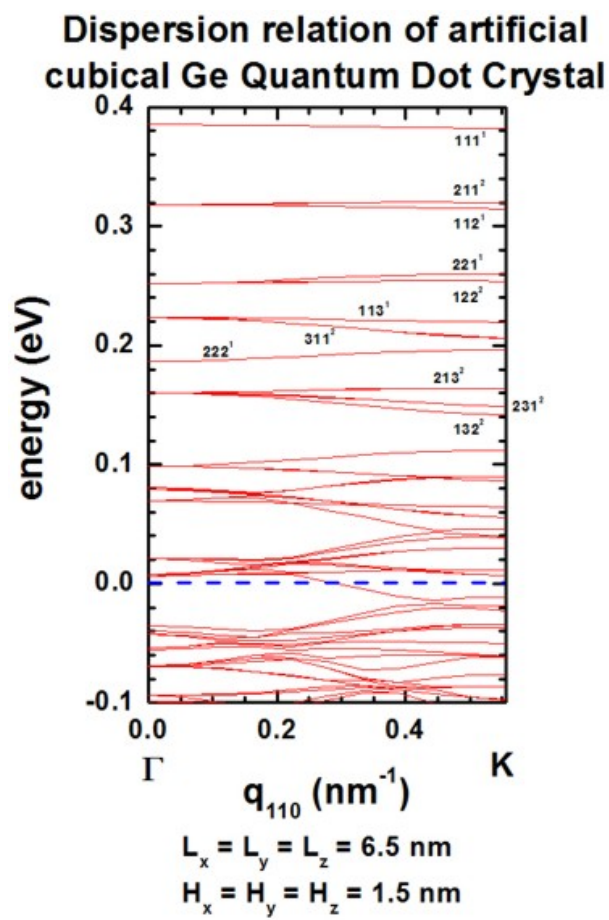


Figure 3.20.24.2: Superlattice dispersion along [110] for heavy-hole states in an artificial cubic Ge quantum dot crystal.

### Dispersion relation of artificial cubical Ge Quantum Dot Crystal

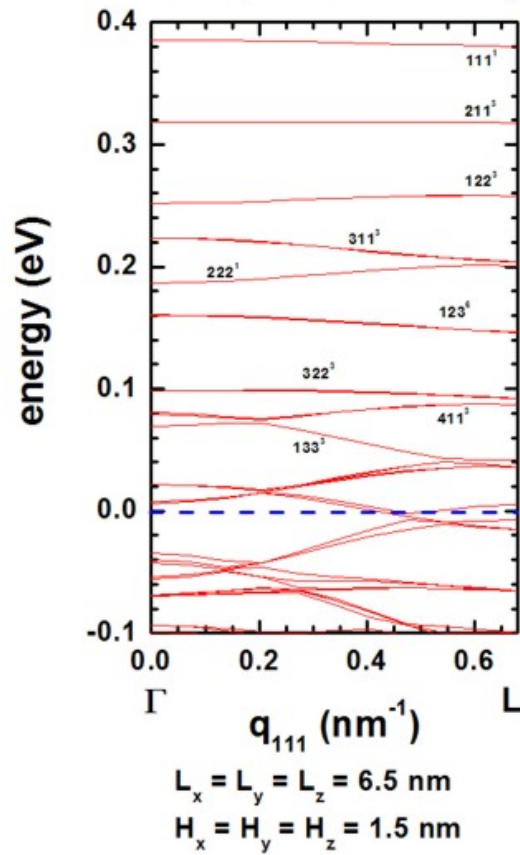


Figure 3.20.24.3: Superlattice dispersion along [111] for heavy-hole states in an artificial cubic Ge quantum dot crystal.

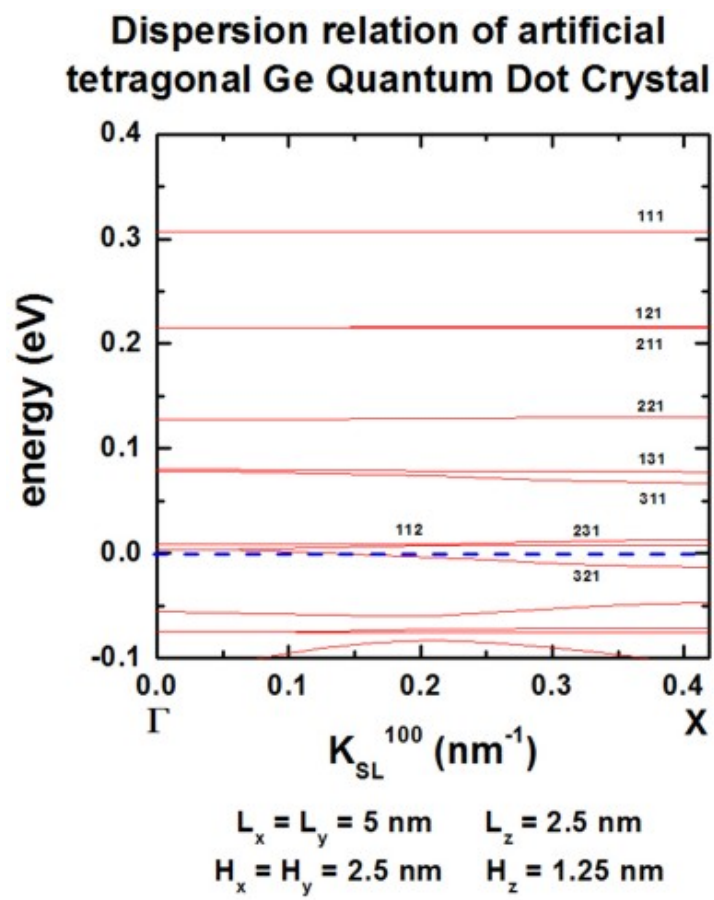


Figure 3.20.24.4: Superlattice dispersion along [100] for heavy-hole states in an artificial tetragonal Ge quantum dot crystal.

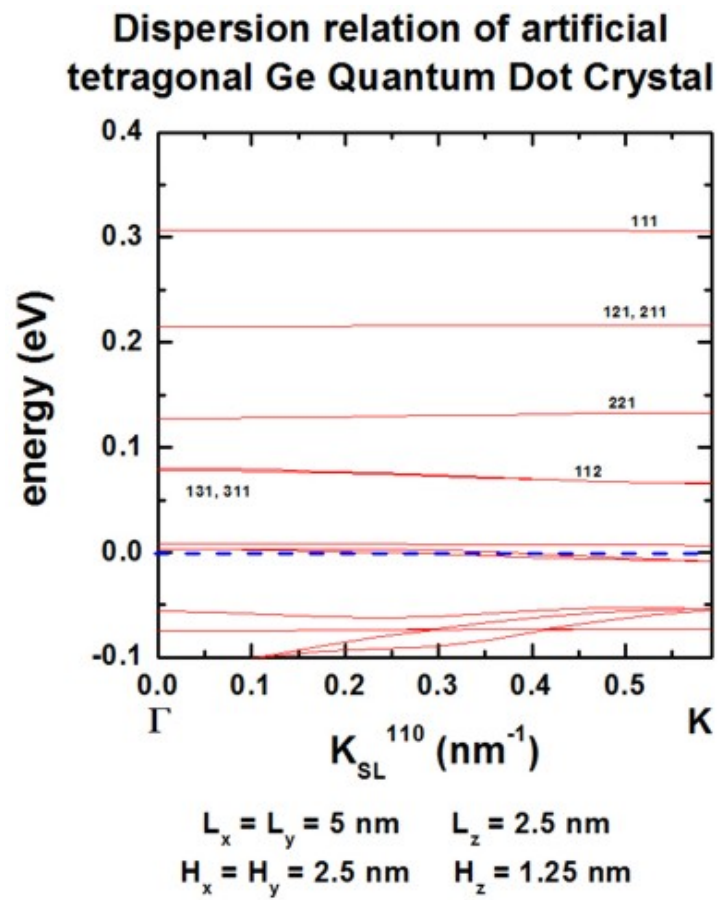


Figure 3.20.24.5: Superlattice dispersion along [110] for heavy-hole states in an artificial tetragonal Ge quantum dot crystal.

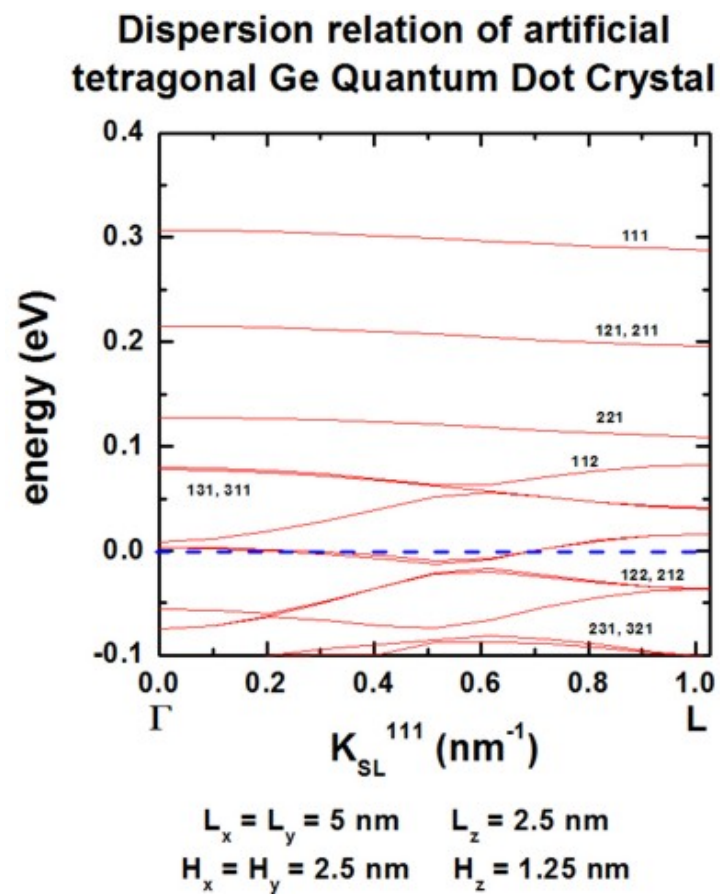


Figure 3.20.24.6: Superlattice dispersion along  $[111]$  for heavy-hole states in an artificial tetragonal Ge quantum dot crystal.

**Input files:**

- *QDArtificialAtom\_CdSe\_3D\_spherical\_nnp.in*
- *QDArtificialAtom\_CdSe\_3D\_ellipsoidal\_nnp.in*
- *ParabolicQW\_GaAs\_2D\_nnp.in*

**Scope:**

- In this tutorial we calculate the eigenenergies of a spherical and ellipsoidal CdSe quantum dot (“artificial atom”). The tutorial is based on [\[Ferreira2006\]](#).

**Output files:**

- *bias00000\Quantum\energy\_spectrum\_quantum\_region\_Gamma\_00000.dat*
- *bias00000\Quantum\overlap\_integrals\_quantum\_region\_HH\_Gamma.txt*

**Energy levels of an “artificial atom” - Spherical CdSe Quantum Dot**

Here, we want to calculate the energy levels and the wave functions of a spherical CdSe quantum dot of radius  $r = 5$  nm shown in [Figure 3.20.25.1](#).

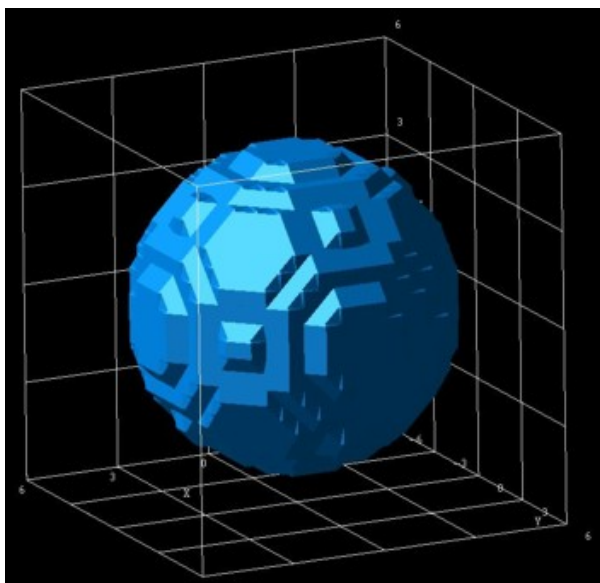


Figure 3.20.25.1: Spherical quantum dot.

We assume that the barriers at the QD boundaries are infinite. The potential inside the QD is assumed to be 0 eV. We use a grid resolution of 0.5 nm. We solve the single-band Schrödinger equation within the effective-mass approximation. The electron effective mass of CdSe is assumed to be  $m_e = 0.112m_0$ .

A spherically symmetric potential leads to an energy spectrum where some eigenvalues are degenerate. We want to study the “shell structure” (degeneracy scheme) of a CdSe quantum dot of radius 5 nm. [Figure 3.20.25.2](#) shows the calculated energy spectrum for the lowest 20 electron eigenvalues. One can clearly identify the shell structure 1s, 2p, 3d, 2s, 4f and 3p which is similar to the shell structure of the periodic table. This is the reason why quantum dots are often called “artificial atoms”. Note that each eigenstate is two-fold degenerate due to spin. Thus, the s states are two-fold degenerate, the p states are six-fold degenerate, the d states are ten-fold degenerate and the degeneracy of the f states is 14.

We have also solved the single-band Schrödinger equation for the holes assuming an isotropic effective mass for simplicity. Obviously, this is a crude approximation. From the electron and hole wave functions, we calculate their spatial overlap matrix elements (overlap integrals). In this simple model, due to symmetry arguments, only the following transitions are allowed: 1s - 1s, 2p - 2p, 3d - 3d, 2s - 2s, 4f - 4f, ...

[Figure 3.20.25.3](#) shows the calculated overlap integrals as a function of energy. (Note: The figure has to be updated: Now we output the square of this matrix element.)

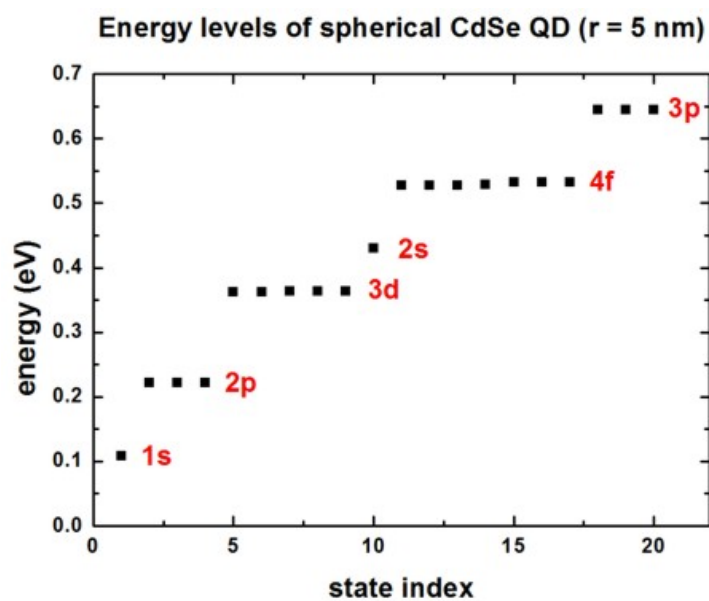


Figure 3.20.25.2: Eigenenergies of the lowest 20 states in the QD.

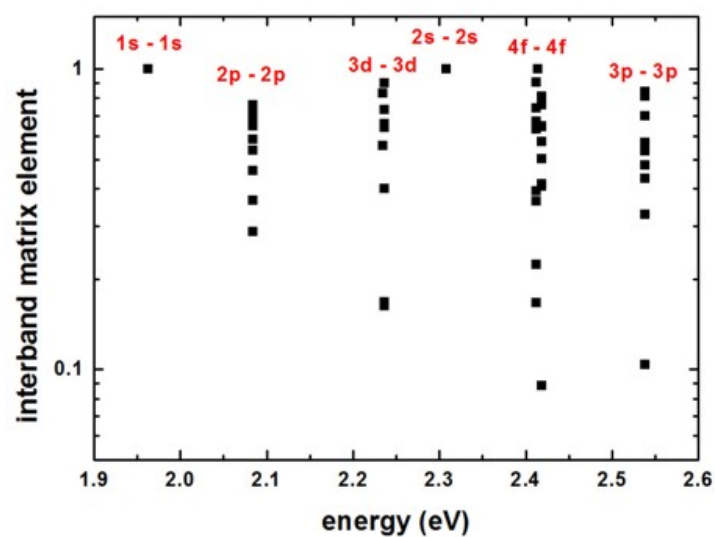


Figure 3.20.25.3: Overlap integrals.



Both figures are in reasonable agreement with Fig. 1 and Fig. 2 (inset) in [Ferreira2006].

### Energy levels of an “artificial atom” - Ellipsoidal, cigar-shaped CdSe quantum dot

For an ellipsoidal, cigar-shaped CdSe quantum dot ( $r_x = 5$  nm,  $r_y = 5$  nm,  $r_z = 10$  nm), we calculate the lowest 30 eigenvalues.

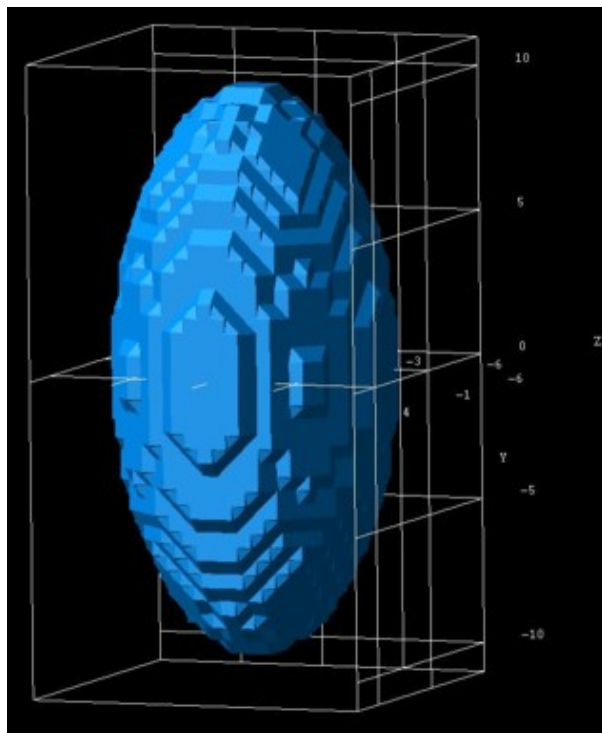


Figure 3.20.25.4: Ellipsoidal quantum dot.

The energy spectrum (degeneracy spectrum) looks very different from the spherical QD spectrum (c.f. Figure 3.20.25.5)

The overlap integrals are shown in Figure 3.20.25.6 (Note: The figure has to be updated: Now we output the square of this matrix element.)

### Energy levels of an “artificial atom” - 2D harmonic potential

The following figure shows the energy spectrum of a “two-dimensional disc” which we approximate as a cylindrically symmetric parabolic (harmonic) potential. We solve the 2D Schrödinger equation for this system. The harmonic potential is assumed to be  $\hbar\omega = 3$  meV. Each shell is thus separated by 3 meV. From the energy spectrum of this two-dimensional shell structure, one can derive “magic numbers”. (They include spin degeneracy.)

Version of this tutorial for nextnano<sup>3</sup> can be found [here](#).

Last update: 27/05/2025

## 3.20.26 Quantum Dot Molecule

In this tutorial, we study two coupled quantum dots (QDs), i.e. two “artificial atoms” that form an “artificial molecule”. The two QDs are asymmetric and differ with respect to their height (4 nm and 6 nm).

With no electric field, the groundstates of both electron and hole are localized at the larger QD. By applying the electric field and increasing its strength, the hole groundstate becomes bonding state and then localizes at the smaller QD. At the same time the electron groundstate is still localized at the larger QD because of the weaker

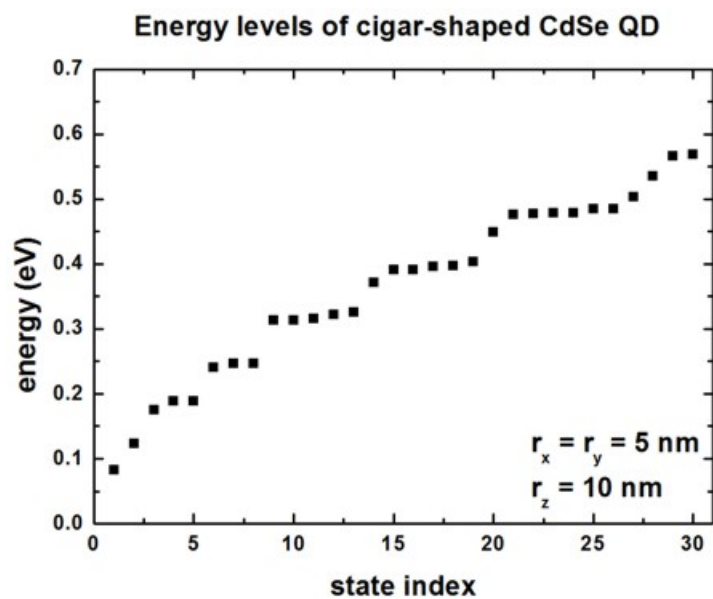


Figure 3.20.25.5: Eigenenergies of the lowest 30 states in the QD.

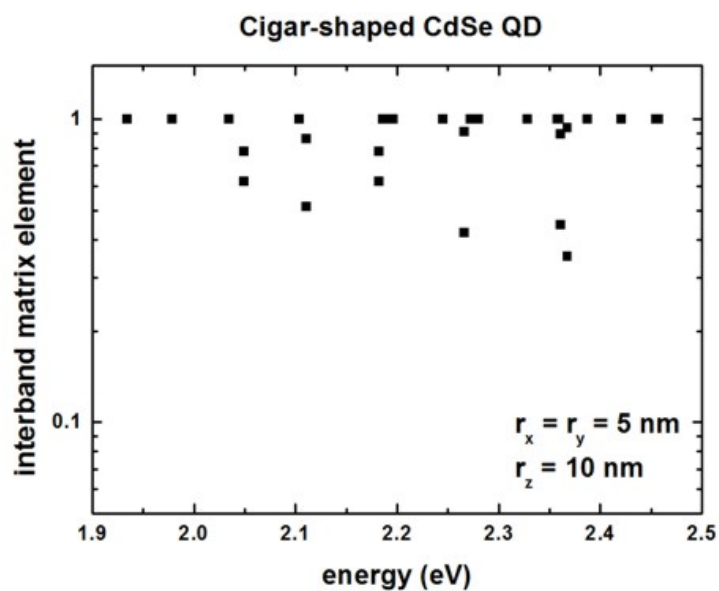


Figure 3.20.25.6: Overlap integrals.

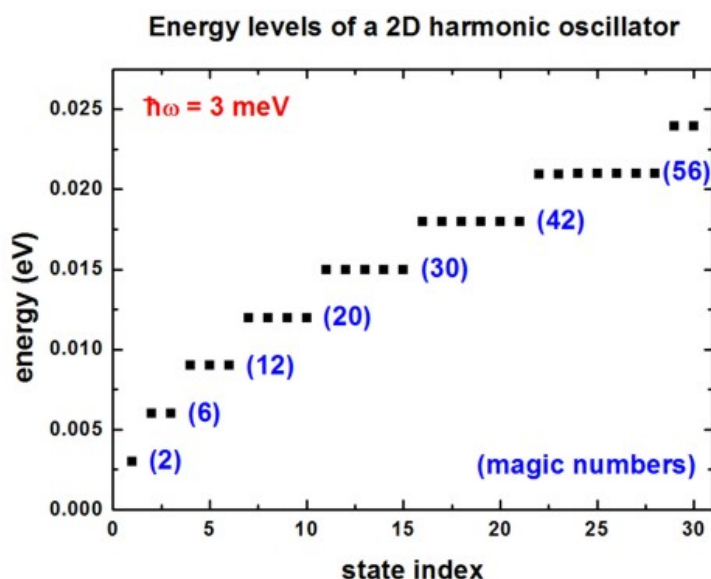


Figure 3.20.25.7: Eigenenergies of the lowest 30 states in a harmonic potential.

coupling between the two QDs due to the heigher barrier height. We will see this leads to the change from an direct exciton to indirect exciton.

The relevant input files are as followings:

- `3DQD_molecule_cuboid_asymmetric_nn3.in` / `*_nnp.in`

Some of the material parameters that are used in this tutorial are based on the paper of

M. Grundmann, D. Bimberg

Formation of quantum dots in twofold cleaved edge overgrowth

Phys. Rev. B 55 (7), 4054 (1997).

i.e. they are the same as in the [3D CEO QD](#) tutorial (apart from the effective masses).

## Simulation

This simulation has the following features:

- We keep things simple by using cuboidal shaped GaAs QDs surrounded by  $\text{Al}_{0.35}\text{Ga}_{0.65}\text{As}$  barriers, i.e. we neglect strain and piezoelectric effects which is reasonable as the two materials GaAs and  $\text{Al}_{0.35}\text{Ga}_{0.65}\text{As}$  have pretty similar lattice constants.
- We also neglect the wetting layers and excitonic effects.
- In order to keep the CPU time to a minimum, we do not use the k.p approximation, i.e. we use for both electrons and the heavy hole a single-band effective mass approximation for the Schrödinger equation (parabolic and isotropic effective mass tensor). Nevertheless, this is sufficient to show some basic quantum physical effects of this QD molecule.
- We use different electron and hole masses in the barrier and well material, respectively.
- The left QD has the dimensions 10 nm x 10 nm x **4 nm** (smaller dot). The right QD has the dimensions 10 nm x 10 nm x **6 nm** (larger dot).
- The two QDs are separated by a 2 nm  $\text{Al}_{0.35}\text{Ga}_{0.65}\text{As}$  barrier.
- The grid resolution is 0.5 nm (rectangular tensor grid). This leads to a 3D Schrödinger matrix of dimension 50,225.
- We apply Dirichlet boundary conditions to the Schrödinger equation, i.e. the wave functions are allowed to penetrate the following distances into the barrier material (on each side): - along the x and y directions: 4 nm - along the z direction: 4.5 nm

- We vary the electric field along the growth direction (z axis) in steps of 2.5 kV/cm, i.e. from 0 kV/cm to 40 kV/cm. Note that the direction of the electric field is opposite to the z-direction.

## Results

### Electron and heavy hole ground states

The following figure show the square of the first electron (**el1**, top) and heavy hole (**h1**, bottom) wave functions (isosurfaces at  $\psi^2 = 0.0015$ ) for different applied electric fields (0 kV/cm, 17.5 kV/cm, 40 kV/cm).

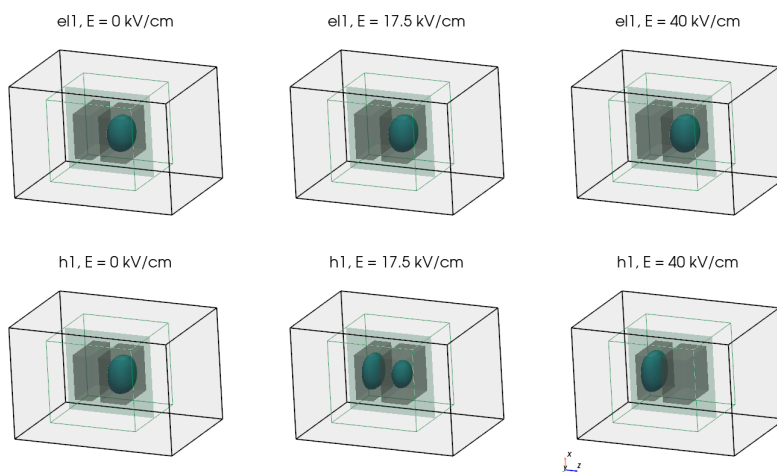


Figure 3.20.26.1: Probabilities of first electron and first heavy hole states at different electric fields. The green wireframe shows the quantum region, the grey shaded areas are the quantum dots. The shown isosurfaces are at  $\psi^2 = 0.0015$ . The shaded green slice is at  $y = 0$ , i.e. through the center of the QDs.

The following figure shows the electron (**el1**, top) and heavy hole (**h1**, bottom) probabilities from the 2D slices defined in Figure 3.20.26.1 at different electric fields.

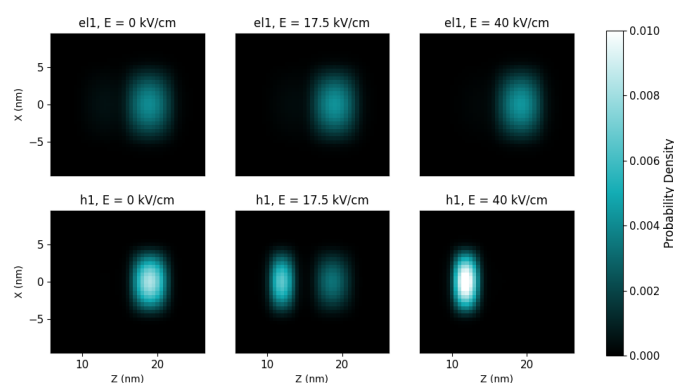


Figure 3.20.26.2: 2D slices at  $y = 0$  of probabilities of first electron and first heavy hole states at different electric fields.

At zero applied electric field, both electron and heavy hole are located in the larger dot and form a **direct (bright) exciton**.

At an electric field of 17.5 kV/cm, the electron is still located in the larger dot on the right side, whereas the heavy hole located in both wells (strong coupling). The heavy hole states at the field of 17.5 kV/cm are discussed further below. The exciton that is formed is something in between a direct and an indirect exciton.

At an electric field of 40 kV/cm, the electron is still located in the larger dot on the right side, whereas the heavy hole ground state is now located in the left QD. An indirect (dark) exciton is formed. The exciton is called dark because the electron-hole overlap is much smaller and thus its oscillator strength (probability of optical transition) is much weaker (see Figure 3.20.26.6 below on spatial electron-hole overlap integrals).

### Electron and heavy hole energies

Figure 3.20.26.3 shows the electron (left) energies of the ground state (**e1**) and the first excited electron state (**e2**) of the QD molecule, as well as the heavy hole (right) energies of the ground state (**h1**) and the excited hole states (**h2-h5**) of the QD molecule. The electron ground state (**e1**) is always located in the larger QD (right side) whereas the first excited electron state (**e2**) is always located in the smaller QD (left side). The third and the fourth eigenstate (**e3**, **e4**) are degenerate (not shown) because our QD molecule has a symmetry with respect to the x and y coordinates. They are always located in the right QD. In contrast to the electrons, the hole coupling between the two QDs is much stronger due to the smaller barrier height. At 17.5 kV/cm anticrossing between the states occur due to the formation of bonding and antibonding states.

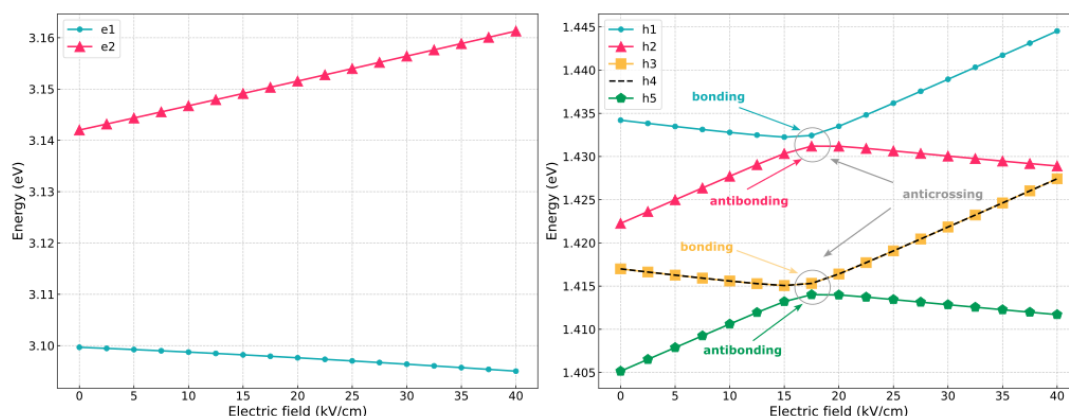


Figure 3.20.26.3: Electron (left) and heavy hole (right) energies

### Bonding and antibonding heavy hole state at anticrossing point

The following figure shows the envelopes of the first six hole wave functions (isosurfaces  $\psi = \pm 0.015$ ) at an electric field of 17.5 kV/cm.

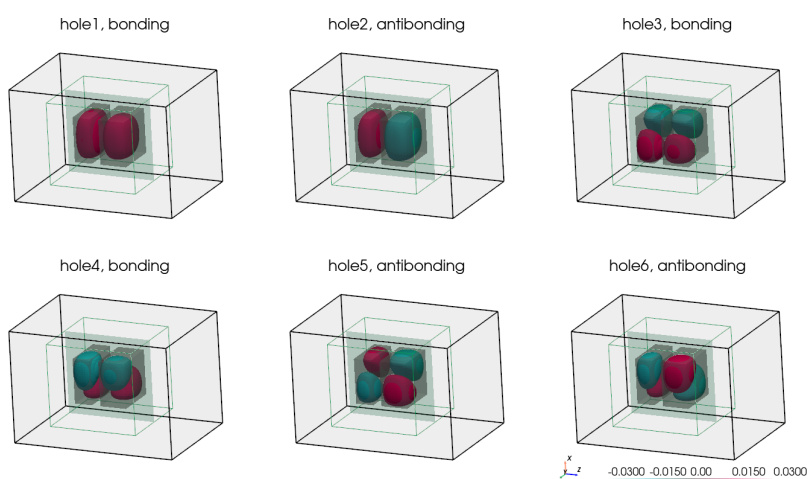


Figure 3.20.26.4: Envelope of the first 6 hole states at electric field of 17.5 kV/cm (some are bonding and some are antibonding states). The green wireframe shows the quantum region, the grey shaded areas are the quantum dots. The shown isosurfaces are at  $\psi = \pm 0.015$ . The shaded green slice is at  $y = 0$ , i.e. through the center of the QDs.

The following figure shows hole envelopes from the 2D slices defined in Figure 3.20.26.4.

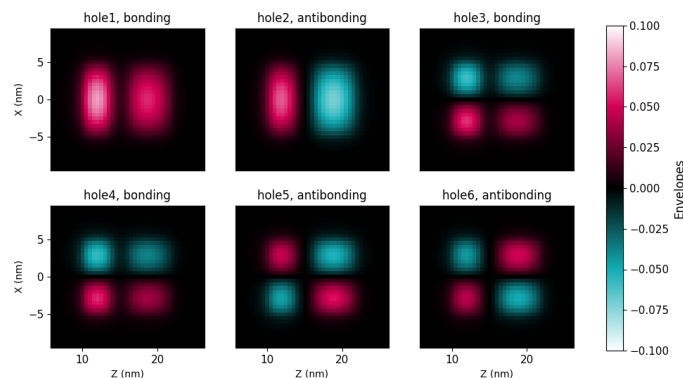


Figure 3.20.26.5: 2D slices at  $y = 0$  of envelopes of the first 6 hole states at electric field of 17.5 kV/cm (some are bonding and some are antibonding states).

### Electron-hole transition energies and overlap

To understand the strength of the optical transitions we have to evaluate the matrix elements of the envelope functions, i.e. the spatial overlap integral over the electron and hole wave functions.

$$\int \psi_{el,i}^*(x) \psi_{hl,j} dx$$

The following figure shows the five lowest electron-hole transition energies of the QD molecule as a function of electric field (left) and the overlap between the first electron (**el1**) and the first and second heavy hole (**h1** and **h2**) wave functions (right).

For fields smaller than 17.5 kV/cm a direct (bright) exciton is the ground state (both electron and hole wave function are located in the larger QD (right side), whereas for fields larger than 17.5 kV/cm an indirect (dark) exciton is the ground state where the electron is located in the larger QD (right side) and the hole is located in the smaller QD (left side). Therefore, the nature of the QD molecule ground state changes from **direct** to **indirect**.

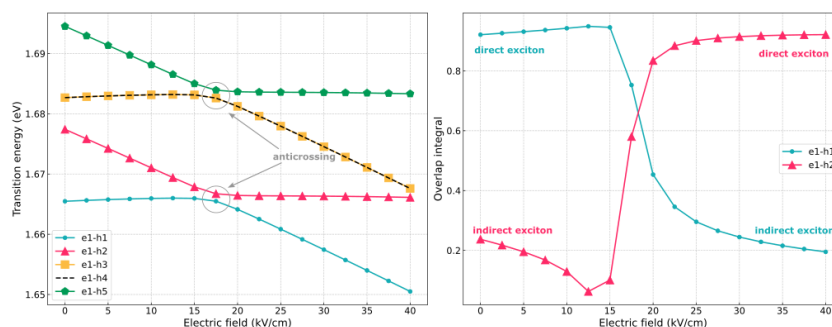


Figure 3.20.26.6: Electron-hole transition energies (left) and overlap (right) as a function of electric field.

Last update: 2025/06/25

### 3.20.27 Hole energy levels of an “artificial atom” - Spherical Si Quantum Dot (6-band k.p)

Input files:

- `3DsphericSiQD_d5nm_6bandkp_nnp.in`

**Scope:**

In this tutorial, we calculate the energy spectrum of a spherical Si quantum dot of radius 2.5 nm.

**Output Files:**

- *bias\_00000\Quantum\energy\_spectrum\_qr\_6band\_kp6\_00000.dat*

**Introduction**

We assume that the barriers at the QD boundaries are infinite. The potential inside the QD is assumed to be 0 eV. We use a grid resolution of 0.25 nm. We solve the 6-band k.p Schrödinger equation for the hole eigenstates.

The following 6-band k.p parameters are used:

```
kp_6_bands{
  L = -6.8           # [Burdov] V.A. Burdov, JETP 94, 411 (2002)
  M = -4.43          # [Burdov]
  N = -8.61          # [Burdov]
}
```

These L, M, N parameters correspond to the following Luttinger parameters:

- $\gamma_1 = 4.22$
- $\gamma_2 = 0.395$
- $\gamma_3' = 1.435$

**Results**

Figure 3.20.27.1 shows the isosurfaces of the probability density of the first six hole eigenstates of a spherical Si quantum dot.

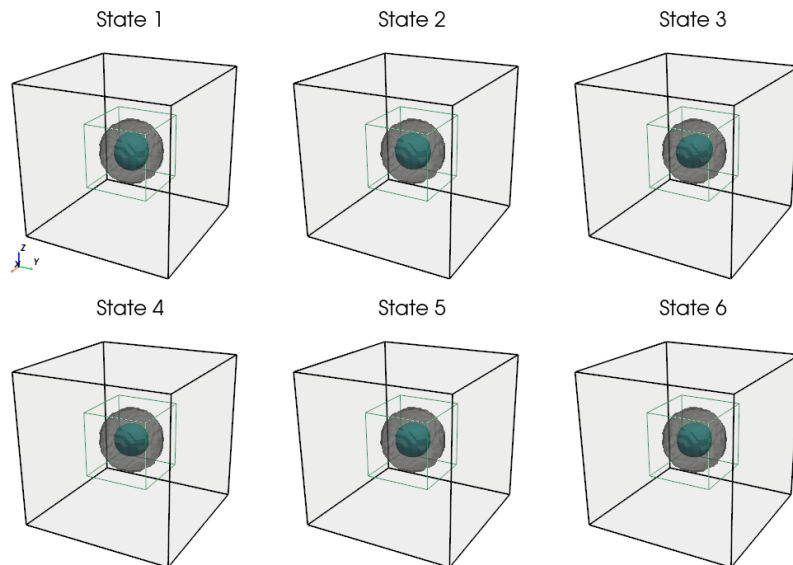


Figure 3.20.27.1: Isosurfaces of the first six hole eigenstates in a spherical Si QD calculated with a 6-band k.p Hamiltonian.

Figure 3.20.27.2 shows the hole eigenenergy spectrum of the Si QD (diameter = 5 nm) calculated with a 6-band k.p Hamiltonian.

For comparison, we also display the energy spectrum where we assumed zero spin-orbit splitting energy. In this case there is a six-fold symmetry. Spin-orbit splitting reduces this degeneracy to 4 and 2. In general, each state is two-fold degenerate due to spin.

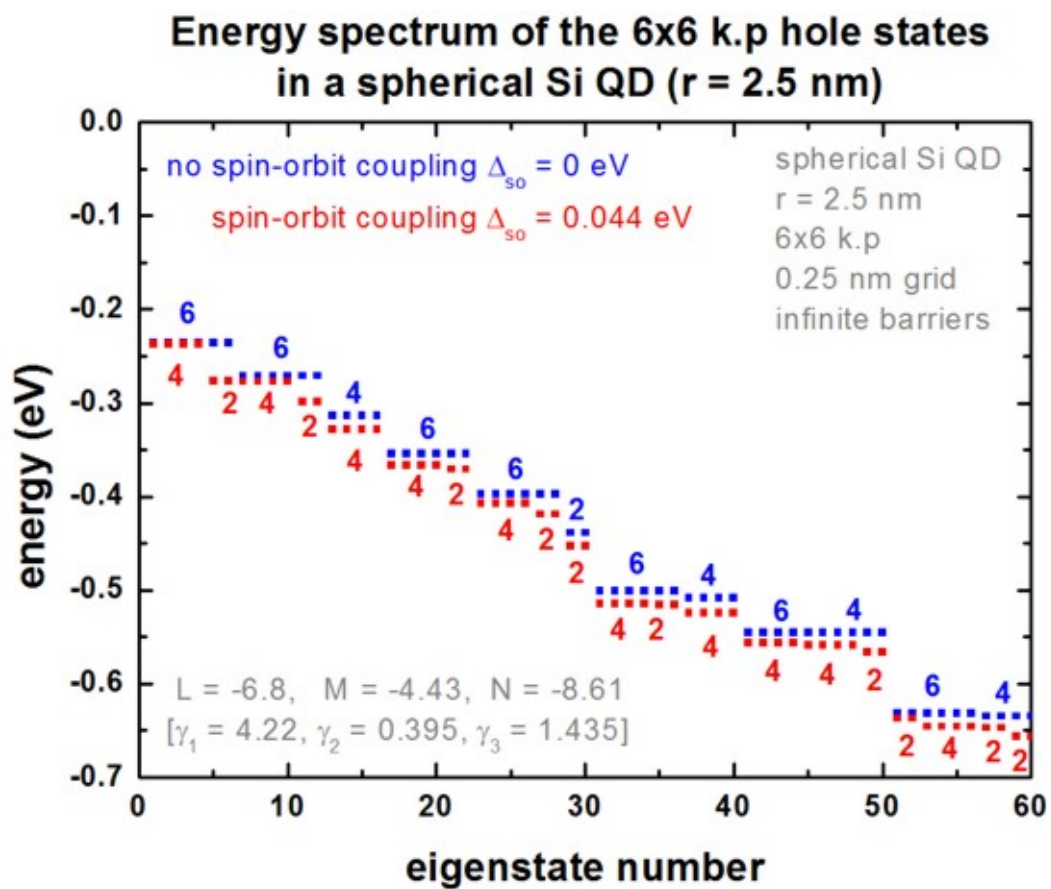


Figure 3.20.27.2: Energy spectrum of the 6-band k.p hole states in a spherical Si QD.



**Note**

The *nextnano++* tool only allows a cuboidal shaped quantum region, thus we can't employ a spherical quantum region that would reduce the dimension of the 6-band k.p Hamiltonian matrix and thus the overall execution time.

Following the paper of [Burdov2002], one can calculate the ground state energy for this particular system from the L and M parameters:

$$E_1 = -\frac{\hbar^2 \pi^2}{2m_h R^2} = -0.314 \text{ eV}$$

using  $m_h = 0.192 m_0$  as [Burdov2002], where he uses incorrect k.p parameters: In his definition L must be -5.8 and M = -3.43.

$$E_1 = -\frac{\hbar^2 \pi^2}{2m_h R^2} = -0.254 \text{ eV}$$

using  $m_h = 0.237 m_0$  as [BelyakovBurdov2008]. The latter is in much better agreement to our calculations.  $m_h$  is given by:

$$m_h = \frac{3m_0}{L + 2M} = \frac{3m_0}{-6.8 + 2 \cdot (-4.43)} = -0.192 m_0$$

in [Burdov2002] and

$$m_h = \frac{3m_0}{(L + 1) + 2(M + 1)} = \frac{3m_0}{-5.8 + 2 \cdot (-3.43)} = -0.237 m_0$$

in [BelyakovBurdov2008]. The latter definition is consistent to our implementation of the k.p Hamiltonian. The discrepancy of these equations arises because there are two different definitions of the L, M parameters available in the literature.

### Comparison of nextnano<sup>3</sup> and nextnano++

Figure 3.20.27.3 compares the *nextnano<sup>3</sup>* results with the *nextnano++* results. The results of both simulators are in excellent agreement.

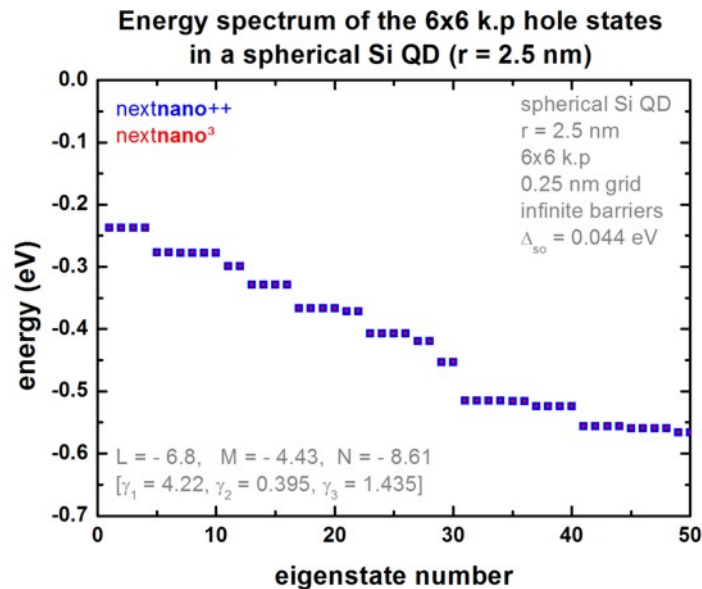


Figure 3.20.27.3: Energy spectrum of the 6-band k.p hole states in a spherical Si QD (Comparison *nextnano++* and *nextnano<sup>3</sup>*).

### Additional comment for experts

For this particular geometry, the eigenvalues are highly degenerate, not only due to spin, but also due to geometry. This might cause problems for certain eigenvalue solvers as they might miss some of these degenerate eigenvalues. So the tool should be used with care. In our case, the ‘chearn’ eigenvalue solver (Arnoldi method that uses Chebyshev polynomials as preconditioner) missed some degenerate eigenvalues. So probably one has to adjust some eigenvalue solver parameters to increase the accuracy. For this reason it is of great advantage if any numerical software has redundancy in terms of several eigensolvers where one can choose from in order to check results for consistency and accuracy, as well as performance.

---

*Last update: 2025/06/25*

## 3.20.28 — DEV — Ultrathin-body DG MOSFET with 2-nm channel

### Warning

This tutorial is under development.

---

- *Double Gate MOSFET*
  - *Input file*
  - *Electron densities*
  - *I-V characteristics*
- 

### Files for the tutorial

- *DG-MOSFET-2-nm\_zb\_IV\_2D\_classical.in* (Figure 3.20.28.2, Figure 3.20.28.4, Figure 3.20.28.7)
- *DG-MOSFET-2-nm\_zb\_IV\_2D\_quantum.in* (Figure 3.20.28.2, Figure 3.20.28.6)
- *DG-MOSFET-2-nm\_zb\_IV\_3D\_classical.in* (Figure 3.20.28.2, Figure 3.20.28.4, Figure 3.20.28.7)

### Important variables:

- `$Temperature` – temperature of the crystal and electrons
- `$SourceDrainVoltage` – bias between the source and drain
- `$DopingConcentration` – doping concentration of the source and drain

### Important output files

- *Structure\last\_region.avs.fld* (Figure 3.20.28.2)
  - *Structure\contact.avs.fld* (Figure 3.20.28.2)
  - *bias\_xxxxx\density\_electron.avs.fld* (Figure 3.20.28.4, Figure 3.20.28.5, Figure 3.20.28.6)
  - *IV\_characteristics.dat* (Figure 3.20.28.7)
-

## Double Gate MOSFET

This tutorial aims to simulate the I-V characteristics of a double gate metal oxide semiconductor field effect transistor (DG MOSFET). The main idea of a DG MOSFET is to control the Si channel very efficiently by choosing the Si channel width to be very small and by applying a gate contact to both sides of the channel. This concept helps to suppress short channel effects and leads to higher currents as compared with a MOSFET having only one gate.

The geometry of the simulated Double Gate MOSFET structure is shown in Figure 3.20.28.1. The width of the Si channel is 2 nm. The distance between the two gates is 6 nm, i.e., the isolating SiO<sub>2</sub> is 2 nm thick on each side. The width of the two gates is 20 nm. The distance between source and drain is 60 nm. The widths and the lengths of source, drain, left, and right doped source regions are 10 nm x 10 nm each. The length of the 2 nm Si channel (without the square doped source and drain regions) is 40 nm.

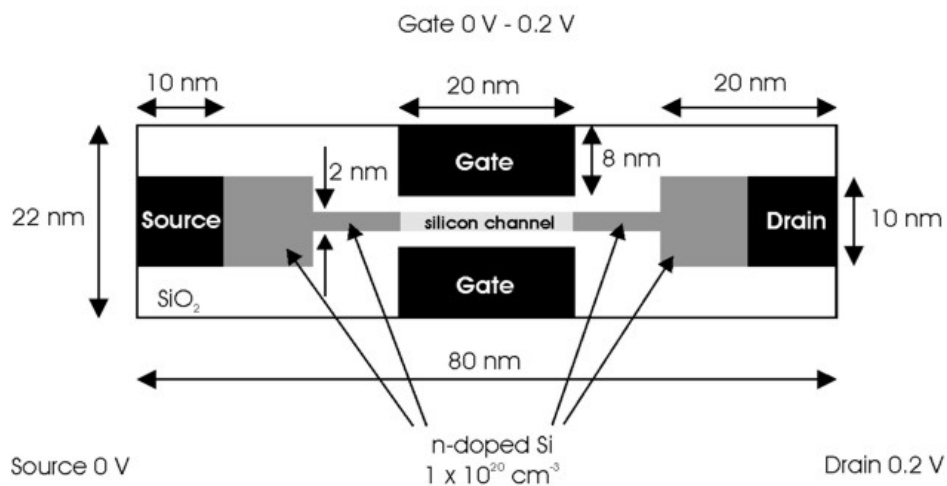


Figure 3.20.28.1: Geometry of the simulated Double Gate MOSFET.

The material regions defined in the input file for the *nextnano++* simulations are shown in Figure 3.20.28.2. The blue squares (Si) are n-doped with a concentration of  $1 \cdot 10^{20} \text{ cm}^{-3}$ . The 2 nm channel is n-doped with the same concentration from 20 nm to 30 nm and from 50 nm to 60 nm.

A constant bias of 0.0 V and 0.2 V is applied to source and drain, respectively. At the two gates we apply Schottky barriers of 3.443 eV, and sweep over the applied bias from 0 V to 1 V.

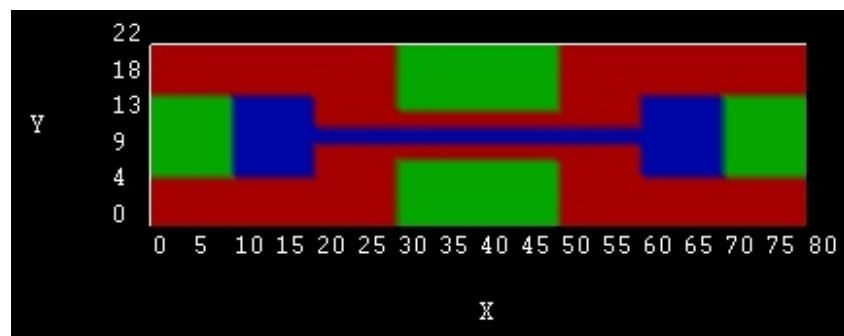


Figure 3.20.28.2: Schematic top view of the material regions defined in the *nextnano++* simulations.

The numerical grid employed in the simulations is shown in Figure 3.20.28.3.

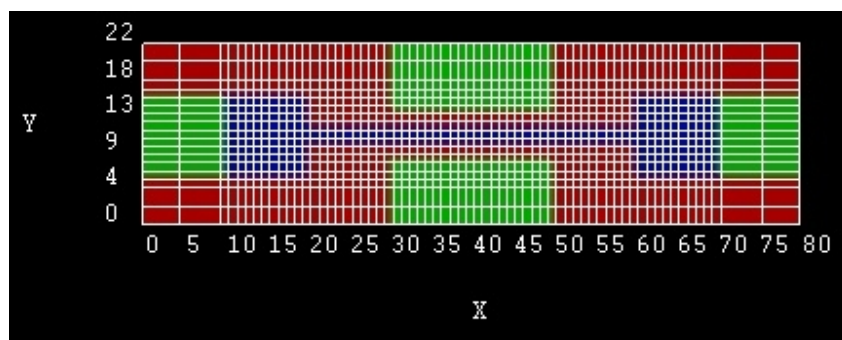


Figure 3.20.28.3: Grid lines of the Double Gate MOSFET

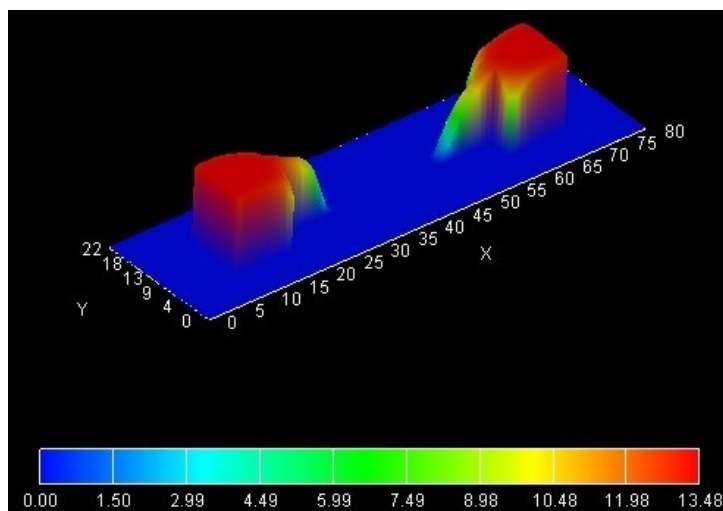
### Input file

For the simulations, the following parameters, which are specified in the corresponding input file *DG-MOSFET-2-nm\_zb\_IV\_2D\_classical.in*, are used:

- The lattice temperature is taken to be 300 Kelvin.
- The classical current and nonlinear Poisson equations are solved self-consistently without including the effect of strain.
- A two-dimensional simulation is performed. The overall simulation domain, that is the real space region in which the device is defined, is taken to be a rectangle having the size 22 nm x 80 nm.

### Electron densities

In Figure 3.20.28.4 the electron density inside the MOSFET structure at 0 V is shown. The corresponding data is contained in the file *bias\_00000\density\_electron.avs.fld*.

Figure 3.20.28.4: Electron density in units of  $1 \cdot 10^{18} \text{cm}^{-3}$  at 0 V gate voltage.

In Figure 3.20.28.5 the electron density inside the MOSFET structure at 0.2 V is shown. The corresponding data is contained in the file *bias\_00002\density\_electron.avs.fld*. One can clearly see that the electron density has the highest values at the Si – SiO<sub>2</sub> interfaces.

For comparison, Figure 3.20.28.6 shows the quantum mechanical electron density inside the MOSFET structure at 0.2 V. The corresponding input which includes the quantum mechanical computation of the charge density is *DG-MOSFET-2-nm\_zb\_IV\_2D\_quantum.in*. One can clearly see that the electron density has the highest values in the middle of the channel and not at the Si – SiO<sub>2</sub> interfaces. This is because the wave functions tend to zero at the Si – SiO<sub>2</sub> interfaces. The peak values in the source and drain regions are due to classical densities because the quantum region did not extend over the whole source and drain regions.

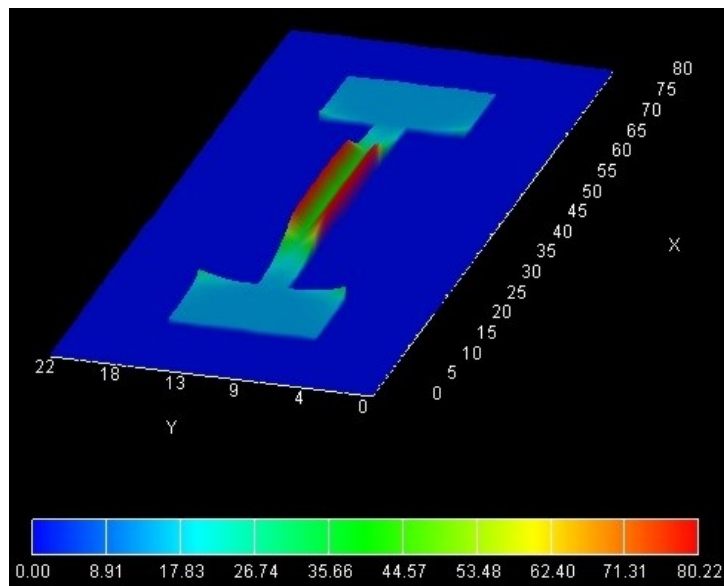


Figure 3.20.28.5: Electron density in units of  $1 \cdot 10^{18} \text{cm}^{-3}$  at 0.2 V gate voltage.

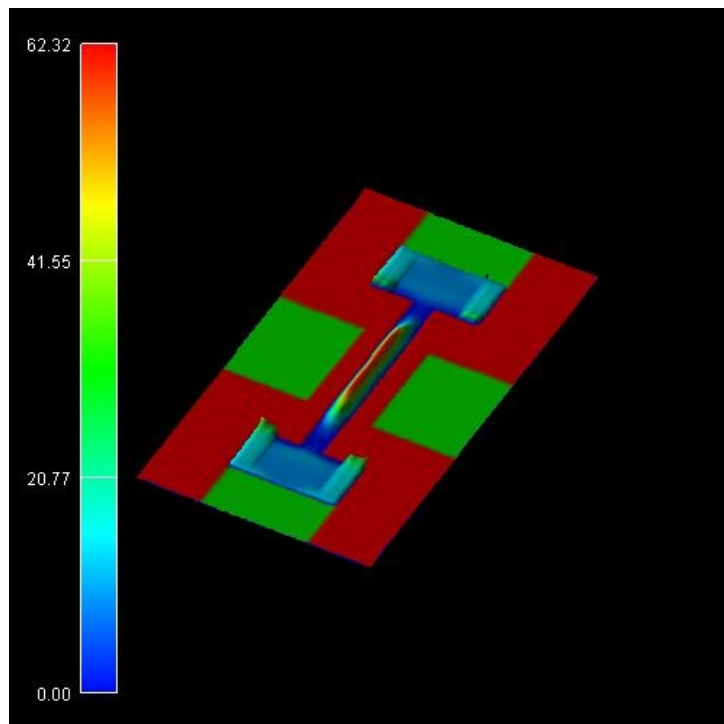


Figure 3.20.28.6: Quantum mechanical electron density in units of  $1 \cdot 10^{18} \text{cm}^{-3}$  at 0.2 V gate voltage.

### I-V characteristics

In order to test the implementation of the three-dimensional drift-diffusion current, we performed a three-dimensional simulation of the Double Gate MOSFET. The corresponding input file is *IV\_DG-MOSFET\_Si\_3D\_classical\_nnp*, where we assume complete ionization of the doping atoms. We further assume that the structure is homogeneous along the  $z$ -direction and assume the  $z$ -direction to be 10 nm long with grid spacing of 2 nm. The calculated current values in units of [A] can be found in *IV\_characteristics.dat*. The current has to be divided by the length of the device along the  $z$ -direction, i.e. by 10 nm, in order to obtain it in units of [A/m]. Figure 3.20.28.7 confirms that the 3D results are in agreement with the 2D results.

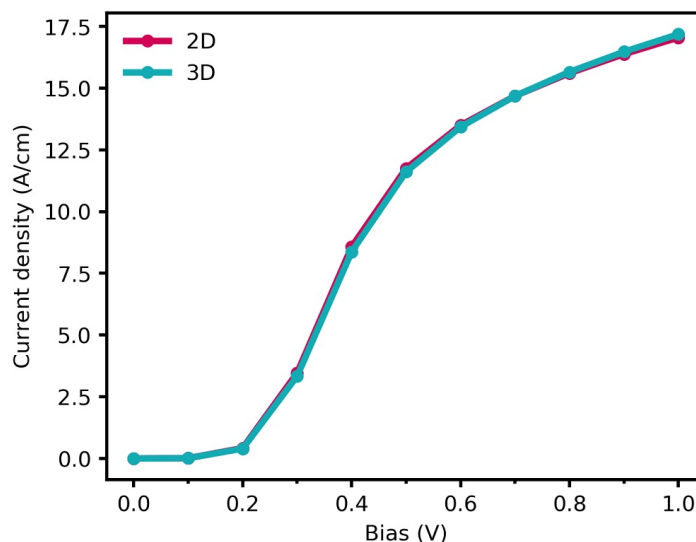


Figure 3.20.28.7: Comparison of the current-voltage characteristics between 2D and 3D simulations.

Last update: 21/10/2024

### 3.20.29 I–V characteristic of GaAs p–n junction | 1D/2D/3D

#### Warning

This tutorial is under construction

#### Input Files:

- *pn\_junction\_GaAs\_ForwardBias\_1D\_nnp.in*
- *pn\_junction\_GaAs\_ForwardBias\_2D\_nnp.in*
- *pn\_junction\_GaAs\_ForwardBias\_3D\_nnp.in*

#### Scope:

This tutorial shows how to perform bias sweeps to compute IV curves.

#### Most relevant keywords:

- `contacts{ ohmic{ bias } }`
- `contacts{ ohmic{ steps } }`

#### Output Files:

*IV\_characteristics.dat*

## Introduction

In the present tutorial we are concerned with the question of how to determine the I-V characteristics of a device. For this purpose, one side of the device is biased, and the simulation is repeatedly executed for a range of different voltages. The *nextnano++* tool offers a convenient way to perform this bias sweep. The computed current and voltage values are automatically collected in one file. In what follows, we simulate a simple p-n junction (see also [p-n junction tutorial](#)), to demonstrate the usage of the keywords which are relevant to trigger the bias sweep.

## Input File

First, two contact regions at both ends of the structure are needed: one as source and the other as drain channel. The contact regions will allow us to bias the structure by applying an explicit voltage to either side of the device.

```
structure{
    ...

    region{
        line{ x = [ -$BOUNDARY, -$SIZE] } # contact on left device boundary
        contact{ name = leftgate }        # contact name
    }
    region{
        line{ x=[ $SIZE, $BOUNDARY] }     # contact on right device boundary
        contact{ name = rightgate }        # contact name
    }

    ...
}
```

The actual properties of the contacts are specified inside the group `contacts{ }`. There are several contact types available (e.g. `ohmic{}`, `schottky{}`, `fermi{}`, ...), each imply different boundary conditions which are applied to the electrostatic potential  $\phi(x)$ . In our case we choose `ohmic{}` contacts.

The voltage on the right side is set to zero (bias = 0 V) and the left contact is biased. In order to sweep over different voltages automatically, the bias for the left contact is to be specified as a vector with start and end value (bias = [V<sub>start</sub>, V<sub>end</sub>]). The attribute `steps` specifies the total number of voltage values.

```
contacts{
    ohmic{ # left contact
        name = leftgate           # refer to region labeled 'leftgate'
        bias = [ 0 , 1.0]         # [V] start and end value of bias sweep
        steps = 20                # number of sweep values
    }
    ohmic{ # right contact
        name = rightgate          # refer to region labeled 'rightgate'
        bias = 0.0                # [V] unbiased
    }
}
```

For simulating charge carrier transport the Poisson and Current equation are solved self consistently. It is important to use proper convergence parameter inside the group `run{ }`.

### Note

It is important to be aware that applying different voltages change the physical properties of the system, e.g. the electric field, and therefore it is not guaranteed that one set of convergence parameters are applicable to all voltages of the sweep.

```
poisson{
  charge_neutral{} # initialize Fermi levels in the contacts that charge_
  ↪neutrality is obtained

  # output settings
  output_potential{}
  output_electric_field{}
}

currents{
  electron_mobility{
    low_field_model = minimos # mobility model
  }
  hole_mobility{
    low_field_model = minimos # mobility model
  }

  recombination_model{ # recombination models
    SRH      = yes
    Auger    = yes
    radiative = yes
  }

  insulator_bandgap = 0.5 # initialization parameter

  minimum_density_electrons = 1e-10 # convergence parameter
  minimum_density_holes     = 1e-10 # convergence parameter
  maximum_density_holes     = 1e15  # convergence parameter

  # output settings
  output_fermi_levels{}
  output_currents{ }
  output_mobilities{}
  output_recombination{}
}

run{
  current_poisson{
    iterations      = 1000 # max iteration
    current_repetitions = 10 # current repetition
    alpha_fermi      = 0.7 # under-relaxation parameter
    residual_fermi    = 1e-12 # desired residual of Fermi levels
    output_log        = yes # information about convergence behavior
  }
}
```

## Results

When the input file is executed, simulation results for each bias value are written in separate folders. These are located in the output folder of the simulation under *bias\_xxxxx* and contain e.g. band edges, electric fields, convergence behaviors, etc.

The output folder also contains a file with the combined current-voltage values. The corresponding file is labeled *IV\_characteristics.dat*. The I-V curve, as presented in [Figure 3.20.29.1](#), can be directly visualized in *nextnanomat*.

### Input Files for *nextnano*<sup>3</sup>:

- *pn\_junction\_GaAs\_1D\_ForwardBias\_nn3.in*



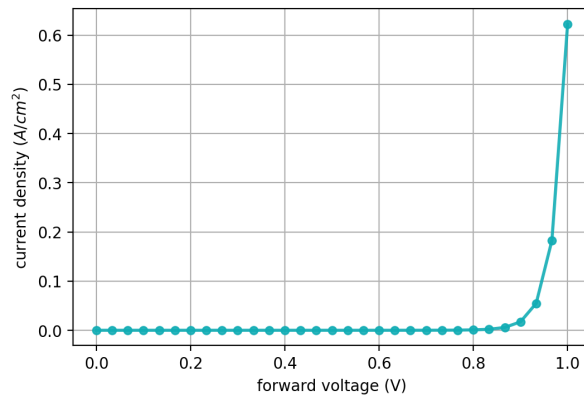


Figure 3.20.29.1: Current density as function of applied bias (1d simulation)

Last update: nnnn/nn/nn

### 3.20.30 — DEV — I–V characteristics of n-doped Si structure

#### Input files:

- *I-V\_n-doped-Si\_1D\_nnp.in*
- *I-V\_n-doped-Si\_2D\_nnp.in*
- *I-V\_n-doped-Si\_3D\_nnp.in*
- *I-V\_nin-doped-Si\_1D\_nnp.in*
- *I-V\_nin-doped-Si\_2D\_nnp.in*
- *I-V\_nin-doped-Si\_3D\_nnp.in*

#### Scope:

This tutorial aims to simulate the I-V characteristics of n-doped and n-i-n doped Si structures.

#### Output files:

- *IV\_characteristics.dat*
- *bias\_xxxx/bandedges.dat*

### I-V characteristics of an n-doped Si structure

#### Structure

The structure we are dealing with consists of bulk Si that is sandwiched between two contacts. The whole structure has the following dimensions (see also):

- along *x*-axis: 20 nm (1 nm contact, 18 nm Si, 1 nm contact)
- along *y*-axis: 5 nm

The Si is n-type doped with a donor concentration of  $N_D = 1 \cdot 10^{20} \text{ cm}^{-3}$ . The energy level is 0.044 eV below the conduction band edge. This leads to an electron density of  $n = 13.48 \cdot 10^{18} \text{ cm}^{-3}$ , which corresponds to the concentration of the ionized donors. The Fermi level  $E_F$  is taken to be at 0 eV in an equilibrium simulation, i.e.  $V = 0 \text{ V}$ . The distance of the conduction band from the Fermi level can be calculated in the following way:

- For the effective electron mass at the  $\Delta$ -point we have:

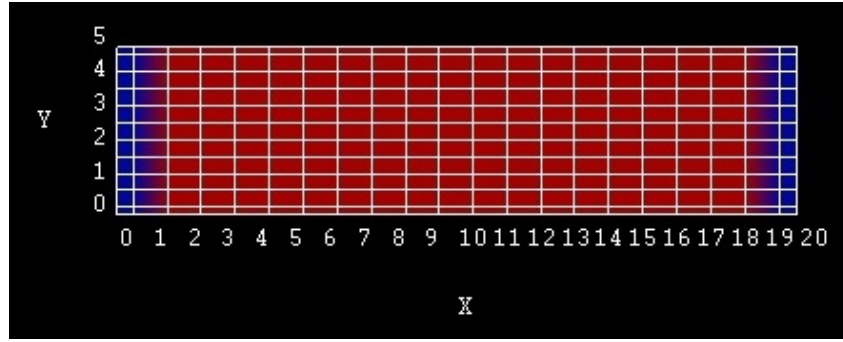


Figure 3.20.30.1: Simulated structure consisting of a left and right contact (blue) and n-doped Si layer (red).

$$m_e = m_{e,\text{DOS}}^* = (m_l \cdot m_t \cdot m_t)^{\frac{1}{3}} = (0.916 \cdot 0.19^2)^{\frac{1}{3}} m_0 = 0.321 m_0,$$

where  $m_l$  is the longitudinal and  $m_t$  is the transversal mass of the effective mass tensor.

- The effective density of states reads:

$$N_c = 12 \cdot \left( \frac{2\pi m_e k_B T}{h^2} \right)^{\frac{3}{2}} = 12 \cdot (0.321 \cdot 0.026 \cdot 2.0886 \cdot 10^{14})^{\frac{3}{2}} = 12 \cdot 2.282 \cdot 10^{18} \text{ cm}^{-3} = 2.738 \cdot 10^{19} \text{ cm}^{-3},$$

where the factor of 12 arises due to the six-fold degeneracy of Si at  $\Delta$  and the two-fold spin degeneracy. Similarly, we obtain the effective density of states for holes:

$$\begin{aligned} N_{v,\text{hh}} &= 9.875 \cdot 10^{18} \text{ cm}^{-3}, \\ N_{v,\text{lh}} &= 1.502 \cdot 10^{18} \text{ cm}^{-3}. \end{aligned}$$

Note that heavy and light holes are degenerate for  $k = 0$ , i.e.  $N_v = N_{v,\text{hh}} + N_{v,\text{lh}} = 1.1377 \cdot 10^{19} \text{ cm}^{-3}$ .

- The Semiconductor equation is given by

$$np = n_i^2 = N_c N_v \exp\left(-\frac{E_{\text{gap}}}{k_B T}\right) = N_c \cdot 1.138 \cdot 10^{19} \text{ cm}^{-3} \exp\left(-\frac{1.095}{0.026}\right) = 1.238 \cdot 10^{20} \text{ cm}^{-6},$$

with  $E_{\text{gap}} = 1.095 \text{ eV}$ ,  $n_i = 1.113 \cdot 10^{10} \text{ cm}^{-3}$  and  $p = n_i^2/n = 9.185 \text{ cm}^{-3}$ .

- The occupation of the different energy states can either be described by Maxwell-Boltzmann statistics:

$$\begin{aligned} n(T) &= N_c(T) \exp\left(\frac{E_F - E_c}{k_B T}\right), \\ p(T) &= N_v(T) \exp\left(\frac{E_v - E_F}{k_B T}\right), \end{aligned}$$

or Fermi-Dirac statistics:

$$\begin{aligned} n(T) &= N_c(T) \mathcal{F}_{1/2}\left(\frac{E_F - E_c}{k_B T}\right), \\ p(T) &= N_v(T) \mathcal{F}_{1/2}\left(\frac{E_v - E_F}{k_B T}\right), \end{aligned}$$

where  $\mathcal{F}_{1/2}$  is the Fermi-Dirac integral of order 1/2 multiplied by the factor  $2/\sqrt{\pi}$  (i.e.  $\mathcal{F}_{1/2}$  includes the Gamma pre-factor)

When using the Maxwell-Boltzmann statistics as an approximation, we obtain:

$$E_c = k_B T \ln \left( \frac{N_c}{n} \right) = 0.026 \text{ eV} \cdot \ln \left( \frac{2.738 \cdot 10^{19} \text{ cm}^{-3}}{13.478 \cdot 10^{18} \text{ cm}^{-3}} \right) = 0.026 \text{ eV} \cdot \ln(2.031) = 18.3 \text{ meV},$$

$$E_v = -k_B T \ln \left( \frac{N_v}{p} \right) = -0.026 \text{ eV} \cdot 42.538 = -1.099 \text{ eV}.$$

Note that *nextnano++* uses the Fermi-Dirac integrals (Fermi-Dirac statistics), where the following results are obtained:  $E_c = 13.85 \text{ meV}$  and  $E_v = -1.0815 \text{ eV}$ .

## Results

We sweep the voltage at the right contact from 0.0 V to 0.2 V in 10 steps. The input files used for the simulations are *I-V\_n-doped-Si\_1D\_nnp.in*, *I-V\_n-doped-Si\_2D\_nnp.in* *I-V\_n-doped-Si\_3D\_nnp.in*. The calculated current density for each bias point can be found in *IV\_characteristics.dat*. The resulting I-V characteristics is depicted in Figure 3.20.30.2.

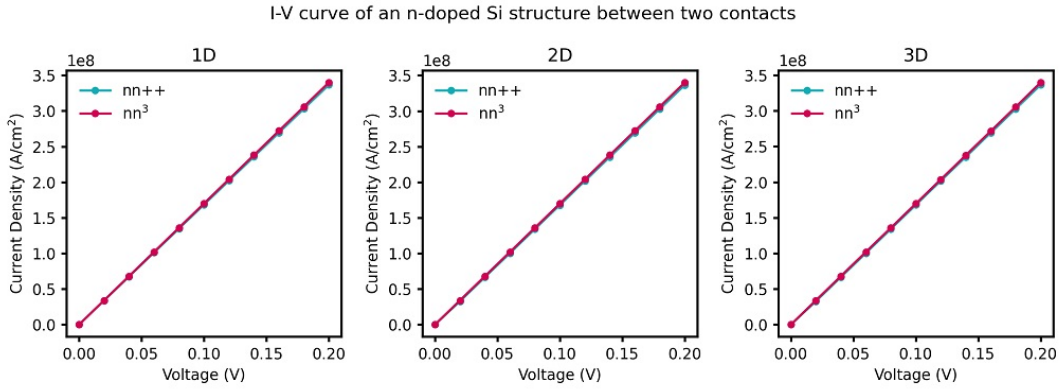


Figure 3.20.30.2: Simulated I-V characteristics of an n-doped Si structure using constant mobility model.

The *nextnano++* results are in agreement with the I-V characteristics obtained with *nextnano*<sup>3</sup>. The units for the current in a 2D simulation are [A/m]. Dividing this two-dimensional current value by the width of the device (in our case 5 nm) we obtain the current in units of [A/cm<sup>2</sup>], which is the usual unit of a 1D simulation. As our simple 2D example structure is basically equivalent to a 1D structure we can easily compare our 2D results with the 1D results to check for consistency. It is also possible to perform a 3D simulation. In this case, the units for the three-dimensional current are [A]. Dividing by the area of the device of 25 nm<sup>2</sup>, we obtain the 1D units of [A/cm<sup>2</sup>].

## I-V characteristics of an n-i-n-doped Si structure

### Structure

The second example is an n-i-n (n-doped, intrinsic, n-doped) Si structure, which is shown in Figure 3.20.30.3. The width of the intrinsic region is 14 nm, and the n-doped regions are both 2 nm wide.

## Results

In Figure 3.20.30.4 the current-voltage (I-V) characteristic is shown. The input files used for the simulations are *I-V\_nin-doped-Si\_1D\_nnp.in*, *I-V\_nin-doped-Si\_2D\_nnp.in* *I-V\_nin-doped-Si\_3D\_nnp.in*. The data of the I-V curve can be found in the corresponding file *IV\_characteristics.dat*.

In order to compare the results from 1D, 2D and 3D simulations, we have divided the 2D current by the width of the device (in our case 5 nm) and the 3D current by the cross-section area of the device of (in our case 25 nm<sup>2</sup>), to get the current density in units of [A/cm<sup>2</sup>]. The obtained results are in perfect agreement.

Figure 3.20.30.5 shows the conduction band profile (*bias\_XXXX/bandedges.dat*) for different voltages.

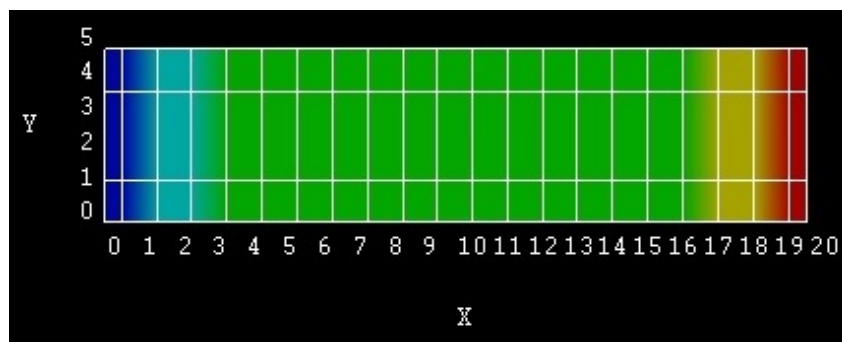


Figure 3.20.30.3: Simulated n-i-n structure consisting of a left contact (dark blue), n-doped Si (light blue), intrinsic Si (green), n-doped Si (yellow) and right contact (red).

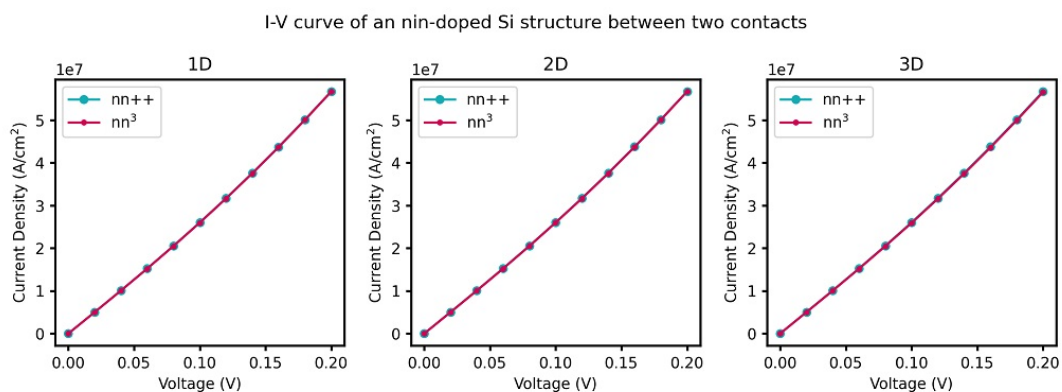


Figure 3.20.30.4: Simulated I-V characteristics of the n-i-n doped Si structure using constant mobility model.

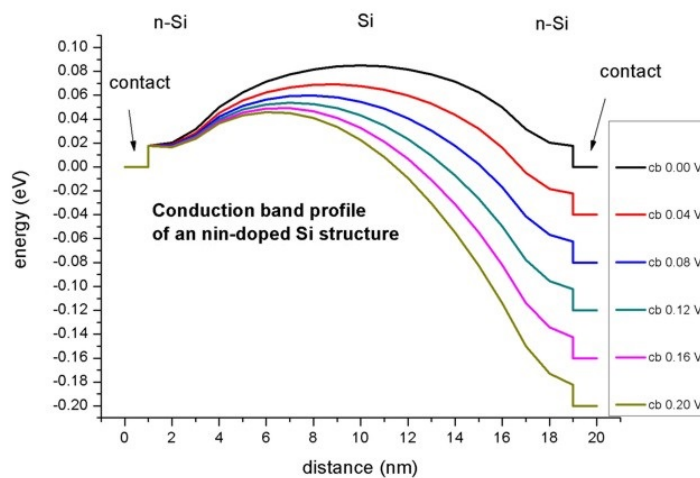


Figure 3.20.30.5: Simulated conduction band profile of the n-i-n Si structure for different voltages.

Last update: 17/07/2024

### 3.20.31 Hexagonal shaped GaN quantum dot embedded in AlN (wurtzite)

#### ⚠ Attention

Figures in this tutorial are transferred from old documentation and are consistent with *nextnano*<sup>3</sup> input files. They differ from results obtained with *nextnano++* as these input file has been improved, including change of simulation domain and boundary conditions, to represent results from the cited publication more accurately. However, qualitative tendencies are preserved.

- *Header*
- *Conduction and valence band alignment in AlN/GaN QWs (unstrained)*
- *Conduction and valence band alignment in AlN/GaN QWs (pseudomorphically strained)*
- *Conduction and valence band edges in AlN/GaN QWs (pseudomorphically strained, including piezo- and pyroelectric fields)*
- *Electron and hole wave functions in AlN/GaN QWs*
- *Hexagonal shaped GaN quantum dot embedded in AlN (wurtzite)*

#### Header

#### 📘 Note

The tutorial is based on [Andreev2000].

#### Input files in *examples\quantum\_dots\*:

- *QD\_GaN\_Andreev\_PRB\_2000\_1D\_nnp\_band-offsets.in*
- *QD\_GaN\_Andreev\_PRB\_2000\_1D\_nnp\_strain.in*
- *QD\_GaN\_Andreev\_PRB\_2000\_1D\_nnp\_strain-PzPr-poisson-1b.in*
- *QD\_GaN\_Andreev\_PRB\_2000\_1D\_nnp\_strain-PzPr-poisson-6kp.in*
- *QD\_GaN\_Andreev\_PRB\_2000\_3D\_nnp.in*

#### Scope:

The influence of strain and pyro-/ piezoelectric fields on the electronic structure of hexagonal shaped GaN/AlN quantum dots.

#### Conduction and valence band alignment in AlN/GaN QWs (unstrained)

In this section the input file *QD\_GaN\_Andreev\_PRB\_2000\_1D\_nnp\_band-offsets.in* is used to compute band offsets.

Figure 3.20.31.1 shows the conduction and valence band edge alignment in AlN/GaN structures (unstrained). In AlN, the light hole (LH) is the highest valence band whereas in GaN, this is the heavy hole (HH). We assumed a valence band offset of  $VBO = 0.5$  eV, the conduction band offset is much larger ( $CBO = 2.3$  eV). All material parameters are based on [Andreev2000] although meanwhile better parameters are available.

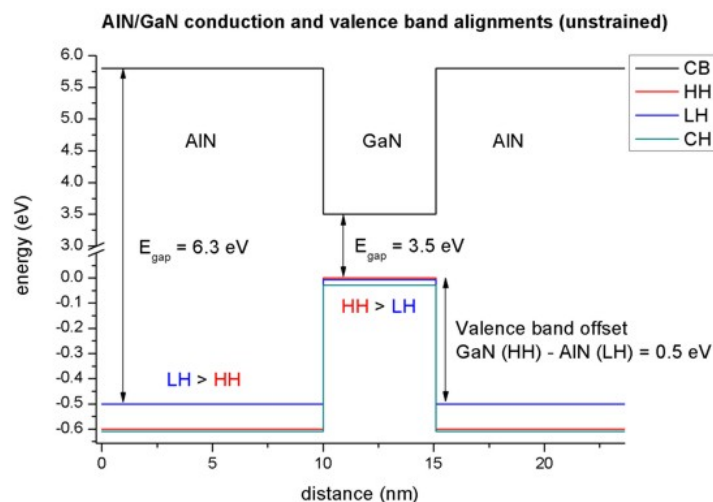


Figure 3.20.31.1: Conduction band edge (CB) and valence band edges (HH, LH, CH) of the 1D AIN/ GaN QD (unstrained).

### Conduction and valence band alignment in AIN/GaN QWs (pseudomorphically strained)

In this section the input file *QD\_GaN\_Andreev\_PRB\_2000\_1D\_nnp\_strain.in* is used to show impact of the strain on the band edges without piezo effects.

Figure 3.20.31.2 shows the conduction and valence band edge alignment in the AIN/GaN structure, which is strained with respect to the AIN substrate. The lattice constants in GaN are larger than in AIN, thus GaN is compressively strained. The AIN band edges are the same as in Figure 3.20.31.1, only the GaN edges have changed:

- The band gap of GaN has increased (compressive strain increases the band gap).
- Now the crystal-field split-hole (CH) in GaN lies above the light hole (LH) and close to the heavy hole (HH).
- The valence band offset has decreased to  $VBO = 0.46$  eV.
- The conduction band offset has decreased to  $CBO = 2.15$  eV.

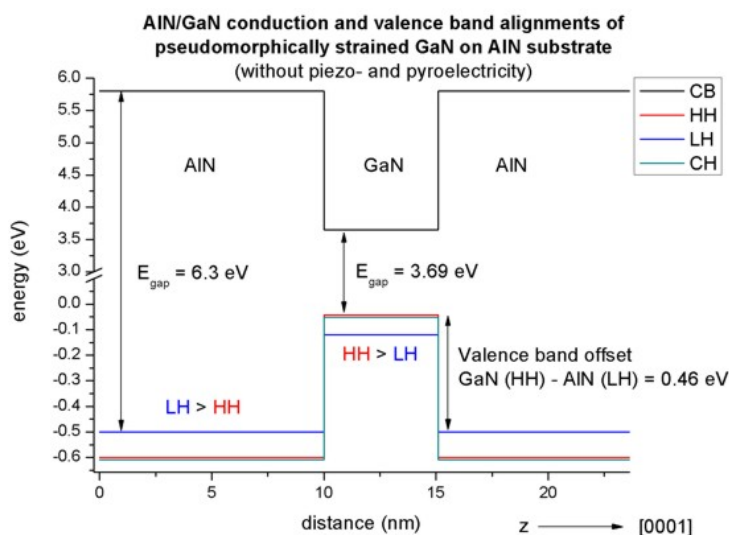


Figure 3.20.31.2: Conduction band edge (CB) and valence band edges (HH, LH, CH) of the 1D AIN/GaN QD (strained on AIN substrate).

### Conduction and valence band edges in AlN/GaN QWs (pseudomorphically strained, including piezo- and pyroelectric fields)

In this section the input file *QD\_GaN\_Andreev\_PRB\_2000\_1D\_nnp\_strain-PzPr-poisson-1b.in* or *QD\_GaN\_Andreev\_PRB\_2000\_1D\_nnp\_strain-PzPr-poisson-6kp.in* can be used to observe piezo effect on the design.

In Figure 3.20.31.3 the effect of piezo- and pyroelectric fields on the band edge is shown. The band edge gets tilted due to the additional electric potential arising from the piezo- and pyroelectric charges. The electrostatic potential which is the solution to the Poisson equation is also shown in Figure 3.20.31.3.

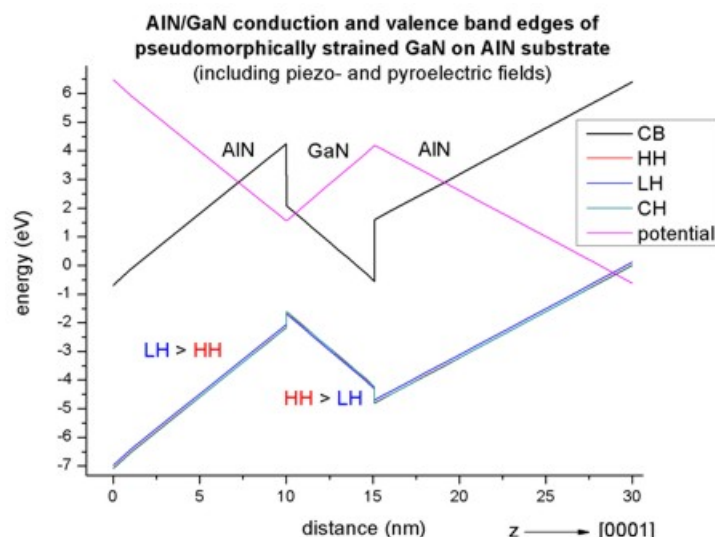


Figure 3.20.31.3: Conduction band edge (CB) and valence band edges (HH, LH, CH) of the 1D AlN/GaN QD (strained on AlN substrate) including piezo- and pyroelectric fields.

### Electron and hole wave functions in AlN/GaN QWs

Figure 3.20.31.4 shows the electron and hole wavefunctions ( $\Psi^2$ ) in a 5.1 nm AlN/GaN/AlN quantum well. For the electrons, the single-band effective-mass approximation was used whereas for the holes the 6-band k.p model was used. The figure shows the four lowest electron eigenstates and the 6 highest valence band eigenstates. All eigenstates are two-fold degenerate due to spin.

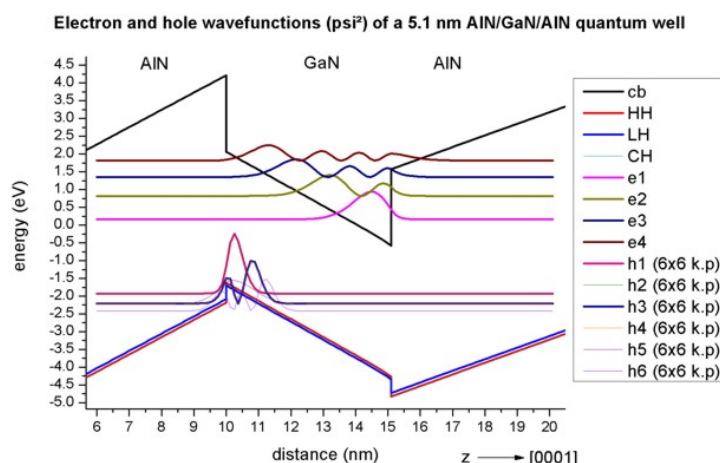


Figure 3.20.31.4: Electron and hole wavefunctions  $\Psi^2$  of a 5.1 nm AlN/GaN/AlN quantum well.



### Hexagonal shaped GaN quantum dot embedded in AlN (wurtzite)

The simulated hexagonal GaN quantum dot (height = 4.1 nm) is embedded in an AlN matrix, input file *QD\_GaN\_Andreev\_PRB\_2000\_3D\_nnp.in*. The wetting layer is 1 nm thick and consists of GaN. The structure and a cross-section of the structure along x-y are shown in Figure 3.20.31.5 and Figure 3.20.31.6, respectively.

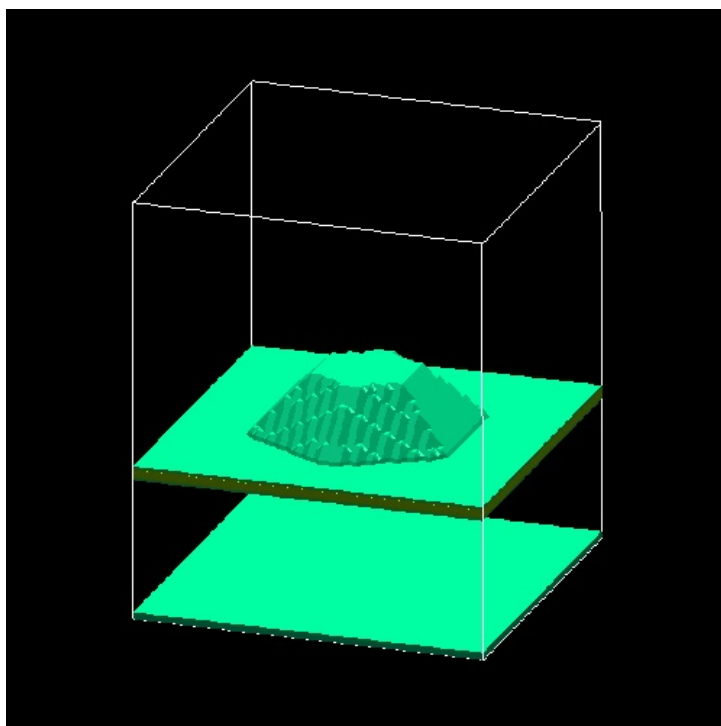


Figure 3.20.31.5: 3D AlN/GaN quantum dot.

The strain tensor components of a line through the center of the quantum dot along the z axis are shown in Figure 3.20.31.7. This figure is related to Fig. 2 (a) in [Andreev2000].

Figure 3.20.31.8 and Figure 3.20.31.9 show the strain tensor components along the [10-10] direction (y direction) for a line through the bottom of the quantum dot and for a line through the wetting layer, respectively. These figures are related to Fig. 2 (b) in [Andreev2000].

The strain induced piezoelectric fields and the pyroelectric fields lead to the electrostatic potential which is shown in Figure 3.20.31.10 and Figure 3.20.31.11. The figures of the potential are related to Fig. 4 in [Andreev2000]. In Figure 3.20.31.10 one can clearly see that the electrostatic potential has its maximum at the top of the QD and its minimum in the wetting layer area just below the QD. Figure 3.20.31.11 shows a cut of the electrostatic potential through the wetting layer plane.

The conduction and valence band edges are shown in Figure 3.20.31.12 and Figure 3.20.31.13. One can clearly see that the conduction band minimum is located in the top of the quantum dot whereas the maximum for the valence band is located inside the wetting layer (WL) (which is equivalent to the bottom of the quantum dot). Thus, one expects the electrons, which are located in the top area of the QD, to be spatially separated from the holes, which are located in the WL (bottom of the QD). The energy scale is in units of [eV]. The figures of the conduction and valence band edges are related to Figs. 5 and 6 in [Andreev2000].

The electron states are located near the top of the quantum dot where the conduction band has a minimum. Figure 3.20.31.14 shows the electron ground state.

The following figures show the six lowest electron states of the quantum dot. The 2<sup>nd</sup> and 3<sup>rd</sup> eigenstates are degenerate, as well as the 4<sup>th</sup>, 5<sup>th</sup> and 6<sup>th</sup>. The figures of the wave functions ( $\Psi^2$ ) are related to Fig. 7 in [Andreev2000].

Version of this tutorial for nextnano<sup>3</sup> can be found [here](#) .



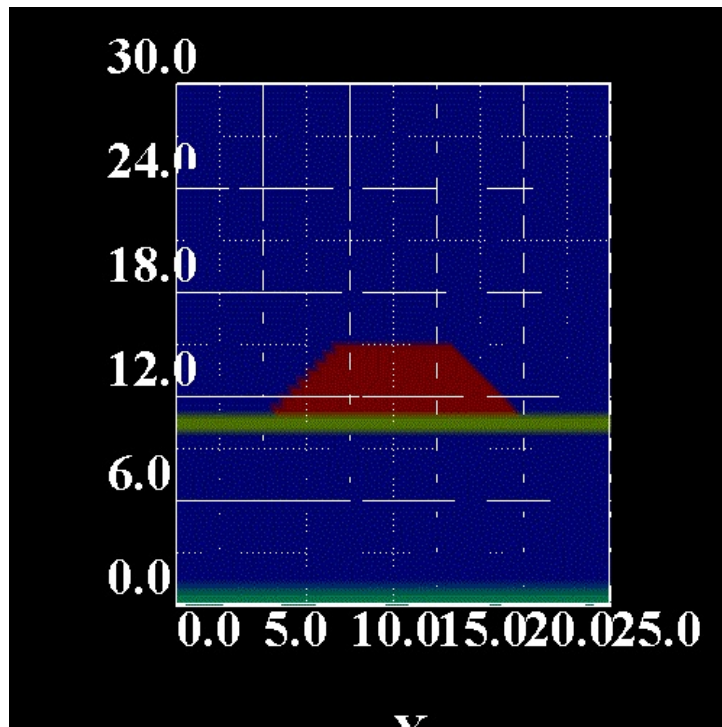


Figure 3.20.31.6: Cross-section of the hexagonal shaped AlN/GaN quantum dot.

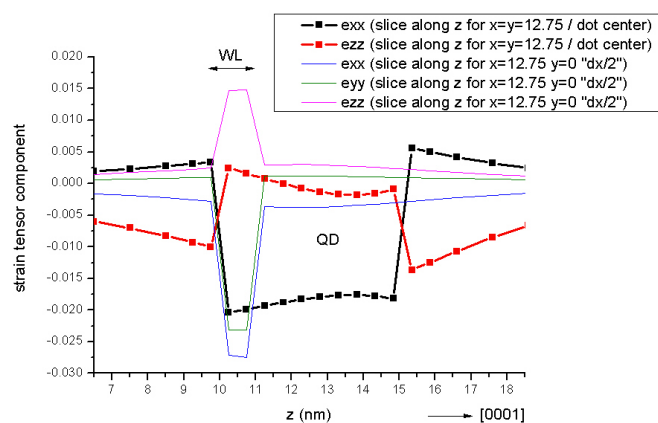


Figure 3.20.31.7: Strain tensor along the z-axis through the points  $(x, y) = (12.75, 12.75)$  nm and  $(x, y) = (12.75, 0)$  nm.

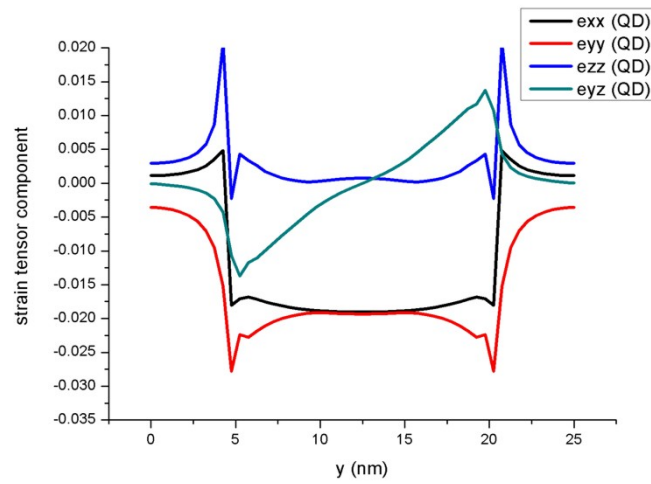


Figure 3.20.31.8: Strain tensor along the y-axis through the quantum dot (QD).

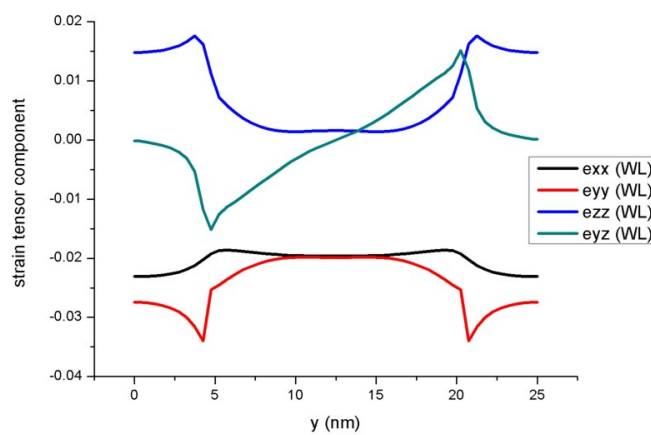


Figure 3.20.31.9: Strain tensor along the y-axis through the wetting layer (WL).

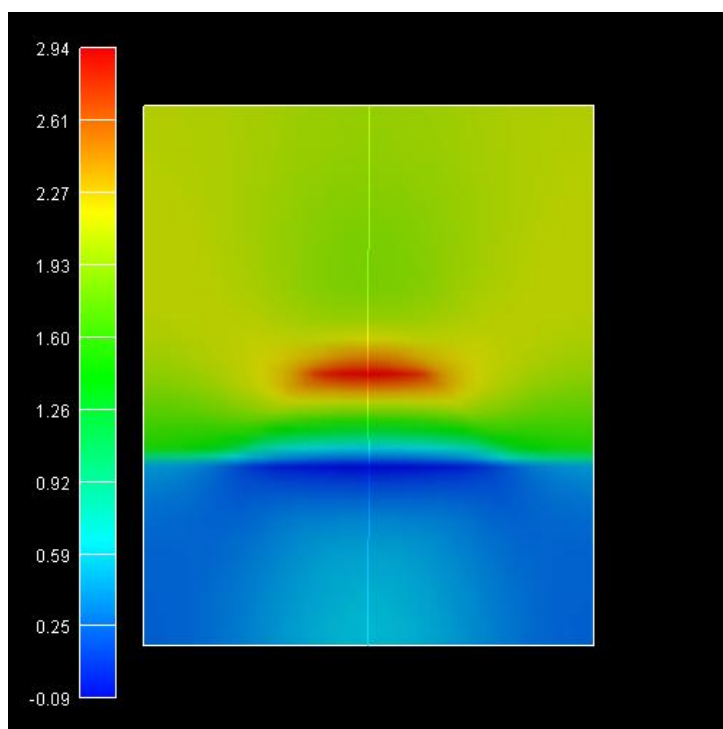


Figure 3.20.31.10: Electrostatic potential inside the quantum dot.

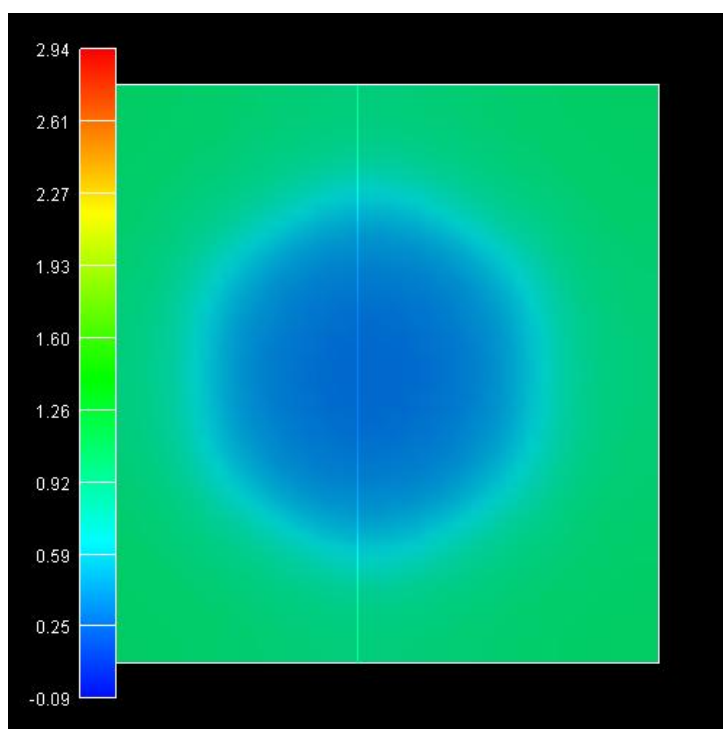


Figure 3.20.31.11: Electrostatic potential inside the wetting layer.

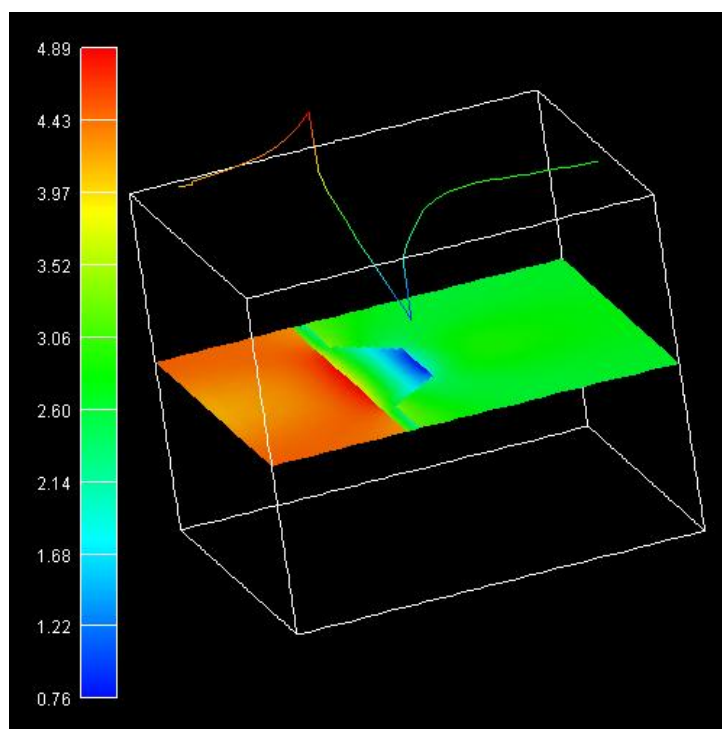


Figure 3.20.31.12: Conduction band edge of the QD.

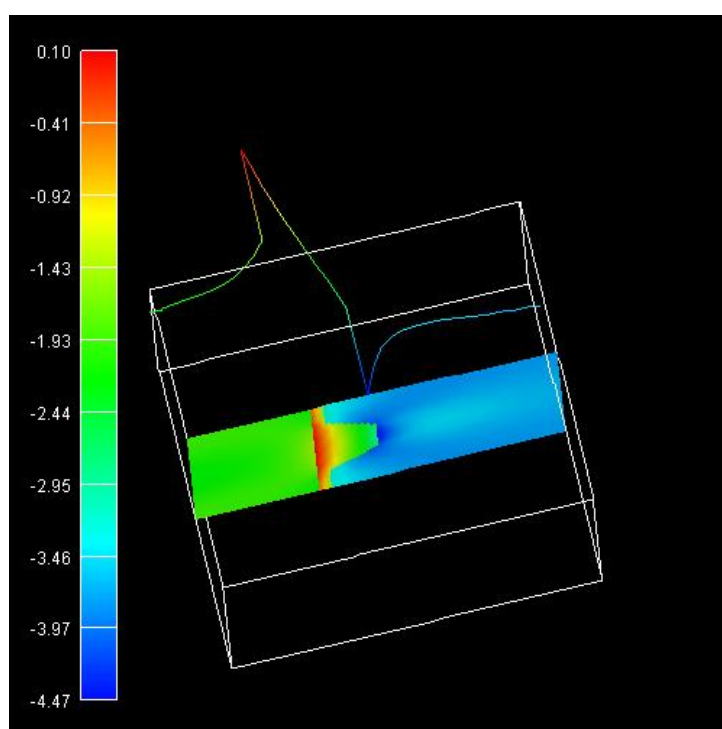


Figure 3.20.31.13: Valence band edge of the QD.

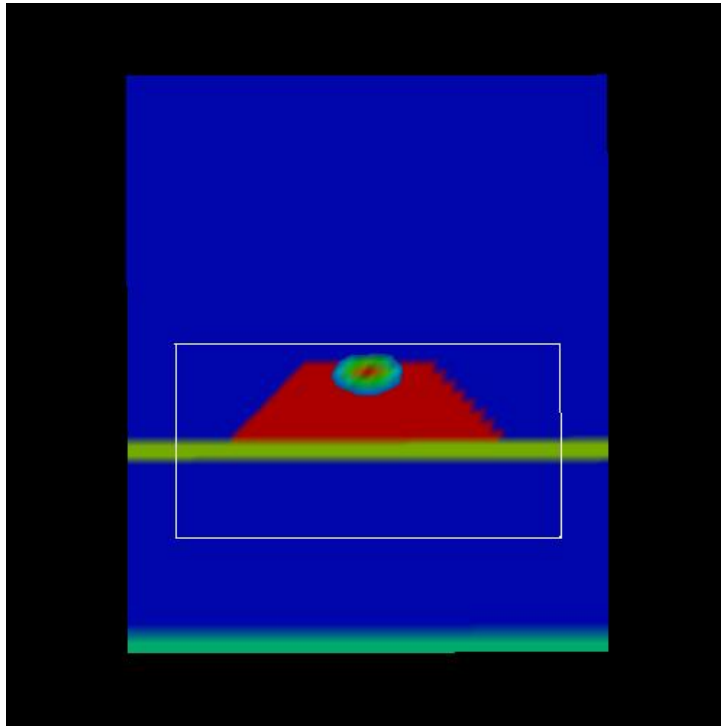


Figure 3.20.31.14: Electron ground state of the QD.

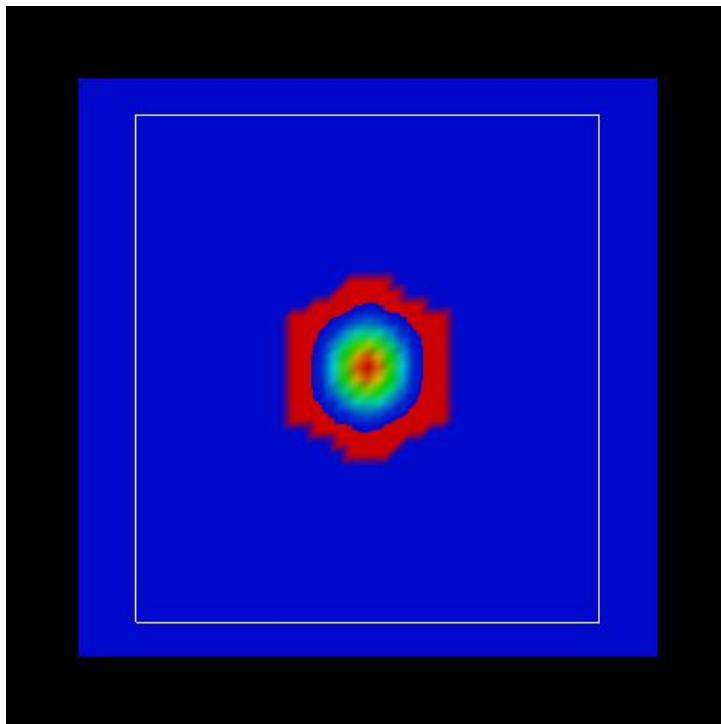


Figure 3.20.31.15: Probability density  $\Psi^2$  of the 1<sup>st</sup> electron state in the QD.

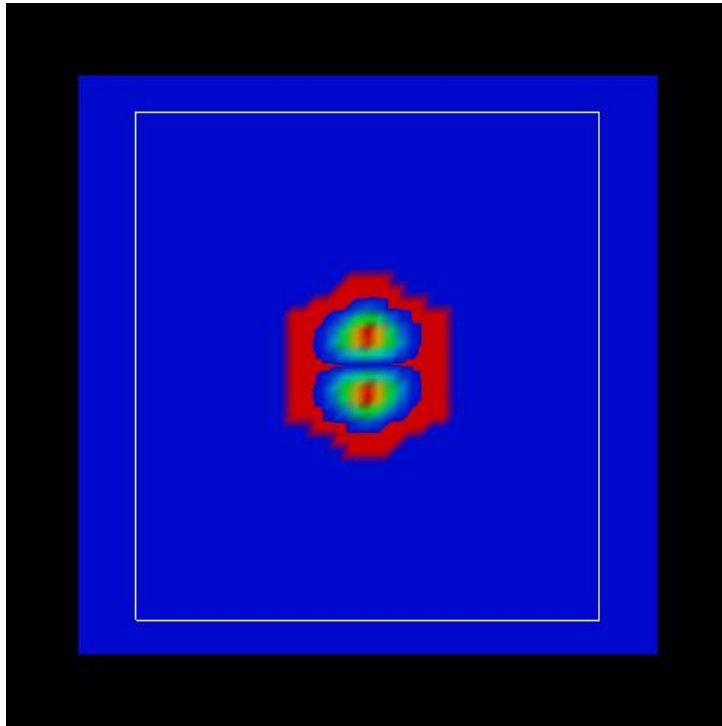


Figure 3.20.31.16: Probability density  $\Psi^2$  of the 2<sup>nd</sup> electron state in the QD.

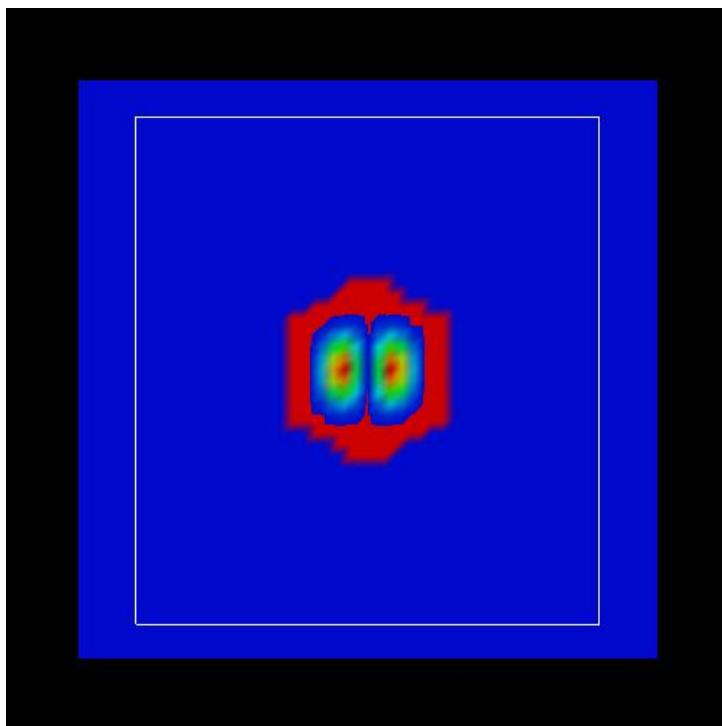


Figure 3.20.31.17: Probability density  $\Psi^2$  of the 3<sup>rd</sup> electron state in the QD.

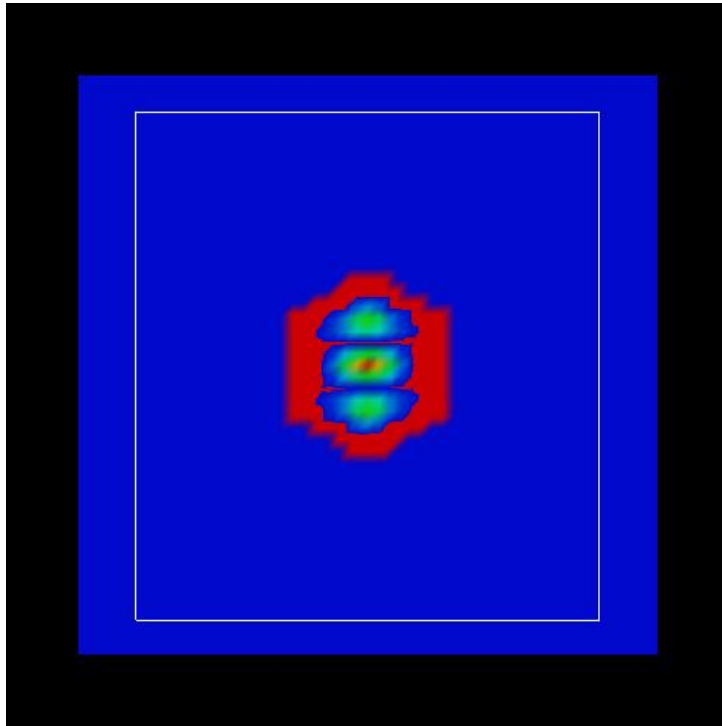


Figure 3.20.31.18: Probability density  $\Psi^2$  of the 4<sup>th</sup> electron state in the QD.

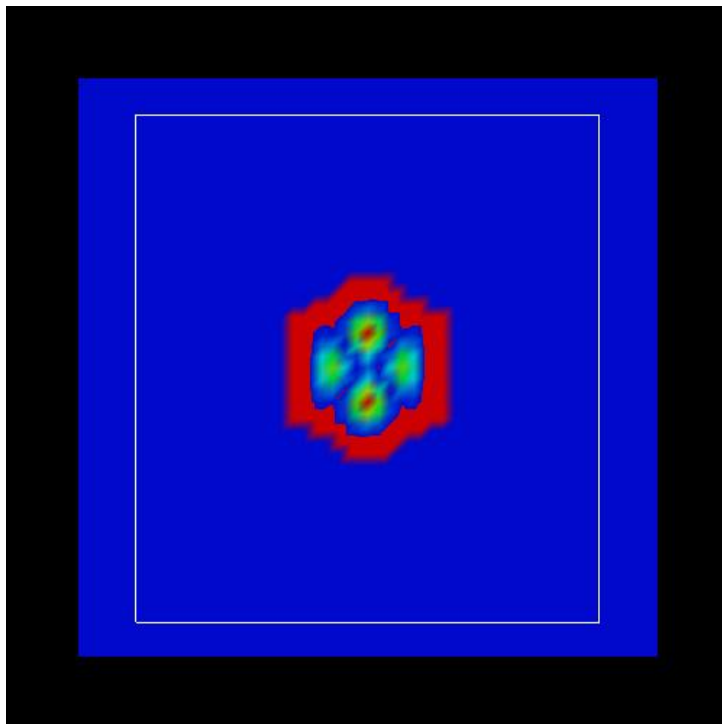


Figure 3.20.31.19: Probability density  $\Psi^2$  of the 5<sup>th</sup> electron state in the QD.

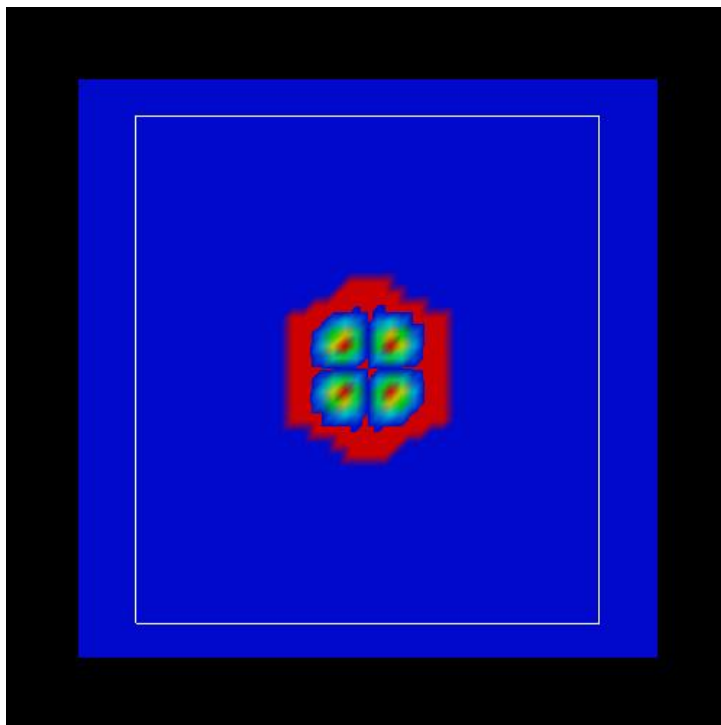


Figure 3.20.31.20: Probability density  $\Psi^2$  of the 6<sup>th</sup> electron state in the QD.

*Last update: 11/07/2024*

### 3.20.32 k.p dispersion of an unstrained GaN QW embedded between strained AlGaIn layers

#### Input files:

- *1DGaN\_AlGaIn\_QW\_k\_zero.nn3*
- *1DGaN\_AlGaIn\_QW\_k\_parallel.nn3*
- *1DGaN\_AlGaIn\_QW\_k\_zero\_10m10.nn3*
- *1DGaN\_AlGaIn\_QW\_k\_parallel\_10m10.nn3*
- *1DGaN\_AlGaIn\_QW\_k\_parallel\_10m10\_whole.nn3*

#### Scope:

In this tutorial we aim to reproduce results of [Park2000]. The material parameters are taken from [ParkChunag2000], except those listed in Table 1 of [Park2000].

#### [0001] growth direction

#### Calculation of electron and hole energies and wave functions for $k_{||} = 0$

Input file: *1DGaN\_AlGaIn\_QW\_k\_zero.nn3*

The structure consists of a 3 nm unstrained GaN quantum well, embedded between 8.4 nm strained  $\text{Al}_{0.2}\text{Ga}_{0.8}\text{N}$  barriers. The AlGaIn layers are strained with respect to the GaN substrate. The GaN quantum well is assumed to be unstrained.

The structure is modeled as a superlattice (or multi quantum well, MQW), i.e. we apply periodic boundary conditions to the Poisson equation.

The growth direction is along the hexagonal axis, i.e. along [0001].

#### Conduction and valence band profile



Figure 3.20.32.1 shows the conduction and valence (heavy hole, light hole and crystal-field split-off hole) band edges of our structure, including the effects of strain, piezo- and pyroelectricity. The ground state electron and the ground state heavy hole wave functions ( $\Psi^2$ ) are shown. Due to the built-in piezo- and pyroelectric fields, the electron wave function are shifted to the right and the hole wave function to the left (Quantum Confined Stark Effect, QCSE)

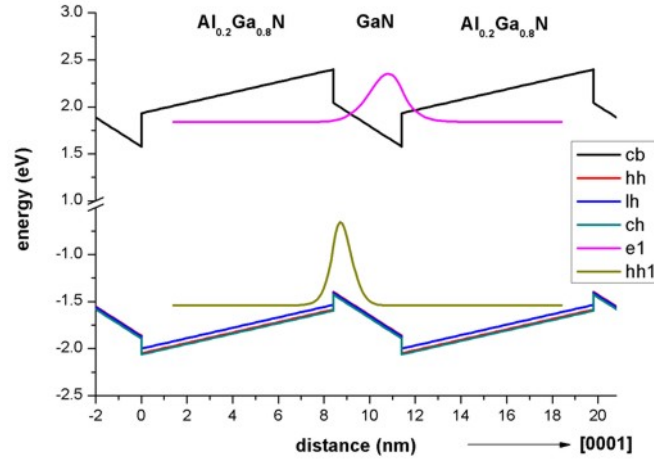


Figure 3.20.32.1: Calculated band edge profile.

### Strain

The strain inside the GaN quantum well layer is zero. The tensile strain in the  $\text{Al}_{0.2}\text{Ga}_{0.8}\text{N}$  barriers has been calculated to be

$$e_{xx} = e_{yy} = \frac{a_{\text{substrate}} - a}{a} = 0.486.$$

[Park2000] gives a value of 0.484.

The output of the strain tensor can be found in this file: *strain\strain\_crystal.dat*

### Piezoelectric polarization

The piezoelectric polarization for the [0001] growth direction is zero inside the GaN QW, because the strain is zero in the QW. In the  $\text{Al}_{0.2}\text{Ga}_{0.8}\text{N}$  barriers, the piezoelectric polarization has been calculated to be  $0.0081 \text{ C/m}^2$  in agreement with Fig. 1(a) of [Park2000] for angle  $\theta = 0$ . The resulting piezoelectric polarization

- at the  $\text{Al}_{0.2}\text{Ga}_{0.8}\text{N}$  / GaN interface  $-0.0081 \text{ C/m}^2$  and
- at the GaN /  $\text{Al}_{0.2}\text{Ga}_{0.8}\text{N}$  interface is  $0.0081 \text{ C/m}^2$ .

### Pyroelectric polarization

The pyroelectric polarization for the [0001] growth direction is  $-0.029 \text{ C/m}^2$  inside the GaN QW. In the  $\text{Al}_{0.2}\text{Ga}_{0.8}\text{N}$  barriers, the pyroelectric polarization has been calculated to be  $-0.0394 \text{ C/m}^2$ . The resulting pyroelectric polarization

- at the  $\text{Al}_{0.2}\text{Ga}_{0.8}\text{N}$  / GaN interface is  $-0.0104 \text{ C/m}^2$  and
- at the GaN /  $\text{Al}_{0.2}\text{Ga}_{0.8}\text{N}$  interface is  $0.0104 \text{ C/m}^2$ .

These results are in excellent agreement with Fig. 1(a) of [Park2000] for angle  $\theta = 0$ .

### Poisson equation

Solving the Poisson equation with periodic boundary conditions (to mimic the superlattice) leads to the following electric fields: Inside the GaN QW the electric field has been calculated to be  $-1.551 \text{ MV/cm}$ . [Park2000] reports an electric field of  $-1.55 \text{ MV/cm}$  inside the QW. The electric field in the AlGa<sub>0.8</sub>N barrier has been found to be  $0.554 \text{ MV/cm}$ .

The output of the electrostatic potential (units [V]) and the electric field (units [kV/cm]) can be found in these files:

- *bias\_00000\potential*
- *bias\_00000\electric\_filed.dat*

### Schrödinger equation

Figure 3.20.32.2 shows the electron and hole wave functions ( $\Psi^2$ ) of the GaN/AlGa<sub>0.2</sub>N structure for  $k_{||} = 0$ . The heavy and light hole wave functions are very similar in shape.

In agreement with [Park2000], we calculated the electron levels within the single-band effective mass approximation and the hole levels within the 6-band k.p approximation.

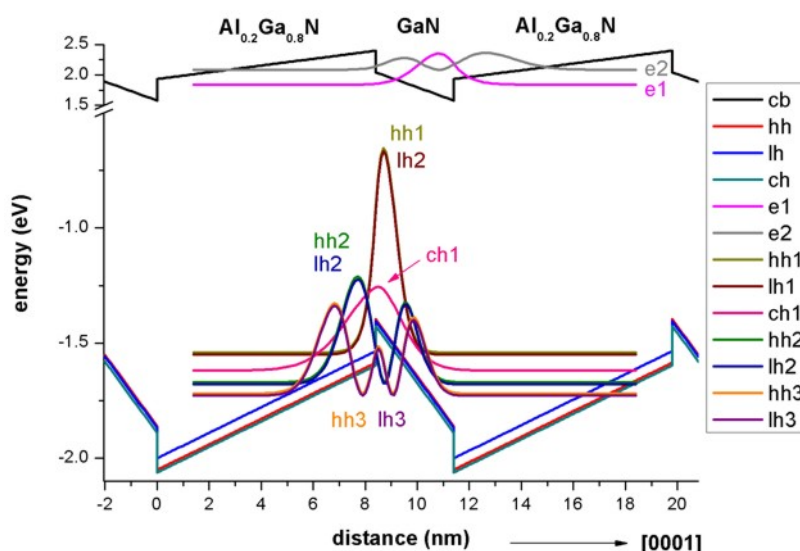


Figure 3.20.32.2: Calculated wave functions of lowest eigenstates.

**$k_{||}$  dispersion: Calculation of the electron and hole energies and wave functions for  $k_{||} \neq 0$ .**

Input file: *1DGaN\_AlGa<sub>0.2</sub>N\_QW\_k\_parallel.nn3*

The grid has a spacing of 0.1 nm leading to a sparse matrix of dimension 1050 which has to be solved for each  $k_{||}$  point for the eigenvalues (and wave functions).

We chose as input:

```
calculate_dispersion{
  num_points = 1849 # This corresponds to 1849  $k_{||}$  points in the 2D ( $k_x, k_y$ ) plane,
  ↳ i.e.  $(2 * 21 + 1) * (2 * 21 + 1) = 1849$ .
}
```

Due to symmetry arguments, we solved the Schrödinger equation only for the  $k_{||}$  points along the line ( $k_x > 0, k_y = 0$ ), i.e. we had to solve the Schrödinger equation 22 times (i.e. to calculate the eigenvalues of a 1050 x 1050 matrix 22 times).

The energy dispersion  $E(k_{||}) = E(k_y, k_z)$  displayed in Figure 3.20.32.3 is contained in this folder: *bias\_00000\Quantum\Dispersion*

Because our quantum well is not symmetric (due to the piezo- and pyroelectric fields), the eigenvalues for spin up and spin down are not degenerate anymore. They are only degenerate at  $k_{||} = 0$ . This lifting of the so-called Kramer's degeneracy in the in-plane dispersion relations is because of the field-induced asymmetry. In Fig. 3 (a) of [Park2000] only the spin-up eigenstates are plotted because the splitting of the Kramer's degeneracy was assumed to be very small.

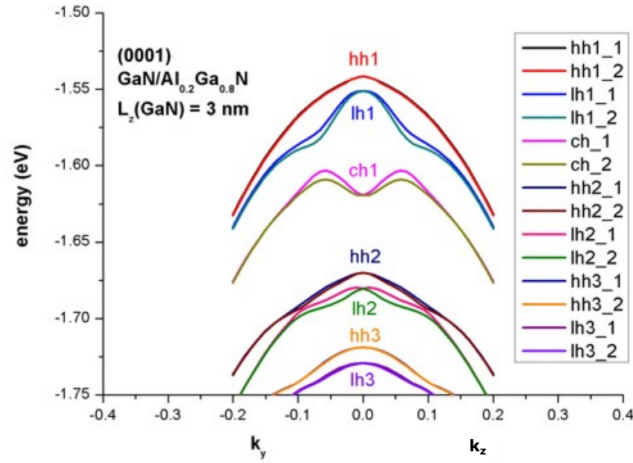


Figure 3.20.32.3: Calculated energy dispersion  $E(k_{||}) = E(k_y, k_z)$ .

### [10-10] growth direction (m-plane)

Input file: `1DGaN_AlGaN_QW_k_zero_10m10.nn3`

If one grows the quantum well along the [10-10] growth direction, then the pyroelectric and piezoelectric fields along the [10-10] direction are zero. In this case, the quantum well (i.e. the conduction and valence band profile) is symmetric.

Figure 3.20.32.4 shows the electron and hole wave functions ( $\psi^2$ ) of the (10-10)-oriented GaN/AlGaN QW for  $k_{||} = 0$ . Obviously, the interband transition matrix elements (i.e. the probability for electron-hole transitions) are much larger than for the [0001] growth direction.

In agreement with [Park2000], we calculated the electron levels within the single-band effective mass approximation and the hole levels within the 6-band k.p approximation.

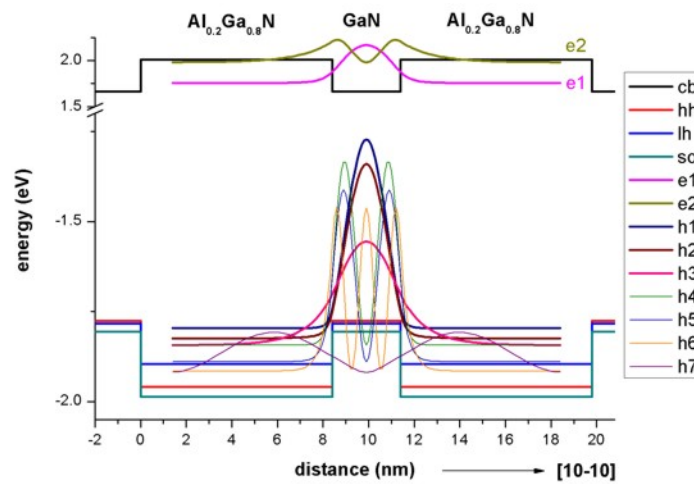


Figure 3.20.32.4: Calculated wave functions of lowest eigenstates.

$k_{||}$  dispersion: Calculation of the electron and hole energies and wave functions for  $k_{||} \neq 0$ .

Input file: `1DGaN_AlGaN_QW_k_parallel_10m10.nn3`

Due to the symmetry of the quantum well, we expect degenerate eigenvalues for the in-plane dispersion relation (Kramer's degeneracy). Our results, depicted in Figure 3.20.32.5, compare well with Fig. 3(c) of [Park2000].

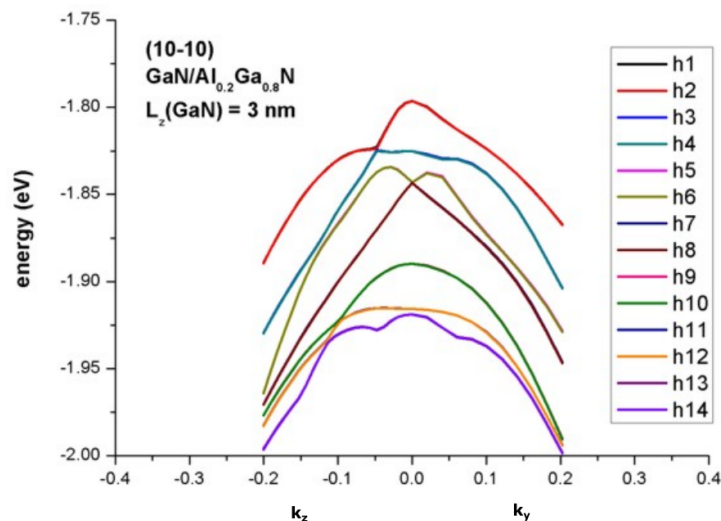


Figure 3.20.32.5: Calculated energy dispersion  $E(k_{||}) = E(k_y, k_z)$ .

Last update: 2025-11-04

### 3.20.33 T-shaped quantum wire grown by cleaved edge overgrowth (CEO): wave functions without strain

#### **Note**

The tutorial is related to the PhD Thesis of R. Schuster [*SchusterPhD2005*]

- *Header*
- *Structure*
- *Input file*
- *Results*
  - *Effective mass approximation*
  - *6-band  $k.p$  approximation*
  - *Eigenenergies*
- *Including anisotropic effects in the effective mass model*

#### **Header**

##### **Input files:**

- `examples\quantum_wires\T-QWR_zb_III-V_Schuster_PhD_2005_2D_nnp.in`

##### **Scope:**

Electron and hole wave functions of a T-shaped quantum wire (QWR).

##### **Output files:**

- `\bias_xxxx\Quantum\probabilities_quantum_region_Gamma.fld`
- `\bias_xxxx\Quantum\probabilities_quantum_region_HH.fld`

- `\bias_xxxx\Quantum\probabilities_quantum_region_LH.fld`
- `\bias_xxxx\Quantum\probabilities_quantum_region_kp6_00000.fld`

## Structure

Similar to the 1D confinement in a quantum well, it is possible to confine electrons or holes in two dimensions, i.e. in a quantum wire. In this tutorial we consider the quantum wire, which is formed at the T-shaped intersection of two 10 nm GaAs type-I quantum wells, surrounded by  $\text{Al}_{0.35}\text{Ga}_{0.65}\text{As}$  barriers (see Figure 3.20.33.1). The electrons and holes are free to move along the  $z$  direction only, thus the wire is oriented along the  $[0-11]$  direction. Such a heterostructure can be manufactured by growing the layers along two different growth directions with the CEO (cleaved edge overgrowth) technique. Due to the nearly identical lattice constants of GaAs and AlAs it is possible to assume this heterostructure as being unstrained.

The wave function is indicated at the T-shaped intersection in yellow. Here, the wave function can extend into a larger volume (as compared to the quantum well) and thus reduce its energy. Quantum mechanics tells us that the ground state can be found at this intersection and electrons are only allowed to move one-dimensionally along the  $z$  direction. Figure 3.20.33.1 b) shows a 60 nm x 60 nm extract of the schematic layout including the dimensions, the material composition and the orientation of the wire with respect to the crystal coordinate system.

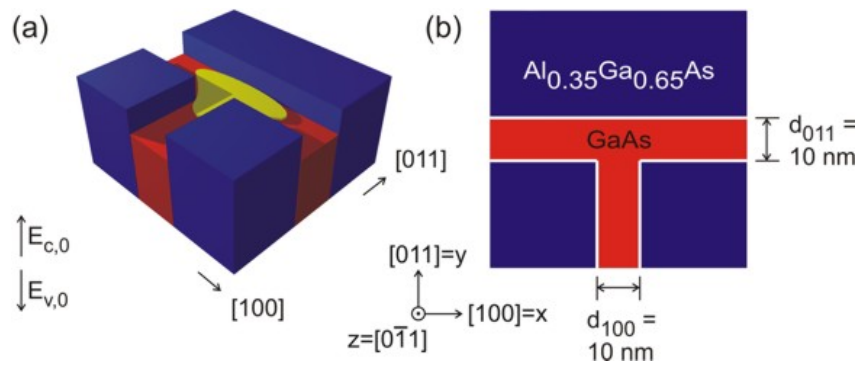


Figure 3.20.33.1: Two-dimensional conduction band edges of the T-shaped quantum wire.

## Input file

It is sufficient to describe this heterostructure within a 2D simulation as it is translationally invariant along the  $z$  direction. The simulation coordinate system is oriented in the following way:

```
global{
  simulate2D{
    crystal_zb{
      x_hkl = [1, 0, 0]
      y_hkl = [0, 1, 1]
    }
  }
}
```

As we do not have doping and no piezoelectric fields (the structure is assumed to be unstrained) and as the temperature is assumed to be 4 K, we do not have to deal with charge redistributions. Thus, we can refrain from solving Poisson's equation, and we also do not have to take care about self-consistency.

Material parameters of relevance are the conduction band and valence band offset between GaAs and  $\text{Al}_{0.35}\text{Ga}_{0.65}\text{As}$ :

$$\begin{aligned}
 \text{CBO} &= 0.2847 \text{ eV} \\
 \text{VBO} &= -0.1926 \text{ eV} \\
 E_{\text{gap}, \text{Al}_{0.35}\text{Ga}_{0.65}\text{As}} &= 2.2883 \text{ eV} \\
 E_{\text{gap}, \text{GaAs}} &= 1.5193 \text{ eV}
 \end{aligned}$$

## Results

Using the input file *T-QWR\_GaAs-AlGaAs\_Schuster\_2005\_2D\_nnp.in* we calculate the electron, heavy hole and light hole wavefunctions for the T-shaped quantum wire structure.

### Effective mass approximation

The electron and hole wave functions can be calculated within the effective mass theory (envelope function approximation) by using position dependent effective masses. In our example, the effective masses are constant within each material but have discontinuities at the material interfaces. In *nextnano++* the effective masses are assumed to be isotropic. Both, the heavy hole and the light hole band edge energies are degenerate but the effective masses differ. Thus, we have to solve three Schrödinger equations, namely for the conduction band, heavy hole band and light hole band. To trigger the 1-band effective mass model for calculating the eigenstates, use the following setting in the input file *T-QWR\_GaAs-AlGaAs\_Schuster\_2005\_2D\_nnp.in*:

```
$kp6 = 0      # choose 1 (6 band k.p) or 0 (effective mass approximation)
↪(ListOfValues: 1,0)
```

In Figure 3.20.33.2 we show the normalized probability densities ( $\psi^2$ ) for the electron, heavy hole and light hole ground states, which are obtained by the effective mass approximation.

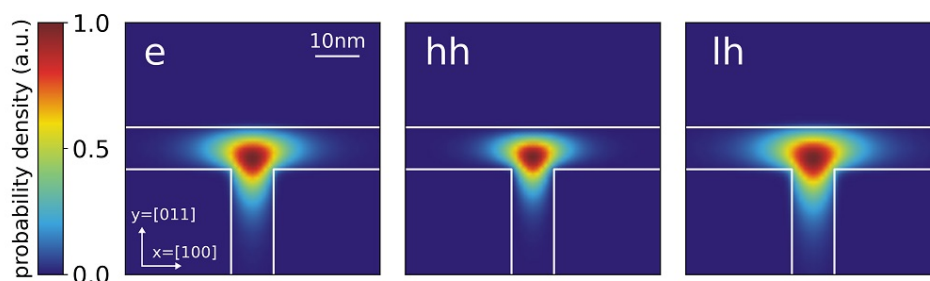


Figure 3.20.33.2: Probability densities of the electron (e), heavy hole (hh) and light hole (lh) state calculated using the effective mass approximation. The wavefunctions are normalized so that the maxima are equal to one.

In addition to these ground states for  $k_z = 0$ , excited states are possible as well. Similar to the subbands of a 1D quantum well that show a  $E(k_x, k_y)$  dispersion one can assign a subband with the energy dispersion  $E(k_z)$  to each quantum wire eigenvalue which describes the free motion along the quantum wire axis ( $z$  axis). A more advanced treatment would be to use k.p theory to calculate the eigenvalues for different  $k_z$  in order to obtain the (nonparabolic) energy dispersion  $E(k_z)$ .

### 6-band k.p approximation

For the same structure as above we perform the calculations again, but this time using the 6-band k.p model instead of the single-band effective mass approximation. To trigger the 6-band k.p model for calculating the eigenstates, the following setting in the input file *T-QWR\_GaAs-AlGaAs\_Schuster\_2005\_2D\_nnp.in* can be used:

```
$kp6 = 1      # choose 1 (6 band k.p) or 0 (effective mass approximation)
↪(ListOfValues: 1,0)
```

Figure 3.20.33.3 shows the probability density ( $\psi^2$ ) for the hole ground state. For the results shown on the left we used the following Luttinger parameters for GaAs:  $\gamma_1 = 6.98$ ,  $\gamma_2 = 2.06$ ,  $\gamma_3 = 2.93$ , which corresponds to:  $L = -16.220$ ,  $M = -3.860$ ,  $N = -17.580$ . For the results shown on the right, we modified the Luttinger parameters for GaAs to  $\gamma_1 = 6.98$ ,  $\gamma_2 = 2.06 = \gamma_3$ , which corresponds to  $L = -16.220$ ,  $M = -3.860$ ,  $N = -12.36$ . Choosing  $\gamma_2 = \gamma_3$  corresponds to an isotropic effective mass.

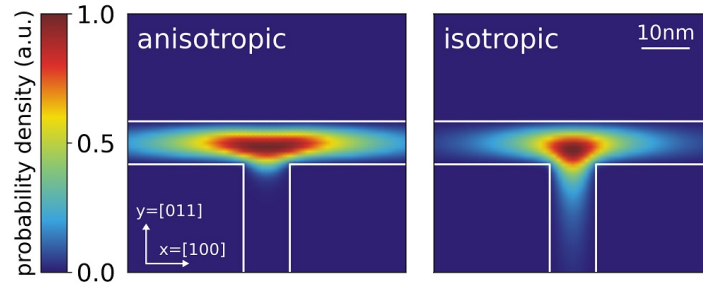


Figure 3.20.33.3: Probability density ( $\psi^2$ ) for the hole ground state using anisotropic and isotropic k.p parameters.

## Eigenenergies

The calculated eigenvalues for the ground states are:

| effective-mass       |                |                | 6-band k.p             |
|----------------------|----------------|----------------|------------------------|
| electron energy (eV) | hh energy (eV) | lh energy (eV) | hole state energy (eV) |
| 3.006                | 1.455          | 1.437          | 1.455                  |

## Including anisotropic effects in the effective mass model

Compared to *nextnano++*, *nextnano<sup>3</sup>* allows using anisotropic effective masses for solving the Schrödinger equation within the effective mass approximation. The effective mass  $m^*$  depends now on the chosen direction, which is described by a tensor. The components of the effective mass tensor, which are mass along the crystal coordinate axes, can be derived from the 6-band k.p parameters (or Luttinger parameters). Using the Luttinger parameters  $\gamma_1$ ,  $\gamma_2$  and  $\gamma_3$ , the effective masses for heavy and light holes along [110] and [010] in units of  $m_0$  can be calculated as follows:

$$m_{hh,[100]}^* = \frac{1}{\gamma_1 - 2\gamma_2},$$

$$m_{hh,[011]}^* = \frac{1}{\gamma_1 - 0.5 \cdot (\gamma_2 + 3\gamma_3)},$$

$$m_{lh,[100]}^* = \frac{1}{\gamma_1 + 2\gamma_2},$$

$$m_{lh,[011]}^* = \frac{1}{\gamma_1 + 0.5 \cdot (\gamma_2 + 3\gamma_3)}.$$

The Luttinger parameters for GaAs are given by:  $\gamma_1 = 6.98$ ,  $\gamma_2 = 2.06$  and  $\gamma_3 = 2.93$ . The relations between the Luttinger parameters and the isotropic effective masses are

$$m_{hh,isotropic}^* = \frac{1}{\gamma_1 - 0.8\gamma_2 - 1.2\gamma_3},$$

$$m_{lh,isotropic}^* = \frac{1}{\gamma_1 + 0.8\gamma_2 + 1.2\gamma_3}.$$

Usually the database entries for the effective masses assume spherical symmetry for the holes and are specified with respect to the crystal coordinate system. Their default values (isotropic) and the values which were derived from the Luttinger parameters are given in this table:

|                                      | heavy hole (GaAs) | light hole (GaAs) |
|--------------------------------------|-------------------|-------------------|
| along [100] direction                | 0.350             | 0.090             |
| along [011] direction                | 0.643             | 0.081             |
| isotropic                            | 0.551             | 0.082             |
| <i>nextnano<sup>3</sup></i> database | 0.500             | 0.068             |



In this tutorial, however, we calculated the effective masses for different directions and, therefore, we do not have spherical symmetry anymore. Thus, we have to rotate the new eigenvalues of the effective mass tensors that are given in the  $x = [100]$ ,  $y = [011]$ ,  $z = [0-11]$  simulation coordinate system into the crystal coordinate system where  $x_{cr} = [100]$ ,  $y_{cr} = [010]$ ,  $z_{cr} = [001]$ . First, we have to overwrite the default entries in the database so that they contain the eigenvalues of the effective mass tensors in the simulation system:

```
valence-band-masses = 0.35d0 0.643d0 0.643d0 ! eigenvalues of the heavy hole
↪ effective mass tensor [100] [011] [0-11]
                    0.090d0 0.081d0 0.081d0 ! eigenvalues of the light
↪ hole effective mass tensor [100] [011] [0-11]
```

To project these eigenvalues onto the crystal coordinate system we need to know the principal axis system which these eigenvalues refer to (The normalization of these vectors will be done internally by the program):

```
principal-axes-vb-masses = 1d0 0d0 0d0 ! heavy hole [100]
                        0d0 1d0 1d0 ! [011]
                        0d0 -1d0 1d0 ! [0-11]

                        1d0 0d0 0d0 ! light hole [100]
                        0d0 1d0 1d0 ! [011]
                        0d0 -1d0 1d0 ! [0-11]
```

Figure 3.20.33.4 and Figure 3.20.33.5 show the probability densities ( $\psi^2$ ) of the ground states of the confined electron, heavy and light hole eigenstates of the quantum wire. The lowest hole state is the heavy hole state and the second hole state is the light hole state. No further hole states are confined. Also, in the conduction band only the ground state is confined. One can clearly see that each ground state wave function is localized at the T-shaped intersection and shows the T-shaped symmetry. Due to the anisotropy of the heavy hole effective mass, the heavy hole wave function prefers to extend along the  $[100]$  direction and hardly penetrates into the quantum well that is aligned along the  $[011]$  direction. The heavy hole mass along the  $[100]$  direction is only half the value as along the  $[011]$  direction. The light hole anisotropy is only minor and thus its symmetry resembles the one of the isotropic electron.

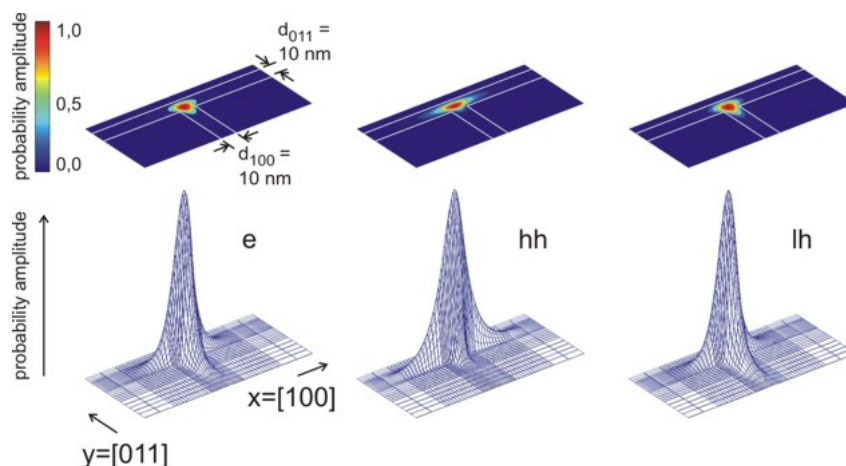


Figure 3.20.33.4: Probability amplitudes of the electron (e), heavy hole (hh) and light hole (lh) envelope functions at an unstrained T-shaped intersection of two 10 nm wide GaAs quantum wells embedded by  $\text{Al}_{0.35}\text{Ga}_{0.65}\text{As}$  barriers. The wave functions are normalized so that the maxima are equal to one.

These results are in very good qualitative agreement with the heavy hole and light hole wave functions calculated within the 6-band k.p calculation. This demonstrates the impact of an isotropic (for electrons and light holes) or anisotropic (for heavy hole) effective mass on the obtained wavefunctions.

#### Acknowledgement:

We would like to thank Robert Schuster from the University of Regensburg for providing experimental data and some figures for this tutorial.



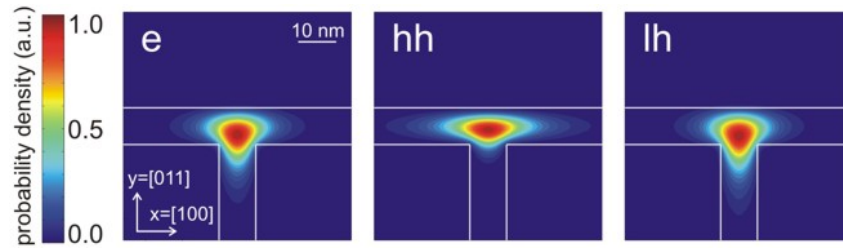


Figure 3.20.33.5: Contour diagram of the probability amplitudes of the electron (e), heavy hole (hh) and light hole (lh) eigenfunctions (same figures as Figure 3.20.33.4, but this time viewed from the top). The wave functions are normalized so that the maxima are equal to one.

*Last update: 27/05/2025*



## MATERIAL DATABASE

### 4.1 The database keywords

Here you can find detailed descriptions about the keywords and specifiers in the default parameter database input file `database_nn3.in` and the general definitions file `database_nn3_keywords.val`.

The scheme is

- `keywords.val` (general definitions) `<== input_file.in` (input file)
- `database_nn3_keywords.val` (general definitions) `<== database_nn3.in` (database file)

Valid keywords for the parameters of the default parameter database are listed in `database_nn3_keywords.val`. This is the place to enter new keywords if one wants to add new parameters. In principle, one should only enter into this database keywords, which are marked as **required**. Through this it is warranted that corresponding entries in the database actually do occur. The input parser checks this, but only their existence. Additionally, the new keywords have to be entered into `keywords.val`. E.g. for zinc blende materials the keyword is `$binary_zb_default`. On this basis, the entries in the data base `database_nn3.in` have to be complemented for the newly declared values for all affected known materials. This also includes the adaption of changes in the data types of already existing parameters (e.g. `double ==> double_array`).

Material parameters can be found in the review articles of Vurgaftman and Meyer ([VurgaftmanJAP2001], [VurgaftmanJAP2003]).

### 4.2 Keywords

#### 4.2.1 \$binary-zb-default

The documentation for this keyword is available here: [https://www.nextnano.com/nextnano3/input\\_parser/database/binary-zb-default.htm](https://www.nextnano.com/nextnano3/input_parser/database/binary-zb-default.htm)

#### 4.2.2 \$binary-wz-default

The documentation for this keyword is available here: [https://www.nextnano.com/nextnano3/input\\_parser/database/binary-wz-default.htm](https://www.nextnano.com/nextnano3/input_parser/database/binary-wz-default.htm)

#### 4.2.3 \$ternary-zb-default

Ternary zinc blende parameters

Parameters for zinc blende type ternary alloys. This set of parameters refers to the binary constituents and their material parameters. Here, the bowing parameters for interpolation between these binaries are specified.

A bowing parameter  $b$  is defined as follows for the material parameter  $Q$ . Note that there is a minus sign in front of the term  $bx(1-x)$ .

$$Q[A_xB_{1-x}C] = x \cdot Q[AC] + (1-x) \cdot Q[BC] - b \cdot x \cdot (1-x)$$

$b$  is defined as  $b = 4Q(A_{0.5}B_{0.5}C) - 2(Q[AC] + Q[BC])$ .

The advantage of the bowing model is that it requires knowledge of the relevant quantity only at a composition  $x=0.5$  together with the values for the binaries.

Please check the keywords section for more details: \$ternary-zb-default

For the meaning of the material parameters and its units, please check the keywords section for \$binary-zb-default for more details. The units for binary material parameters and for the bowing parameters are the same.

|                                       |              |                 |
|---------------------------------------|--------------|-----------------|
| <b>\$ternary-zb-default</b>           |              | <b>required</b> |
| ternary-type                          | character    | required        |
| binary(x)                             | character    | required        |
| binary(1-x)                           | character    | required        |
| bow-conduction-band-masses            | double_array | optional        |
| bow-conduction-band-nonparabolicities | double_array | optional        |
| bow-conduction-band-energies          | double_array | optional        |
| bow-valence-band-masses               | double_array | optional        |
| bow-valence-band-nonparabolicities    | double_array | optional        |
| bow-valence-band-energies             | double_array | optional        |
| bow-band-gaps                         | double_array | optional        |
| bow-static-dielectric-constants       | double_array | optional        |
| bow-optical-dielectric-constants      | double       | optional        |
| bow-lattice-constants                 | double_array | optional        |
| bow-elastic-constants                 | double_array | optional        |
| bow-piezo-electric-constants          | double_array | optional        |
| bow-abs-deformation-pot-vb            | double       | optional        |
| bow-abs-deformation-pots-cbs          | double_array | optional        |
| bow-uniax-vb-deformation-pots         | double_array | optional        |
| bow-uniax-cb-deformation-pots         | double_array | optional        |
| bow-Luttinger-parameters              | double_array | optional        |
| bow-6x6kp-parameters                  | double_array | optional        |
| bow-8x8kp-parameters                  | double_array | optional        |
| bow-LO-phonon-energy                  | double       | optional        |
| bow-band-shift                        | double       | optional        |
| band-shift                            | double       | optional        |
| <b>\$end_ternary-zb-default</b>       |              | <b>required</b> |

#### Example 1

If no bowing parameters are specified, linear interpolation is done.

```

!-----!
$ternary-zb-default
ternary-type           = Al(x)In(1-x)Sb-zb-default
binary(x)              = AlSb-zb-default
binary(1-x)            = InSb-zb-default
$end_ternary-zb-default
!-----!

```

#### Example 2

If a bowing parameter is nonzero, bowing is used. Bowing can be quite complicated, especially if

- valence band offset (average valence band edge energy)
- spin-orbit splitting energy
- band gap

are bowed simultaneously.

Note that you can use `band-shift` to shift the resulting band edges by the desired amount.

It is hard to find bowing parameters for a default material database that works for every alloy parameter.

```

!-----!
$ternary-zb-default
ternary-type           = In(x)Ga(1-x)As-zb-default
binary(x)              = InAs-zb-default
binary(1-x)            = GaAs-zb-default

bow-conduction-band-masses = 0.0091    0.0091    0.0091    !
→=> 0.04300 [m0] (for In0.53Ga0.47As)
                        0.0      0.0      0.0      !
→no bowing for L
                        0.0      0.0      0.0      !
→no bowing for X

bow-band-gaps           = 0.477      0.33      1.4      !
→[Vurgaftman1]
bow-conduction-band-energies = 0.477      0.33      1.4      !
→[Vurgaftman1] for gaps. This is good (better than zero bowing).

bow-valence-band-masses = -0.145     -0.145     -0.145     !
→hh along [001]
                        0.0202    0.0202    0.0202    !
→lh along [001]
                        0.0      0.0      0.0      !
→so

bow-valence-band-energies = -0.38      !
→[Vurgaftman1] (valence band offset bowing) (Does this value refer to the
→valence band edge or to the average valence band edge?)
!bow-valence-band-energies = -0.0511107 !
→to get a band gap of 0.816 eV (for In0.53Ga0.47As)

bow-abs-deformation-pots-cbs = 2.61      2.61      2.61      !
→[Vurgaftman1] (Gamma) absolute deformation potentials of conduction band
→minima (Gamma, L, X)

bow-Luttinger-parameters = 11.52388599  5.191489362  5.67282216 !
→[Vurgaftman1] to get original values of Alavi et al. for In0.53Ga0.47As,
→recommended by Vurgaftman
!bow-Luttinger-parameters = 0.0      0.0      0.0      !
→gamma3 - gamma2 = 0.481 [Vurgaftman1]
                        0.0      0.0      !
bow-6x6kp-parameters     = -32.28984344 -1.140907266 -34.03693296 !
→L = -28.73, M = -3.65, N = -29.04 (for In0.53Ga0.47As)
                        0.15      !
→Delta_so [Vurgaftman1]
bow-8x8kp-parameters     = 0.0      -1.140907266  0.0      !
                        0.0 -1.48      3.54      !
→bowing(E_P) = -1.48 eV ([Vurgaftman1] => E_P = 25.3 eV), bowing(S) = 2 *
→bowing(F) = 2 * 1.77 ([Vurgaftman1])
!                        0.0 12.1678040 3.54      !
→bowing(E_P) = 12.1678040 eV (Sirtori => E_P = 21.9 eV = 21.5 * 0.53 + 28.
→8 * 0.47 - 0.53 * 0.47 * 12.16780409)

$end_ternary-zb-default
!-----!

```

Example 3

You can define special ternaries for certain conditions, e.g. low temperature or high In content.

```
!-----!
$ternary-zb-default
  ternary-type           = Al(x)In(1-x)Sb-4K-zb-default
!ternary-type           = Al(x)In(1-x)Sb-zb-default
  binary(x)              = AlSb-zb-default
  binary(1-x)            = InSb-zb-default

! Define here special bowing parameters for 4 K.
...

$end_ternary-zb-default
!-----!
```

ternary-type e.g. Al(x)Ga(1-x)As-zb-default, must be a defined ternary material. This string is usually a known material. If a material-type with material-model = ternary-zb-default is specified within the \$material keyword which is unknown, you have to provide a complete set of input data for this material type. In this case the material-type must be equal to string. However, the binary constituents can still be either known or unknown binary materials.

binary(x) and binary(1-x) must be a binary material of type binary-zb-default, e.g. AlSb-zb-default, must be a defined binary material. This string can be either a known binary or an arbitrary name. In case this binary is not a known material, you will be prompted for all material parameters.

#### band-shift

```
type
  double

presence
  optional

value
  0.0
```

If nonzero, the resulting ternary material is shifted by this amount (independent of the alloy-content). So the default value should be 0.0.

#### bow-valence-band-energies

```
type
  double

presence
  optional

value
  0.0
```

This bowing applies to the average valence band edge energy and not to the valence band maximum.

### 4.2.4 \$ternary-wz-default

Ternary wurtzite parameters

Parameters for wurtzite type ternary alloys. This set of parameters refers to the binary constituents and their material parameters. Here, the bowing parameters for interpolation between these binaries are specified.

|                             |                     |                 |
|-----------------------------|---------------------|-----------------|
| <b>\$ternary-wz-default</b> |                     | <b>required</b> |
| ternary-type                | <b>character</b>    | <b>required</b> |
| binary(x)                   | <b>character</b>    | <b>required</b> |
| binary(1-x)                 | <b>character</b>    | <b>required</b> |
| bow-conduction-band-masses  | <b>double_array</b> | <b>optional</b> |

(continues on next page)

(continued from previous page)

|                                                                 |              |                |
|-----------------------------------------------------------------|--------------|----------------|
| bow-conduction-band-nonparabolicities                           | double_array | optional       |
| bow-conduction-band-energies                                    | double_array | optional       |
| bow-valence-band-masses                                         | double_array | optional       |
| bow-valence-band-nonparabolicities                              | double_array | optional       |
| bow-valence-band-energies                                       | double_array | optional !     |
| → "average" valence band edge energy $E_v$ (without splittings) |              |                |
| bow-band-gaps                                                   | double_array | optional       |
| bow-static-dielectric-constants                                 | double_array | optional       |
| bow-optical-dielectric-constants                                | double_array | optional ! for |
| → zinblend, this is double                                      |              |                |
| bow-lattice-constants                                           | double_array | optional       |
| bow-piezo-electric-constants                                    | double_array | optional       |
| bow-pyro-polarization                                           | double_array | optional ! for |
| → zinblend, this does not exist                                 |              |                |
| bow-abs-deformation-pot-vb                                      | double       | optional ! not |
| → used in wurtzite                                              |              |                |
| bow-abs-deformation-pots-cbs                                    | double_array | optional       |
| bow-uniax-vb-deformation-pots                                   | double_array | optional       |
| bow-uniax-cb-deformation-pots                                   | double_array | optional ! not |
| → used in wurtzite                                              |              |                |
| bow-elastic-constants                                           | double_array | optional       |
| bow-Luttinger-parameters                                        | double_array | optional       |
| bow-6x6kp-parameters                                            | double_array | optional       |
| bow-8x8kp-parameters                                            | double_array | optional       |
| bow-LO-phonon-energy                                            | double_array | optional ! for |
| → zinblend, this is double                                      |              |                |
| bow-band-shift                                                  | double       | optional       |
| band-shift                                                      | double       | optional       |
| \$end_ternary-wz-default                                        |              | required       |

Here, no further explanations are given. The meaning is analogous to *\$ternary-zb-default*.

## 4.2.5 \$default-materials

Default materials.

|                                |           |                 |
|--------------------------------|-----------|-----------------|
| <b>\$default-materials</b>     |           | <b>required</b> |
| material-name                  | character | required        |
| material-type                  | character | required        |
| material-model                 | character | required        |
| material-property              | character | optional        |
| <b>\$end_default-materials</b> |           | <b>required</b> |

Explanation of specifiers.

### material-name

#### type

character

#### presence

required

#### value

e.g. Si

An arbitrary name can be chosen, e.g. Si or silicon.

### material-type

**type**  
character

**presence**  
required

**value**  
e.g. Si-zb-default

The material parameters of Si-zb-default have to be defined in the database.

#### material-model

**type**  
character

**presence**  
required

**value**  
binary-zb-default, binary-wz-default, ternary-zb-default,  
ternary-wz-default; not supported yet: quaternary-wz-default

Zinc blende (zb) and wurtzite (wz) binary and ternary alloys are supported. Materials with diamond crystal structure can be specified using zinc blende. SiGe alloys are called “ternaries” even if only two atoms are involved.

For alloys (ternaries) only.

#### material-property

**type**  
character

**presence**  
optional

**value**  
swap-(x)-and-(1-x)

If material is a ternary material, this value can be set to swap-(x)-and-(1-x). This allows to specify 2 different types of SiGe alloys that contain the same material parameters but  $x$  and  $y = (1 - x)$  are interchanged.

```
material-name = Si(1-x)Ge(x)    material-type = Si(1-x)Ge(x)-zb-default  ↵
↪material-model = ternary-zb-default
material-name = Si(x)Ge(1-x)    material-type = Si(1-x)Ge(x)-zb-default  ↵
↪material-model = ternary-zb-default  material-property = swap-(x)-and-(1-
↪x)
```

Here, material-name is swapped but both refer to the same material-type.

You can also define your own materials, e.g. GaAs with parameters for 4 K, or explicit parameters for a ternary, e.g. Al(x)In(1-x)P with a fixed value of  $x=0.52$ : Al<sub>0.52</sub>In<sub>0.48</sub>P

```
material-name = GaAs-4K
material-model = binary-zb-default
material-type = GaAs-4K-zb-default

material-name = Al0.52In0.48P
material-model = binary-zb-default
material-type = Al0.52In0.48P-zb-default
```

Further examples

```
!-----!
$default-materials
```

(continues on next page)



(continued from previous page)

```

material-name = Si
material-type = Si-zb-default
material-model = binary-zb-default

!-----
! 'alias' materials: Here, we specify 4 different types of AlGaAs alloy
!-----
!-----
material-name = Al(x)Ga(1-x)As    material-type = Al(x)Ga(1-x)As-zb-
!-----
!-----
material-name = Ga(1-x)Al(x)As    material-type = Al(x)Ga(1-x)As-zb-
!-----
!-----
material-name = Al(1-x)Ga(x)As    material-type = Al(x)Ga(1-x)As-zb-
!-----
!-----
material-name = Ga(x)Al(1-x)As    material-type = Al(x)Ga(1-x)As-zb-
!-----
!-----
material-name = Al(x)Ga(1-x)N
material-type = Al(x)Ga(1-x)N-wz-default
material-model = ternary-wz-default

$send_default-materials
!-----

```

## 4.2.6 \$default-material-models

Default material models.

|                                       |                  |                 |
|---------------------------------------|------------------|-----------------|
| <b>\$default-material-models</b>      |                  | <b>required</b> |
| material-model-number                 | <b>integer</b>   | <b>required</b> |
| material-model                        | <b>character</b> | <b>required</b> |
| needs-alloy-function                  | <b>logical</b>   | <b>required</b> |
| <b>\$send_default-material-models</b> |                  | <b>required</b> |

Zinc blende and wurtzite materials are supported. Binary and ternaries are possible. The latter need an alloy function. Quaternaries are not supported yet.

Example

```

!-----
$default-material-models

! Zinc blende

material-model-number = 1
material-model        = binary-zb-default
needs-alloy-function  = .FALSE.

material-model-number = 2
material-model        = ternary-zb-default
needs-alloy-function  = .TRUE.

```

(continues on next page)

(continued from previous page)

```

! Wurtzite

material-model-number = 3
material-model        = binary-wz-default
needs-alloy-function  = .FALSE.

material-model-number = 4
material-model        = ternary-wz-default
needs-alloy-function  = .TRUE.

! not implemented yet:

material-model-number = 5                                ! not supported yet
material-model        = quaternary-zb-default           ! not supported yet
needs-alloy-function  = .TRUE.                          ! not supported yet

$send_default-material-models
!-----!

```

### 4.2.7 \$Auger-recombination

More information on the Physics: [Auger recombination processes in semiconductor heterostructures](#)

For devices with an extremely high carrier concentration the Auger process is the dominant recombination channel. The process involves three particles and therefore scales with the third power of the carrier densities. The phonon-assisted Auger recombination rate, which plays an important role especially at high carrier injection, respectively high doping levels, will be modeled in the program by the following equation:

$$R_{\text{Aug}} = (C_n n + C_p p)(np - n_i^2)$$

where  $n$  is the electron density,  $p$  is the hole density and  $n_i$  is the intrinsic density.

|                                   |                  |                 |
|-----------------------------------|------------------|-----------------|
| <b>\$Auger-recombination</b>      |                  | <b>optional</b> |
| material-name                     | <b>character</b> | <b>required</b> |
| number-of-parameters              | <b>integer</b>   | <b>required</b> |
| n-C                               | <b>double</b>    | <b>optional</b> |
| p-C                               | <b>double</b>    | <b>optional</b> |
| n-bow-C                           | <b>double</b>    | <b>optional</b> |
| p-bow-C                           | <b>double</b>    | <b>optional</b> |
| <b>\$send_Auger-recombination</b> |                  | <b>optional</b> |

Explanation of specifiers.

#### material-name

##### type

character

##### presence

required

##### value

e.g. GaAs

Name of material to which this set of parameters applies. Name has to be listed in *\$default-materials*.

#### number-of-parameters

##### type

integer

**presence**  
required

**value**  
e.g. 2

Control parameter if the number of parameters provided is the same as demanded.

There are two sets of parameters, one for electrons (n) and one for holes (p).

#### n-C

**type**  
double

**presence**  
required

**value**  
e.g.  $1.0 \times 10^{-30}$

**unit**  
(cm<sup>6</sup>/s)

The  $C_n$  parameter for electrons as specified in the equation above. The order of magnitude is around  $10^{-30}$  (cm<sup>6</sup>/s).

#### n-bow-C

**type**  
double

**presence**  
optional

**value**  
e.g. 0.0

**unit**  
(cm<sup>6</sup>/s)

For ternary alloys there are also bowing parameters possible. n-bow-C = 0.0 means zero bowing, i.e. linear interpolation is used.

#### p-C

**unit**  
(cm<sup>6</sup>/s)

Same as n-C but for holes.

#### p-bow-C

**unit**  
(cm<sup>6</sup>/s)

Same as n-bow-C but for holes.

Example

```
!-----!
$Auger-recombination      !
                             !
material-name              = Si      !
number-of-parameters      = 2       !
n-C                        = 2.8e-31  ! [cm^6/s]
p-C                        = 9.9e-31  ! [cm^6/s]
                             !
```

(continues on next page)

(continued from previous page)

```

material-name      = GaAs                !
number-of-parameters = 2                !
n-C                = 1.0e-30             ! [cm^6/s]
p-C                = 1.0e-30             ! [cm^6/s]
!
material-name      = Al(x)Ga(1-x)As     !
number-of-parameters = 2                !
n-bow-C            = 0.0                ! [cm^6/s]
p-bow-C            = 0.0                ! [cm^6/s]
!
$end_Auger-recombination                !
!-----!

```

There is also a keywords section in the input file for *\$Auger-recombination* which you can use to overwrite default material parameters.

#### 4.2.8 \$direct-recombination

The simplest process for the generation and recombination of electron-hole pairs is the direct process via the emission or absorption of a photon (radiative recombination). This is important for light emitting devices. The local spontaneous emission rate is approximated as

$$R_{dir} = C(np - n_i^2)$$

where  $C$  is called the *bimolecular recombination coefficient* (often abbreviated with the letter  $B$ ),  $n$  is the electron density,  $p$  is the hole density and  $n_i$  is the intrinsic density.

This simple equation includes the full spectrum of photons generated by spontaneous band-to-band transition processes. By definition, the radiative recombination rate (= spontaneous electron recombination rate) per unit volume of an electron-hole pair is equivalent to the spontaneous photon generation rate per unit volume, because obviously, each time an electron recombines with a hole radiatively, a photon is emitted.

|                                   |                  |                 |
|-----------------------------------|------------------|-----------------|
| <b>\$direct-recombination</b>     |                  | <b>optional</b> |
| material-name                     | <b>character</b> | <b>required</b> |
| number-of-parameters              | <b>integer</b>   | <b>required</b> |
| C-opt                             | <b>double</b>    | <b>optional</b> |
| bow-C-opt                         | <b>double</b>    | <b>optional</b> |
| <b>\$end_direct-recombination</b> |                  | <b>optional</b> |

Explanation of specifiers.

##### material-name

###### type

character

###### presence

required

###### value

e.g. GaAs

Name of material to which this set of parameters applies. Name has to be listed in *\$default-materials*.

##### number-of-parameters

###### type

integer

###### presence

required

**value**

e.g. 1

Control parameter if the number of parameters provided is the same as demanded.

**C-opt****type**

double

**presence**

required

**value**e.g.  $7.2e-10$ **unit** $(\text{cm}^3/\text{s})$ 

The parameters are specified as shown in the tables above. The order of magnitude is around  $10^{-10} (\text{cm}^3/\text{s})$ .

**bow-C-opt****type**

double

**presence**

optional

**value**

e.g. 0.0

**unit** $(\text{cm}^3/\text{s})$ 

For ternary alloys there are also bowing parameters possible. `bow-C-opt = 0.0` means zero bowing, i.e. linear interpolation is used.

Example

```
!-----!
$direct-recombination      !
!                           !
material-name              = GaAs      !
number-of-parameters      = 1          !
C-opt                      = 7.2e-10    ! [cm^3/s]
!                           !
material-name              = GaN        !
number-of-parameters      = 1          !
C-opt                      = 2e-10      ! [cm^3/s]
!                           !
material-name              = Al(x)Ga(1-x)As !
number-of-parameters      = 1          !
bow-C-opt                  = 0.0        ! [cm^3/s]
!                           !
$end_direct-recombination  !
!-----!
```

There is also a keywords section in the input file for *\$direct-recombination* which you can use to overwrite default material parameters.

### 4.2.9 \$SRH-recombination

The generation/recombination process can be assisted by impurities. This is modeled by the Shockley-Read-Hall model (SRH). The recombination/generation rates depend on the deviation of the carrier concentration from the equilibrium value and the scattering rates depend on the doping concentration.

$$R_{\text{SRH}} = \frac{pn - n_i^2}{\tau_p(n + n_i) + \tau_n(p + n_i)}$$

where

$$\tau_{p,n}(N_D + N_A) = \frac{\tau_{p0,n0}}{1 + \frac{N_D + N_A}{N_{p,n,\text{ref}}}}$$

and  $n$  is the electron density,  $p$  is the hole density and  $n_i$  is the intrinsic density.

$N_D$  and  $N_A$  are the donor and acceptor concentrations, respectively.

$N_{n,\text{ref}}$  and  $N_{p,\text{ref}}$  are the reference doping concentrations for electrons and holes, respectively.

$\tau_{n0}$  and  $\tau_{p0}$  are the zero doping scattering times for electrons and holes, respectively.

|                                |                  |                 |
|--------------------------------|------------------|-----------------|
| <b>\$SRH-recombination</b>     |                  | <b>optional</b> |
| material-name                  | <b>character</b> | <b>required</b> |
| number-of-parameters           | <b>integer</b>   | <b>required</b> |
| n-N-ref                        | <b>double</b>    | <b>optional</b> |
| n-tau                          | <b>double</b>    | <b>optional</b> |
| p-N-ref                        | <b>double</b>    | <b>optional</b> |
| p-tau                          | <b>double</b>    | <b>optional</b> |
| n-bow-N-ref                    | <b>double</b>    | <b>optional</b> |
| n-bow-tau                      | <b>double</b>    | <b>optional</b> |
| p-bow-N-ref                    | <b>double</b>    | <b>optional</b> |
| p-bow-tau                      | <b>double</b>    | <b>optional</b> |
| <b>\$end_SRH-recombination</b> |                  | <b>optional</b> |

Explanation of specifiers.

#### material-name

##### type

character

##### presence

required

##### value

e.g. GaAs

Name of material to which this set of parameters applies. Name has to be listed in *\$default-materials*.

#### number-of-parameters

##### type

integer

##### presence

required

##### value

e.g. 4

Control parameter if the number of parameters provided is the same as demanded.

There are two sets of parameters, one for electrons (n) and one for holes (p).

#### n-N-ref

##### type

double

**presence**  
required  
**value**  
e.g.  $7.1\text{e}15$   
**unit**  
( $\text{cm}^{-3}$ )

The  $N_{\text{n,ref}}$  parameter for electrons as specified in the equation above.

**n-tau**  
**type**  
double  
**presence**  
required  
**value**  
e.g.  $4.26\text{e}-4$   
**unit**  
(s)

The  $\tau_{\text{n}}$  parameter for electrons as specified in the equation above.

**n-bow-N-ref**  
**type**  
double  
**presence**  
optional  
**value**  
e.g.  $0.0$   
**unit**  
( $\text{cm}^{-3}$ )

For ternary alloys there are also bowing parameters possible. **n-bow-N-ref** =  $0.0$  means zero bowing, i.e. linear interpolation is used.

**n-bow-tau**  
**type**  
double  
**presence**  
optional  
**value**  
e.g.  $0.0$   
**unit**  
(s)

For ternary alloys there are also bowing parameters possible. **n-bow-tau** =  $0.0$  means zero bowing, i.e. linear interpolation is used.

**p-N-ref**  
**unit**  
( $\text{cm}^{-3}$ )

Same as **n-N-ref** but for holes.

**p-tau**

**unit**  
(s)

Same as n-tau but for holes.

#### **p-bow-N-ref**

**unit**  
(cm<sup>-3</sup>)

Same as n-bow-N-ref but for holes.

#### **p-bow-tau**

**unit**  
(s)

Same as n-bow-tau but for holes.

Example

```
!-----!  
$SRH-recombination  
!-----!  
material-name      = Si  
number-of-parameters = 4  
n-N-ref            = 7.1e15      ! [cm^-3]  
n-tau              = 4.26e-4     ! [s]  
p-N-ref            = 7.1e15      ! [cm^-3]  
p-tau              = 3.95e-4     ! [s]  
  
material-name      = GaAs  
number-of-parameters = 4  
n-N-ref            = 1.0e19      ! [cm^-3]  
n-tau              = 1.0e-9       ! [s]  
p-N-ref            = 1.0e18      ! [cm^-3]  
p-tau              = 1.0e-9       ! [s]  
  
material-name      = Al(x)Ga(1-x)As  
number-of-parameters = 4  
n-bow-N-ref        = 0.0          ! [cm^-3]  
n-bow-tau          = 0.0          ! [s]  
p-bow-N-ref        = 0.0          ! [cm^-3]  
p-bow-tau          = 0.0          ! [s]  
  
$end_SRH-recombination  
!-----!
```

There is also a keywords section in the input file for *\$SRH-recombination* which you can use to overwrite default material parameters.

### 4.2.10 \$mobility-model-arora

The documentation for this keyword is available here: [https://www.nextnano.com/nextnano3/input\\_parser/database/mobility-model-arora.htm](https://www.nextnano.com/nextnano3/input_parser/database/mobility-model-arora.htm)

### 4.2.11 \$mobility-model-constant

The constant mobility model is due to lattice scattering (phonon scattering) and depends only on the temperature. The lattice atoms oscillate about their equilibrium sites at finite temperature leading to a scattering of carriers which results in a temperature dependent mobility  $\mu_{\text{const}}$ .  $\mu_L$  is the mobility due to bulk phonon (lattice) scattering. For all semiconductors the temperature dependent lattice mobility is modeled by a power law. The parameter values used



in this model for electrons and holes, respectively, are taken from the PhD thesis of V. Palankovski, [Simulation of Heterojunction Bipolar Transistors](#) (TU Vienna).

This model is suited for undoped structure.

#### **Note**

The  $\gamma$  exponents n-gamma-lattice-temp, p-gamma-lattice-temp have opposite sign in both the PhD thesis of V. Palankovski and in the MINIMOS documentation compared to the implementation of *nextnano*<sup>3</sup>.

In this model the mobility is constant and depends only on the temperature  $T$ . The parameters in the database are given for electrons and holes,

$$\mu_{\text{const}}(T) = \mu_L \left( \frac{T}{T_0} \right)^{-\gamma},$$

where  $T_0 = 300$  K.

|                                      |                  |                 |
|--------------------------------------|------------------|-----------------|
| <b>\$mobility-model-constant</b>     |                  | <b>optional</b> |
| material-name                        | <b>character</b> | <b>required</b> |
| number-of-parameters                 | <b>integer</b>   | <b>required</b> |
| n-mu-lattice-temp                    | <b>double</b>    | <b>optional</b> |
| n-gamma-lattice-temp                 | <b>double</b>    | <b>optional</b> |
| p-mu-lattice-temp                    | <b>double</b>    | <b>optional</b> |
| p-gamma-lattice-temp                 | <b>double</b>    | <b>optional</b> |
| !-----                               |                  |                 |
| ! Bowing parameters for alloys       |                  |                 |
| !-----                               |                  |                 |
| n-bow-mu-lattice-temp                | <b>double</b>    | <b>optional</b> |
| n-bow-gamma-lattice-temp             | <b>double</b>    | <b>optional</b> |
| p-bow-mu-lattice-temp                | <b>double</b>    | <b>optional</b> |
| p-bow-gamma-lattice-temp             | <b>double</b>    | <b>optional</b> |
| <b>\$end_mobility-model-constant</b> |                  | <b>optional</b> |

Explanation of specifiers.

#### **material-name**

##### **type**

character

##### **presence**

required

##### **value**

e.g. Si

Name of material to which this set of parameters applies. Name has to be listed in *\$default-materials*.

#### **number-of-parameters**

##### **type**

integer

##### **presence**

required

##### **value**

e.g. 4

Control parameter if the number of parameters provided is the same as demanded.

There are two sets of parameters, one for electrons (n) and one for holes (p).

**n-mu-lattice-temp**

**type**  
double  
**presence**  
required  
**value**  
e.g. 1417  
**unit**  
(cm<sup>2</sup>/Vs)

$\mu_{L,n}$  is the bulk phonon mobility for electrons.

**n-gamma-lattice-temp**

**type**  
double  
**presence**  
required  
**value**  
e.g. 2.5

$\gamma_n$  is the exponent of the temperature dependence for electrons.

Bowing parameters

**n-bow-mu-lattice-temp**

**type**  
double  
**presence**  
optional  
**value**  
e.g. 0.0  
**unit**  
(cm<sup>2</sup>/Vs)

For ternary alloys there are also bowing parameters possible. `n-bow-mu-lattice-temp = 0.0` means zero bowing, i.e. linear interpolation is used.

**n-bow-gamma-lattice-temp**

**type**  
double  
**presence**  
optional  
**value**  
e.g. 0.0

Bowing parameters `n-bow-gamma-lattice-temp = 0.0` means zero bowing, i.e. linear interpolation is used.

**p-mu-lattice-temp**

**unit**  
(cm<sup>2</sup>/Vs)

Same as `n-mu-lattice-temp` but for holes,  $\mu_{L,p}$ .

**p-gamma-lattice-temp**

Same as `n-gamma-lattice-temp` but for holes,  $\gamma_p$ .

**p-bow-mu-lattice-temp****unit**(cm<sup>2</sup>/Vs)

Same as n-bow-mu-lattice-temp but for holes.

**p-bow-gamma-lattice-temp**

Same as n-bow-gamma-lattice-temp but for holes.

Example

```
!-----
$mobility-model-constant

material-name           = GaAs
number-of-parameters    = 4
n-mu-lattice-temp       = 8500          ! [cm^2/Vs]  PhD thesis V.
→Palankovski
n-gamma-lattice-temp    = 2.2          ! []        PhD thesis V.
→Palankovski but opposite sign compared to MINIMOS
p-mu-lattice-temp       = 800          ! [cm^2/Vs]  PhD thesis V.
→Palankovski
p-gamma-lattice-temp    = 0.9          ! []        PhD thesis V.
→Palankovski but opposite sign compared to MINIMOS

material-name           = Al(x)Ga(1-x)As
number-of-parameters    = 4
n-bow-mu-lattice-temp   = 0.0          ! [cm^2/Vs]
n-bow-gamma-lattice-temp = 0.0          ! []
p-bow-mu-lattice-temp   = 0.0          ! [cm^2/Vs]
p-bow-gamma-lattice-temp = 0.0          ! []

$send_mobility-model-constant
!-----
```

**4.2.12 \$mobility-model-dar**

For the Darwish mobility model, see [DarwishIEEE1997] and [Klaassen1992].

| \$mobility-model-dar |           | optional |
|----------------------|-----------|----------|
| material-name        | character | required |
| number-of-parameters | integer   | required |
| n-acoust-B           | double    | optional |
| n-acoust-C           | double    | optional |
| n-ac-C-exponent      | double    | optional |
| n-acoust-kappa       | double    | optional |
| n-bulk-mu-min        | double    | optional |
| n-bulk-mu-max        | double    | optional |
| n-bulk-mu-Nref1      | double    | optional |
| n-bulk-thetan        | double    | optional |
| n-bulk-alpha1        | double    | optional |
| n-bulk-f-CW          | double    | optional |
| n-bulk-f-BH          | double    | optional |
| n-bulk-P-CW-const    | double    | optional |
| n-bulk-P-BH-const    | double    | optional |
| n-bulk-G-s1          | double    | optional |
| n-bulk-G-s2          | double    | optional |

(continues on next page)

(continued from previous page)

|                   |        |          |
|-------------------|--------|----------|
| n-bulk-G-s3       | double | optional |
| n-bulk-G-s4       | double | optional |
| n-bulk-G-s5       | double | optional |
| n-bulk-G-s6       | double | optional |
| n-bulk-G-s7       | double | optional |
| n-bulk-F-r1       | double | optional |
| n-bulk-F-r2       | double | optional |
| n-bulk-F-r3       | double | optional |
| n-bulk-F-r4       | double | optional |
| n-bulk-F-r5       | double | optional |
| n-bulk-F-r6       | double | optional |
| n-sr-delta        | double | optional |
| n-sr-alpha        | double | optional |
| n-sr-etha         | double | optional |
| n-sr-A            | double | optional |
| n-a-E             | double | optional |
| n-b-E             | double | optional |
| n-E0-saturation   | double | optional |
| n-bulk-Z-c        | double | optional |
| n-bulk-Z-Nref     | double | optional |
|                   |        |          |
| p-acoust-B        | double | optional |
| p-acoust-C        | double | optional |
| p-ac-C-exponent   | double | optional |
| p-acoust-kappa    | double | optional |
| p-bulk-mu-min     | double | optional |
| p-bulk-mu-max     | double | optional |
| p-bulk-mu-Nref1   | double | optional |
| p-bulk-thetan     | double | optional |
| p-bulk-alpha1     | double | optional |
| p-bulk-f-CW       | double | optional |
| p-bulk-f-BH       | double | optional |
| p-bulk-P-CW-const | double | optional |
| p-bulk-P-BH-const | double | optional |
| p-bulk-G-s1       | double | optional |
| p-bulk-G-s2       | double | optional |
| p-bulk-G-s3       | double | optional |
| p-bulk-G-s4       | double | optional |
| p-bulk-G-s5       | double | optional |
| p-bulk-G-s6       | double | optional |
| p-bulk-G-s7       | double | optional |
| p-bulk-F-r1       | double | optional |
| p-bulk-F-r2       | double | optional |
| p-bulk-F-r3       | double | optional |
| p-bulk-F-r4       | double | optional |
| p-bulk-F-r5       | double | optional |
| p-bulk-F-r6       | double | optional |
| p-sr-delta        | double | optional |
| p-sr-alpha        | double | optional |
| p-sr-etha         | double | optional |
| p-sr-A            | double | optional |
| p-a-E             | double | optional |
| p-b-E             | double | optional |
| p-E0-saturation   | double | optional |
| p-bulk-Z-c        | double | optional |
| p-bulk-Z-Nref     | double | optional |

(continues on next page)

(continued from previous page)

```

!-----
! Bowing parameters for alloys
!-----

n-bow-acoust-B           double   optional
n-bow-acoust-C           double   optional
n-bow-ac-C-exponent      double   optional
n-bow-acoust-kappa       double   optional
n-bow-bulk-mu-min         double   optional
n-bow-bulk-mu-max         double   optional
n-bow-bulk-mu-Nref1       double   optional
n-bow-bulk-thetan         double   optional
n-bow-bulk-alpha1         double   optional
n-bow-bulk-f-CW           double   optional
n-bow-bulk-f-BH           double   optional
n-bow-bulk-P-CW-const     double   optional
n-bow-bulk-P-BH-const     double   optional
n-bow-bulk-G-s1           double   optional
n-bow-bulk-G-s2           double   optional
n-bow-bulk-G-s3           double   optional
n-bow-bulk-G-s4           double   optional
n-bow-bulk-G-s5           double   optional
n-bow-bulk-G-s6           double   optional
n-bow-bulk-G-s7           double   optional
n-bow-bulk-F-r1           double   optional
n-bow-bulk-F-r2           double   optional
n-bow-bulk-F-r3           double   optional
n-bow-bulk-F-r4           double   optional
n-bow-bulk-F-r5           double   optional
n-bow-bulk-F-r6           double   optional
n-bow-sr-delta           double   optional
n-bow-sr-alpha           double   optional
n-bow-sr-etha            double   optional
n-bow-sr-A               double   optional
n-bow-a-E                double   optional
n-bow-b-E                double   optional
n-bow-E0-saturation       double   optional
n-bow-bulk-Z-c            double   optional
n-bow-bulk-Z-Nref         double   optional

p-bow-acoust-B           double   optional
p-bow-acoust-C           double   optional
p-bow-ac-C-exponent      double   optional
p-bow-acoust-kappa       double   optional
p-bow-bulk-mu-min         double   optional
p-bow-bulk-mu-max         double   optional
p-bow-bulk-mu-Nref1       double   optional
p-bow-bulk-thetan         double   optional
p-bow-bulk-alpha1         double   optional
p-bow-bulk-f-CW           double   optional
p-bow-bulk-f-BH           double   optional
p-bow-bulk-P-CW-const     double   optional
p-bow-bulk-P-BH-const     double   optional
p-bow-bulk-G-s1           double   optional
p-bow-bulk-G-s2           double   optional

```

(continues on next page)

(continued from previous page)

|                                  |        |          |
|----------------------------------|--------|----------|
| p-bow-bulk-G-s3                  | double | optional |
| p-bow-bulk-G-s4                  | double | optional |
| p-bow-bulk-G-s5                  | double | optional |
| p-bow-bulk-G-s6                  | double | optional |
| p-bow-bulk-G-s7                  | double | optional |
| p-bow-bulk-F-r1                  | double | optional |
| p-bow-bulk-F-r2                  | double | optional |
| p-bow-bulk-F-r3                  | double | optional |
| p-bow-bulk-F-r4                  | double | optional |
| p-bow-bulk-F-r5                  | double | optional |
| p-bow-bulk-F-r6                  | double | optional |
| p-bow-sr-delta                   | double | optional |
| p-bow-sr-alpha                   | double | optional |
| p-bow-sr-etha                    | double | optional |
| p-bow-sr-A                       | double | optional |
| p-bow-a-E                        | double | optional |
| p-bow-b-E                        | double | optional |
| p-bow-E0-saturation              | double | optional |
| p-bow-bulk-Z-c                   | double | optional |
| p-bow-bulk-Z-Nref                | double | optional |
| <b>\$send_mobility-model-dar</b> |        | optional |

**Note**

The mobility models `mobility-model-dar` (Darwish) has been developed for silicon only.

**Note**

In nextnano we use the nominal dopant concentration as specified in the input file and not the ionized one.

## Example

```
!-----!
$mobility-model-dar

material-name           = Si
number-of-parameters   = 70

n-acoust-B              = 3.61e7      ! [cm/s]
n-acoust-C              = 1.70e4      ! [(cm^2/Vs) (V/cm)**(1/
→3)cm**(3tau)]
n-ac-C-exponent         = 0.0233      ! []
n-acoust-kappa          = 1.7         ! []
n-bulk-mu-min            = 52.2        ! [cm^2/Vs]
n-bulk-mu-max            = 1417.0      ! [cm^2/Vs]
n-bulk-mu-Nref1          = 9.68e16     ! [1/cm^3]
n-bulk-thetan           = 1.0         ! []
n-bulk-alpha1           = 0.68        ! []
n-bulk-f-CW              = 2.495      ! []
n-bulk-f-BH              = 3.828      ! []
n-bulk-P-CW-const        = 3.97e13     ! []
n-bulk-P-BH-const        = 1.36e20     ! []
n-bulk-G-s1              = 0.89233     ! []
```

(continues on next page)

(continued from previous page)

```

n-bulk-G-s2           = 0.41372      ! []
n-bulk-G-s3           = 0.19778      ! []
n-bulk-G-s4           = 0.28227      ! []
n-bulk-G-s5           = 0.005978     ! []
n-bulk-G-s6           = 1.80618      ! []
n-bulk-G-s7           = 0.72169      ! []
n-bulk-F-r1           = 0.7643       ! []
n-bulk-F-r2           = 2.2999       ! []
n-bulk-F-r3           = 6.5502       ! []
n-bulk-F-r4           = 2.3670       ! []
n-bulk-F-r5           = -0.01552     ! []
n-bulk-F-r6           = 0.6478       ! []
n-sr-delta            = 3.58e18      ! [V/s]
n-sr-alpha            = 6.85e-21     ! []
n-sr-etha             = 0.0767      ! []
n-sr-A                = 2.58         ! []
n-a-E                 = 2.0           ! []
n-b-E                 = 0.5           ! []
n-E0-saturation       = 8.0e3        ! [V/cm]
n-bulk-Z-c             = 0.21         ! []
n-bulk-Z-Nref         = 4.0e20       ! [1/cm^3]

p-acoust-B            = 1.51e7        ! [cm/s]
p-acoust-C            = 4.18e3        ! [(cm^2/Vs) (V/cm)**(1/
→ 3)cm**(3tau)]
p-ac-C-exponent       = 0.0119       ! []
p-acoust-kappa        = 0.9          ! []
p-bulk-mu-min          = 44.9         ! [cm^2/Vs]
p-bulk-mu-max         = 470.5        ! [cm^2/Vs]
p-bulk-mu-Nref1       = 2.23e17      ! [1/cm^3]
p-bulk-thetan         = 1.0          ! []
p-bulk-alpha1         = 0.719        ! []
p-bulk-f-CW           = 2.495        ! []
p-bulk-f-BH           = 3.828        ! []
p-bulk-P-CW-const     = 3.97e13      ! []
p-bulk-P-BH-const     = 1.36e20      ! []
p-bulk-G-s1           = 0.89233      ! []
p-bulk-G-s2           = 0.41372      ! []
p-bulk-G-s3           = 0.19778      ! []
p-bulk-G-s4           = 0.28227      ! []
p-bulk-G-s5           = 0.005978     ! []
p-bulk-G-s6           = 1.80618      ! []
p-bulk-G-s7           = 0.72169      ! []
p-bulk-F-r1           = 0.7643       ! []
p-bulk-F-r2           = 2.2999       ! []
p-bulk-F-r3           = 6.5502       ! []
p-bulk-F-r4           = 2.3670       ! []
p-bulk-F-r5           = -0.01552     ! []
p-bulk-F-r6           = 0.6478       ! []
p-sr-delta            = 4.10e15      ! [V/s]
p-sr-alpha            = 7.82e-21     ! []
p-sr-etha             = 0.123        ! []
p-sr-A                = 2.18         ! []
p-a-E                 = 1.0           ! []
p-b-E                 = 1.0           ! []
p-E0-saturation       = 1.95e4       ! [V/cm]

```

(continues on next page)

(continued from previous page)

```

p-bulk-Z-c          = 0.50          ! []
p-bulk-Z-Nref       = 7.2e20        ! [1/cm^3]

$send_$mobility-model-dar
!-----!

```

### 4.2.13 \$mobility-model-lom

For the Lombardi mobility model, see [\[LombardiIEEE1988\]](#).

| <b>\$mobility-model-lom</b>    |                  | <b>optional</b> |
|--------------------------------|------------------|-----------------|
| material-name                  | <b>character</b> | <b>required</b> |
| number-of-parameters           | <b>integer</b>   | <b>required</b> |
|                                |                  |                 |
| n-acoust-B                     | double           | optional        |
| n-acoust-C                     | double           | optional        |
| n-ac-C-exponent                | double           | optional        |
| n-bulk-mu-zero                 | double           | optional        |
| n-bulk-mu-max                  | double           | optional        |
| n-bulk-mu-1                    | double           | optional        |
| n-bulk-gamma                   | double           | optional        |
| n-bulk-Cr                      | double           | optional        |
| n-bulk-C0                      | double           | optional        |
| n-bulk-alpha                   | double           | optional        |
| n-bulk-beta                    | double           | optional        |
| n-sr-delta                     | double           | optional        |
| n-alpha-E                      | double           | optional        |
| n-beta-E                       | double           | optional        |
| n-E0-saturation                | double           | optional        |
| n-v0-saturation                | double           | optional        |
| n-temp-dependence-v            | double           | optional        |
| n-kappa-v                      | double           | optional        |
| n-T0-v-saturation              | double           | optional        |
|                                |                  |                 |
| p-acoust-B                     | double           | optional        |
| p-acoust-C                     | double           | optional        |
| p-ac-C-exponent                | double           | optional        |
| p-bulk-mu-zero                 | double           | optional        |
| p-bulk-mu-max                  | double           | optional        |
| p-bulk-mu-1                    | double           | optional        |
| p-bulk-gamma                   | double           | optional        |
| p-bulk-Cr                      | double           | optional        |
| p-bulk-C0                      | double           | optional        |
| p-bulk-alpha                   | double           | optional        |
| p-bulk-beta                    | double           | optional        |
| p-sr-delta                     | double           | optional        |
| p-alpha-E                      | double           | optional        |
| p-beta-E                       | double           | optional        |
| p-E0-saturation                | double           | optional        |
| p-v0-saturation                | double           | optional        |
| p-temp-dependence-v            | double           | optional        |
| p-kappa-v                      | double           | optional        |
| p-T0-v-saturation              | double           | optional        |
|                                |                  |                 |
| !-----!                        |                  |                 |
| ! Bowing parameters for alloys |                  |                 |

(continues on next page)



(continued from previous page)

|                          |        |          |
|--------------------------|--------|----------|
| !-----                   |        |          |
| n-bow-acoust-B           | double | optional |
| n-bow-acoust-C           | double | optional |
| n-bow-ac-C-exponent      | double | optional |
| n-bow-bulk-mu-zero       | double | optional |
| n-bow-bulk-mu-max        | double | optional |
| n-bow-bulk-mu-1          | double | optional |
| n-bow-bulk-gamma         | double | optional |
| n-bow-bulk-Cr            | double | optional |
| n-bow-bulk-C0            | double | optional |
| n-bow-bulk-alpha         | double | optional |
| n-bow-bulk-beta          | double | optional |
| n-bow-sr-delta           | double | optional |
| n-bow-alpha-E            | double | optional |
| n-bow-beta-E             | double | optional |
| n-bow-E0-saturation      | double | optional |
| n-bow-v0-saturation      | double | optional |
| n-bow-temp-dependence-v  | double | optional |
| n-bow-kappa-v            | double | optional |
| n-bow-T0-v-saturation    | double | optional |
|                          |        |          |
| p-bow-acoust-B           | double | optional |
| p-bow-acoust-C           | double | optional |
| p-bow-ac-C-exponent      | double | optional |
| p-bow-bulk-mu-zero       | double | optional |
| p-bow-bulk-mu-max        | double | optional |
| p-bow-bulk-mu-1          | double | optional |
| p-bow-bulk-gamma         | double | optional |
| p-bow-bulk-Cr            | double | optional |
| p-bow-bulk-C0            | double | optional |
| p-bow-bulk-alpha         | double | optional |
| p-bow-bulk-beta          | double | optional |
| p-bow-sr-delta           | double | optional |
| p-bow-alpha-E            | double | optional |
| p-bow-beta-E             | double | optional |
| p-bow-E0-saturation      | double | optional |
| p-bow-v0-saturation      | double | optional |
| p-bow-temp-dependence-v  | double | optional |
| p-bow-kappa-v            | double | optional |
| p-bow-T0-v-saturation    | double | optional |
| \$end_mobility-model-lom |        | optional |

**Note**

The mobility models `mobility-model-lom` (Lombardi) has been developed for silicon only.

**Note**

In nextnano we use the nominal dopant concentration as specified in the input file and not the ionized one.

Example

```

!-----!
$mobility-model-dar

material-name           = Si
number-of-parameters    = 38

n-acoust-B              = 4.75e7      ! [cm/s]
n-acoust-C              = 1.74e5      ! [see paper]
n-ac-C-exponent         = 0.125      ! []
n-bulk-mu-zero          = 52.2       ! [cm^2/Vs]
n-bulk-mu-max           = 1417.0     ! [cm^2/Vs]
n-bulk-mu-1             = 43.4       ! [cm^2/Vs]
n-bulk-gamma            = 2.5        ! []
n-bulk-Cr               = 9.68e16    ! [1/cm^3]
n-bulk-C0               = 3.43e20    ! [1/cm^3]
n-bulk-alpha            = 0.680     ! []
n-bulk-beta             = 2.0        ! []
n-sr-delta              = 5.82e14    ! [V/s]
n-alpha-E               = 2.0        ! []
n-beta-E                = 0.5        ! []
n-E0-saturation         = 8.0e3      ! [V/cm]
n-v0-saturation         = 1.03e7     ! [cm/s]      see SIMBA documentation
n-temp-dependence-v     = 0.0        ! [cm/Ks]     see SIMBA documentation
n-kappa-v               = 2          ! []          see SIMBA documentation
n-T0-v-saturation       = 300.0     ! [K]         see SIMBA documentation

p-acoust-B              = 9.93e7     ! [cm/s]
p-acoust-C              = 8.84e5     ! [see paper]
p-ac-C-exponent         = 0.0317    ! []
p-bulk-mu-zero          = 44.9       ! [cm^2/Vs]
p-bulk-mu-max           = 470.5     ! [cm^2/Vs]
p-bulk-mu-1             = 29.0       ! [cm^2/Vs]
p-bulk-gamma            = 2.2        ! []
p-bulk-Cr               = 2.23e17    ! [1/cm^3]
p-bulk-C0               = 6.1e20     ! [1/cm^3]
p-bulk-alpha            = 0.719     ! []
p-bulk-beta             = 2.0        ! []
p-sr-delta              = 2.05e14    ! [V/s]
p-alpha-E               = 1.0        ! []
p-beta-E                = 1.0        ! []
p-E0-saturation         = 1.95e4     ! [V/cm]
p-v0-saturation         = 1.03e7     ! [cm/s]      see SIMBA documentation
p-temp-dependence-v     = 0.0        ! [cm/Ks]     see SIMBA documentation
p-kappa-v               = 1          ! []          see SIMBA documentation
p-T0-v-saturation       = 300.0     ! [K]         see SIMBA documentation

$end_$mobility-model-lom
!-----!

```

#### 4.2.14 \$mobility-model-masetti

The documentation for this keyword is available here: [https://www.nextnano.com/nextnano3/input\\_parser/database/mobility-model-masetti.htm](https://www.nextnano.com/nextnano3/input_parser/database/mobility-model-masetti.htm)

### 4.2.15 \$mobility-model-minimos

The documentation for this keyword is available here: [https://www.nextnano.com/nextnano3/input\\_parser/database/mobility-model-minimos.htm](https://www.nextnano.com/nextnano3/input_parser/database/mobility-model-minimos.htm)

### 4.2.16 \$mobility-model-simba

The documentation for this keyword is available here: [https://www.nextnano.com/nextnano3/input\\_parser/database/mobility-model-simba.htm](https://www.nextnano.com/nextnano3/input_parser/database/mobility-model-simba.htm)

### 4.2.17 \$mobility-models

Several mobility models can be chosen that consider phonon scattering, impurity scattering and electric field dependence.

|                              |                  |                 |
|------------------------------|------------------|-----------------|
| <b>\$mobility-models</b>     |                  | <b>required</b> |
| model-name                   | <b>character</b> | <b>required</b> |
| model-type-number            | <b>integer</b>   | <b>required</b> |
| <b>\$end_mobility-models</b> |                  | <b>required</b> |

#### **Note**

The mobility models `mobility-model-lom` (Lombardi) and `mobility-model-dar` (Darwish) have been developed for silicon only.

Example

```
!-----!
$mobility-models
  model-name      = mobility-model-simba-0      ! SIMBA, parallel E field_
  ↳dependence according to model 0
  model-type-number = 1

  model-name      = mobility-model-simba-1      ! SIMBA, parallel E field_
  ↳dependence according to model 1
  model-type-number = 2

  model-name      = mobility-model-simba-2      ! SIMBA, parallel E field_
  ↳dependence according to model 2
  model-type-number = 3

  model-name      = mobility-model-simba-3      ! SIMBA, parallel E field_
  ↳dependence according to model 3
  model-type-number = 4

  model-name      = mobility-model-simba-4      ! SIMBA, parallel E field_
  ↳dependence according to model 4
  model-type-number = 5

  model-name      = mobility-model-simba-5      ! SIMBA, parallel E field_
  ↳dependence according to model 5
  model-type-number = 6

  model-name      = mobility-model-simba-0e     ! with perpendicular E field_
  ↳dependence
  model-type-number = 7
```

(continues on next page)

(continued from previous page)

```

model-name          = mobility-model-simba-1e    ! with perpendicular E field.
↪dependence
model-type-number   = 8

model-name          = mobility-model-simba-2e    ! with perpendicular E field.
↪dependence
model-type-number   = 9

model-name          = mobility-model-simba-3e    ! with perpendicular E field.
↪dependence
model-type-number   = 10

model-name          = mobility-model-simba-4e    ! with perpendicular E field.
↪dependence
model-type-number   = 11

model-name          = mobility-model-simba-5e    ! with perpendicular E field.
↪dependence
model-type-number   = 12

model-name          = mobility-model-lom          ! Lobardi model, for Si only
model-type-number   = 13

model-name          = mobility-model-dar          ! Darwish model, for Si only
model-type-number   = 14

model-name          = mobility-model-constant    ! constant mobility model,
↪for undoped structures only (phonon scattering)
model-type-number   = 15

model-name          = mobility-model-masetti      ! Masetti model    - doping.
↪dependent model (phonon and impurity scattering)
model-type-number   = 16

model-name          = mobility-model-arora        ! Arora model      - doping.
↪dependent model (phonon and impurity scattering)
model-type-number   = 17

model-name          = mobility-model-minimos     ! Minimos 6 model - doping.
↪dependent model (phonon and impurity scattering)
model-type-number   = 18

$end_mobility-models
!-----!

```

## 4.2.18 \$transport-models

Charge carrier models.

|                               |                  |                 |
|-------------------------------|------------------|-----------------|
| <b>\$transport-models</b>     |                  | <b>required</b> |
| model-name                    | <b>character</b> | <b>required</b> |
| model-type-number             | <b>integer</b>   | <b>required</b> |
| <b>\$end_transport-models</b> |                  | <b>required</b> |

Transport model for drift-diffusion equation.

**model-name**

**type**  
character

**presence**  
required

**value**  
simple-drift-model

More information available here: [\\$simple-drift-models](#)

#### model-type-number

**type**  
integer

**presence**  
required

**value**  
1

Only 1 model implemented so far.

Example

```
!-----!
$transport-models                               !
↪      !                                         !
model-name      = simple-drift-model             !
model-type-number = 1                           !
$end_transport-models                           !
↪      !                                         !
!-----!
```

### 4.2.19 \$buffer-constant-A(T)

Constant  $A(T)$  used for buffer calculations: The  $pK_a$  value depends on the ionic strength.

|                                   |                     |                 |
|-----------------------------------|---------------------|-----------------|
| <b>\$buffer-constant-A(T)</b>     |                     | <b>optional</b> |
| constant-Centigrade-to-Kelvin     | <b>double</b>       | <b>required</b> |
| T_A(T)                            | <b>double_array</b> | <b>required</b> |
| <b>\$end_buffer-constant-A(T)</b> |                     | <b>optional</b> |

Example

```
!-----!
$buffer-constant-A(T)                           !
constant-Centigrade-to-Kelvin = 273.15          ! Kelvin = Celsius + ↪
↪273.15                                         !
!-----!
!=====!
!              first column: T[C]  second column: A(T) !
!=====!
T_A(T)          =   0.0      0.4918      !   0° C = 273.15 K
                  10.0      0.4989      !  10° C = 283.15 K
                  20.0      0.5070      !  20° C = 293.15 K
                  25.0      0.5114      !  25° C = 298.15 K
                  30.0      0.5161      !  30° C = 303.15 K
                  37.0      0.5321      !  37° C = 310.15 K
                  40.0      0.5262      !  40° C = 313.15 K
                  50.0      0.5373      !  50° C = 323.15 K
```

(continues on next page)

(continued from previous page)

|                            |        |                     |
|----------------------------|--------|---------------------|
| 60.0                       | 0.5494 | ! 60° C = 333.15 K  |
| 70.0                       | 0.5625 | ! 70° C = 343.15 K  |
| 80.0                       | 0.5767 | ! 80° C = 353.15 K  |
| 90.0                       | 0.5920 | ! 90° C = 363.15 K  |
| 100.0                      | 0.6086 | ! 100° C = 373.15 K |
| \$end_buffer-constant-A(T) |        | !                   |
| !-----!                    |        | !                   |

The left column of the specifier T\_A(T) contains the temperature in degrees of Centigrade (Celsius) between 0° C and 100° C.

The right column of the specifier T\_A(T) contains the corresponding value of the constant A as a function of temperature T, i.e. A(T).

The values are taken from page 30 of [Beynon1996].

They can also be approximated by a second-order polynomial ([Beynon1988]):

$$A(T) = 0.4918 + 0.0006614T + 0.000004975T^2$$

If the keyword \$buffer-constant-A(T) is present in the input file, the values for this keyword in the database are overwritten.

### Physical significance of this parameter

The ionic strength of an electrolyte influences the  $pK_a$  value of the buffer. This dependence can be described by the following equation (sometimes known as the Debye-Hückel relationship) where the constant A(T) enters.

$$pK'_a = pK_a + (2z_a - 1)[AI^{1/2}/(1 + I^{1/2}) - 0.1I]$$

where  $I$  is the ionic strength and  $z_a$  is the charge on the conjugate acid species.  $pK'_a$  is the modified  $pK_a$  value. The value of  $A$  (sometimes called Debye-Hückel parameter) is about 0.5 but it is temperature dependent.

Internally, the program takes the temperature  $T_0$  that is given in the input file under the keyword \$global-parameters (in units of Kelvin) and interpolates linearly between the two appropriate neighboring  $A(T)$  values to find the value for  $A(T_0)$ . The conversion between temperature in Kelvin and Centigrade is done by the constant: constant-Centigrade-to-Kelvin = 273.15

### Example

```
lattice-temperature = 288.15      ! 288.15 [K] = 15° [C] + 273.15 [K]

A(T = 10° C) = 0.4989
A(T = 20° C) = 0.5070
```

=> Internally the program calculates the value for  $A(T = 15° C) = (0.4989 + 0.5070)/2 = 0.50295$ .

The following interpolation formula is used:

```
A(T = x°C) = A(T_i) + slope * ('lattice-temperature' - 'constant-Centigrade-to-Kelvin'
↪ - T_i) =
              = A(T_i) + slope * ('lattice-temperature' - '273.15'
↪ - T_i) =
```

where

$$\text{slope} = (A(T_{i+1}) - A(T_i)) / (T_{i+1} - T_i)$$

and it holds:

$$T_i < \text{'lattice-temperature'} - \text{'273.15'} < T_{i+1}$$

$T_{i+1}$  and  $T_i$  are the closest temperature points above and below the specified temperature lattice-temperature.

- If the `lattice-temperature` is smaller than the smallest value of  $A(T)$ , the smallest  $A(T)$  value is taken.
- If the `lattice-temperature` is larger than the largest value of  $A(T)$ , the largest  $A(T)$  value is taken.

The value of  $A$  *always* depends on temperature. This can only be switched off by specifying only *one* value of  $T$  and  $A(T)$  in the database or in the input file. The values for  $T$  and  $A(T)$  that are specified in the database can be overwritten in the input file. For details, have a look at the input file keyword `$buffer-constant-A(T)`.

## 4.2.20 \$buffer-solutions

The documentation for this keyword is available here: [https://www.nextnano.com/nextnano3/input\\_parser/database/buffer-solutions.htm](https://www.nextnano.com/nextnano3/input_parser/database/buffer-solutions.htm)

## 4.2.21 \$physical-constants

The base system for units is SI.

The following physical constants are used within *nextnano*<sup>3</sup>.

|                                 |        |                 |                   |         |                                        |
|---------------------------------|--------|-----------------|-------------------|---------|----------------------------------------|
| <b>\$physical-constants</b>     |        | <b>required</b> |                   |         |                                        |
| electron-charge                 | double | required        | ! [As]            | = [C]   | e: <span style="color: red;">_</span>  |
| → elementary charge             |        |                 |                   |         |                                        |
| electron-mass                   | double | required        | ! [kg]            |         | m0: <span style="color: red;">_</span> |
| → electron mass                 |        |                 |                   |         |                                        |
| planck-constant                 | double | required        | ! [Js]            |         | h_                                     |
| → bar: Planck's constant        |        |                 |                   |         |                                        |
| speed-of-light                  | double | required        | ! [m/s]           | (exact) | c: <span style="color: red;">_</span>  |
| → speed of light in vacuum      |        |                 |                   |         |                                        |
| boltzmann-constant              | double | required        | ! [J/K]           |         | k_                                     |
| → B: Boltzmann constant         |        |                 |                   |         |                                        |
| vacuum-permittivity             | double | required        | ! [As/Vm] = [F/m] | (exact) | <span style="color: red;">_</span>     |
| → epsilon0: electric constant   |        |                 |                   |         |                                        |
| avogadro-number                 | double | required        | ! [1/mol]         |         | N_                                     |
| → A: Avogadro number            |        |                 |                   |         |                                        |
| <b>\$end_physical-constants</b> |        | <b>required</b> |                   |         |                                        |

Example

```

!-----!
$physical-constants      !
electron-charge    = -1.6021766208e-19  ! [C] = [As]      -1.
→ 6021766208(98)e-19
electron-mass      = 9.10938356e-31      ! [kg]              9.
→ 10938356(11)e-31
planck-constant    = 6.626070040e-34      ! [Js]              6.
→ 626070040(81)e-34
speed-of-light     = 2.99792458e+8        ! [m/s]      (exact)
boltzmann-constant = 1.38064852e-23      ! [J/K]              1.
→ 38064852(79)e-23
vacuum-permittivity = 8.854187817e-12     ! [As/Vm]  (exact)  8.854187817...
→ e-12
avogadro-number    = 6.022140857e+23     ! []                6.
→ 022140857(74)e+23
$end_physical-constants      !
!-----!

```

These SI units were taken on 2019-05-10 from <https://physics.nist.gov/cuu/Constants/index.html>. The number in parentheses is the numerical value of the standard uncertainty referred to the corresponding last digits of the quoted result.

**Further constants**

- `reduced_planck_constant` (Planck constant over 2 pi) is calculated internally inside the program:  
 $\text{planck-constant}/(2\pi) = h/2\pi$

```
reduced_planck_constant = 1.054571800139113e-034 ! [Js]
↳(calculated internally from other constants)
1.054571800(13)e-34 ! [Js] (NIST)
```

- `bohr_radius` is calculated internally inside the program:

```
4 * pi * vacuum_permittivity * reduced_planck_constant^2 / (electron_
↳mass * electron_charge^2)
bohr_radius = 0.5291772105267628e-010 ! [m]
↳(calculated internally from other constants)
0.52917721067(12)e-10 ! [m] (NIST)
```

- `hydrogen_ionization_energy_J` (Rydberg constant times  $hc$  in (J)) is calculated internally inside the program:

```
electron_mass * electron_charge^4 / (32 * pi^2 * reduced_planck_constant^
↳2 * vacuum_permittivity^2)
hydrogen_ionization_energy_J = 2.179872325695729e-018 ! [J]
↳(calculated internally from other constants)
2.179872325(27)e-18 ! [J] (NIST)
```

- `hydrogen_ionization_energy_eV` (Rydberg constant times  $hc$  in (eV)) is calculated internally inside the program:

```
hydrogen_ionization_energy_J / electron-charge
hydrogen_ionization_energy_eV = 13.6056930140903 ! [eV]
↳(calculated internally from other constants)
13.605693009(84) ! [eV] (NIST)
↳corresponds to hydrogen_ionization_energy_J
```

- `Hartree_eV` (Hartree energy in (eV)) is calculated internally inside the program:

```
2 * hydrogen_ionization_energy_eV
Hartree_eV = 27.2113860281805 ! [eV]
↳(calculated by nextnano3 from other constants)
27.21138602(17) ! [eV] (NIST)
```

## Derived constants

- $[h_{\text{bar}}^2/(2 \cdot m_0)]$

```
h2b2m_Jm2 = reduced_planck_constant^2 / (2*electron_mass) =
= 6.104264214606464e-039 [J m^2]
h2b2m_evAA2 = h2b2m_Jm2 / ABS(electron_charge) * (1d10)^2 =
= 3.80998208022688 [eV AA^2] ! AA = Angstrom
```

- From the Boltzmann constant  $k_B$ , one obtains  $k_B T$  at room temperature in units of (eV):

```
kBT = 0.02585199101... [eV] (T = 298.15 K = 25°C)
kBT = 0.02569257040... [eV] (T = 300 K)
```

For input scaling factors, see *Input-scaling-factors*.

## Conversion factors

- $\mu\text{m} \Leftrightarrow \text{eV}$ :  $h \cdot c / e \cdot 10^6 = 1.23984197$

Example



$$1.23984 / 8.4 \mu\text{m} = 0.1476 \text{ eV}$$

$$1.23984 / 0.1476 \text{ eV} = 8.4 \mu\text{m}$$

- $\mu\text{m} \Leftrightarrow \text{THz}: c * 10^{-6} = 299.792458$

Example

$$299.79 / 8.4 \mu\text{m} = 35.69 \text{ THz}$$

$$299.79 / 35.69 \text{ THz} = 8.4 \mu\text{m}$$

- $10^{18} \text{ cm}^{-3} \Leftrightarrow \text{M}: 602.21415$

Example

$$30.11 / 602.2 = 0.050 \text{ M}$$

$$0.050 \text{ M} * 602.2 = 30.11$$

- $10^{18} \text{ cm}^{-3} \Leftrightarrow \text{mM}: 0.60221415$

Example

$$30.11 / 0.6022 = 50 \text{ mM}$$

$$50 \text{ mM} * 0.6022 = 30.11$$

## 4.2.22 \$input-scaling-factors

The base system for units is SI. However, most input quantities are assumed to be given in scaled form. For physical constants, see *\$physical-constants*.

|                                    |        |                 |                    |
|------------------------------------|--------|-----------------|--------------------|
| <b>\$input-scaling-factors</b>     |        | <b>required</b> |                    |
| q0                                 | double | required        |                    |
| m0                                 | double | required        |                    |
| l0                                 | double | required        |                    |
| e0                                 | double | required        |                    |
| n3d0                               | double | required        |                    |
| n2d0                               | double | required        |                    |
| n1d0                               | double | required        |                    |
| eps0                               | double | required        |                    |
| phi0                               | double | required        |                    |
| prs0                               | double | required        |                    |
| t0                                 | double | required        |                    |
| kp_k^0_zb                          | double | required        | ! zb = zinc blende |
| kp_k^2_zb                          | double | required        | ! zb = zinc blende |
| kp_k^0_wz                          | double | required        | ! wz = wurtzite    |
| kp_k^2A_wz                         | double | required        | ! wz = wurtzite    |
| kp_k^2B_wz                         | double | required        | ! wz = wurtzite    |
| <b>\$end_input-scaling-factors</b> |        | <b>required</b> |                    |

Example

```
!-----!
$input-scaling-factors      !
q0      = 1.6021766208e-19  ! charge [|electron charge e|]
m0      = 9.10938356e-31   ! mass  [free electron mass m0]
l0      = 1e-9             ! length [nm]
e0      = 1.6021766208e-19  ! energy [eV]
n3d0    = 1e24             ! 3-dimensional number density [10^18_
↪ [cm^-3]]
n2d0    = 1e16             ! 2-dimensional number density [10^12_
↪ cm^-2]
```

(continues on next page)

(continued from previous page)

```

n1d0      = 1e8                ! 1-dimensional number density [10^6 cm^-1]
→cm^-1]
eps0      = 8.854187817e-12     ! dielectric constants [vacuum-permittivity] (exact) 8.854187817...e-12
phi0      = 1.0                ! electrostatic potential [V]
prs0      = 1e9                ! pressure [GPa], e.g. for elastic constants
→constants
t0        = 1e-9               ! time [ns]
!
kp_k^0_zb = 1.6021766208e-19    ! spin-orbit coupling energy, deformation potentials [eV]
→deformation potentials [eV]
kp_k^2_zb = 6.104264214606464e-39 ! k.p parameter L,M,N,B [h_bar^2/(2*m0)]
→(2*m0)]
!
kp_k^0_wz = 1.6021766208e-19    ! spin-orbit coupling energies, deformation potentials [eV]
→deformation potentials [eV]
kp_k^2A_wz = 1.0               ! k.p parameter A1,A2,A3,A4,A5,A6 [], multiplied internally with [h_bar^2/(2*m0)]
→multiplied internally with [h_bar^2/(2*m0)]
kp_k^2B_wz = 6.104264214606464e-39 ! k.p parameter B1,B2,B3 [h_bar^2/(2*m0)] [J/m^2]
→(2*m0)] [J/m^2]
$end_input-scaling-factors      !
!-----!

```

It is assumed that input numbers are scaled to these *units*.

#### Note

$[h\_bar^2/(2*m0)] = kp\_k^2\_zb = kp\_k^2B\_wz$

### 4.2.23 \$global-parameters

Global parameters are general parameters which are valid all over the device. See also *\$global-parameters* in Keywords section.

|                                |               |                 |
|--------------------------------|---------------|-----------------|
| <b>\$global-parameters</b>     |               | <b>required</b> |
| lattice-temperature            | <b>double</b> | <b>required</b> |
| <b>\$end_global-parameters</b> |               | <b>required</b> |

Lattice temperature

#### lattice-temperature

##### type

double

##### presence

required

##### unit

(K)

##### value

e.g. 300.0

#### Note

The band gaps, i.e. the conduction band edges, are adjusted when the temperature changes. This feature can be switched off.

See *\$numeric-control: varshni-parameters-on*

In addition, the lattice constants depend on temperature. This can be switched off as well.

See *\$numeric-control: lattice-constants-temp-coeff-on*

Example

```
!-----!
$global-parameters
  lattice-temperature = 300.0      ! 300 K
$end_global-parameters
!-----!
```

#### 4.2.24 \$program\_restrictions

These are general restrictions to the program.

|                                   |                |                 |
|-----------------------------------|----------------|-----------------|
| <b>\$program_restrictions</b>     |                | <b>required</b> |
| min_dimension_domain              | <b>integer</b> | <b>required</b> |
| max_dimension_domain              | <b>integer</b> | <b>required</b> |
| minimum-grid-factor               | <b>double</b>  | <b>required</b> |
| maximum-grid-factor               | <b>double</b>  | <b>required</b> |
| delta-to-treat-grid-factor-as-one | <b>double</b>  | <b>required</b> |
| minimum-grid-width                | <b>double</b>  | <b>required</b> |
| maximum-grid-width                | <b>double</b>  | <b>required</b> |
| maximum-number-of-grid-points     | <b>integer</b> | <b>required</b> |
| lowest-temperature-allowed        | <b>double</b>  | <b>required</b> |
| <b>\$end_program_restrictions</b> |                | <b>required</b> |

Settings for domain dimensions.

##### **min\_dimension\_domain**

###### **type**

integer

###### **presence**

required

###### **value**

1

The minimum is 1, i.e. for one-dimensional simulations. A value of 0 makes sense for bulk simulations. However, this is not supported yet.

##### **max\_dimension\_domain**

###### **type**

integer

###### **presence**

required

###### **value**

3

The maximum is 3, i.e. for three-dimensional simulations.

Settings for grid spacing.

**minimum-grid-factor**

**type**

double

**presence**

required

**value**

0.1

Be careful, geometric series grow up very rapidly.

**maximum-grid-factor**

**type**

double

**presence**

required

**value**

10.0

Be careful, geometric series grow up very rapidly.

**delta-to-treat-grid-factor-as-one**

**type**

double

**presence**

required

**value**

1.0e-3

To avoid numerical inaccuracies, everything in the interval  $1.0 \pm \Delta$  is treated as 1.0.

**minimum-grid-width**

**type**

double

**presence**

required

**value**

0.01

Only a guess, if exceeded a warning message is printed to standard output. Currently, this value is irrelevant and is ignored in the program, i.e. no warning message is printed.

**maximum-grid-width**

**type**

double

**presence**

required

**value**

200.0

Only a guess, if exceeded a warning message is printed to standard output. Currently, this value is irrelevant and is ignored in the program, i.e. no warning message is printed.

**maximum-number-of-grid-points**

**type**  
integer

**presence**  
required

**value**  
 $1000000 = 100 * 100 * 100$

Only a guess, if exceeded a warning message is printed to standard output.

#### **lowest-temperature-allowed**

**type**  
double

**presence**  
required

**value**  
0.0

#### **Note**

Convergence is more difficult to achieve if the temperature is set to a very small value or to even 0.0 (K).

Example

```
!-----!
$program_restrictions
min_dimension_domain      = 1      !
max_dimension_domain      = 3      !
minimum-grid-factor       = 0.1    !
maximum-grid-factor       = 10.0   !
delta-to-treat-grid-factor-as-one = 1.0e-3 !
minimum-grid-width        = 0.01   !
maximum-grid-width        = 200.0  !
maximum-number-of-grid-points = 1000000 !
!lowest-temperature-allowed = 0.1   ! [K]
lowest-temperature-allowed = 0.0   ! [K]
$end_program_restrictions
!-----!
```

### 4.2.25 \$input\_filename

Only to specify input file name.

|                             |                  |                 |
|-----------------------------|------------------|-----------------|
| <b>\$input_filename</b>     |                  | <b>optional</b> |
| ../Syntax/database_nn3.in   | <b>character</b> | <b>optional</b> |
| <b>\$end_input_filename</b> |                  | <b>optional</b> |

Any keyword with valid syntax. Must be set to optional. The first string in the third line is the name of the input file.

### 4.2.26 \$domain-coordinates-defaults

Specify axis for growth direction with respect to simulation coordinate system.

|                                      |                |                 |
|--------------------------------------|----------------|-----------------|
| <b>\$domain-coordinates-defaults</b> |                | <b>required</b> |
| numbers-per-axis                     | <b>integer</b> | <b>required</b> |

(continues on next page)

(continued from previous page)

|                                          |               |                 |
|------------------------------------------|---------------|-----------------|
| growth-coordinate-axis                   | integer_array | required        |
| axes-possible                            | integer_array | required        |
| <b>\$end_domain-coordinates-defaults</b> |               | <b>required</b> |

Define the possible values for growth-coordinate-axis in keyword *\$domain-coordinates*. They are only relevant for

```
strain-calculation = homogeneous-strain
```

or

```
strain-calculation = hydrostatic-strain
```

Right now, the values specified here are essentially not relevant because growth-coordinate-axis is now required in the input file if it is necessary for the strain model used.

Example

```
!-----!
$domain-coordinates-defaults      !
numbers-per-axis      = 3          ! only 3 numbers can be specified
growth-coordinate-axis = 0 0 1      ! along z axis (default)
axes-possible          = 0 0 1      ! along z axis (default)
                        0 1 0        ! along y axis
                        1 0 0        ! along x axis
$end_domain-coordinates-defaults  !
!-----!
```

#### 4.2.27 \$zb-default-expectations

Number of expected parameters for zinc blende materials.

|                                      |         |                 |
|--------------------------------------|---------|-----------------|
| <b>\$zb-default-expectations</b>     |         | <b>required</b> |
| number-of-8x8kp-parameters           | integer | required        |
| number-of-6x6kp-parameters           | integer | required        |
| number-uniax-vb-def-pots             | integer | required        |
| <b>\$end_zb-default-expectations</b> |         | <b>required</b> |

Specify here how many parameters are expected.

**number-of-8x8kp-parameters**

**type**

integer

**presence**

required

**value**

6

We need 6 parameters for the 8-band  $k \cdot p$  model in zinc blende.

**number-of-6x6kp-parameters**

**type**

integer

**presence**

required

**value**  
4

We need 4 parameters for the 6-band  $\mathbf{k} \cdot \mathbf{p}$  model in zinc blende.

#### **number-of-uniax-vb-def-pots**

**type**  
integer  
**presence**  
required  
**value**  
2

We need 2 parameters for the uniaxial valence band deformation potentials in zinc blende (2 numbers usually: b, d).

Example

```
!-----!
$zb-default-expectations      !
number-of-8x8kp-parameters = 6 !
number-of-6x6kp-parameters = 4 !
number-uniax-vb-def-pots    = 2 ! b, d
$end_zb-default-expectations  !
!-----!
```

### 4.2.28 \$wz-default-expectations

Number of expected parameters for wurtzite materials.

|                                      |                |                 |
|--------------------------------------|----------------|-----------------|
| <b>\$wz-default-expectations</b>     |                | <b>required</b> |
| number-of-8x8kp-parameters           | <b>integer</b> | <b>required</b> |
| number-of-6x6kp-parameters           | <b>integer</b> | <b>required</b> |
| number-uniax-vb-def-pots             | <b>integer</b> | <b>required</b> |
| <b>\$end_wz-default-expectations</b> |                | <b>required</b> |

Specify here how many parameters are expected.

#### **number-of-8x8kp-parameters**

**type**  
integer  
**presence**  
required  
**value**  
13

We need 13 parameters for the 8-band  $\mathbf{k} \cdot \mathbf{p}$  model in wurtzite.

#### **number-of-6x6kp-parameters**

**type**  
integer  
**presence**  
required  
**value**  
9

We need 9 parameters for the 6-band  $\mathbf{k} \cdot \mathbf{p}$  model in wurtzite.

**number-of-uniax-vb-def-pots**

**type**  
integer

**presence**  
required

**value**  
6

We need 6 parameters for the uniaxial valence band deformation potentials in wurtzite (6 numbers usually: d1, d2, d3, d4, d5, d6).

Example

```
!-----!  
$zb-default-expectations      !  
  number-of-8x8kp-parameters = 13  !  
  number-of-6x6kp-parameters = 9   !  
  number-uniax-vb-def-pots   = 6    ! d1, d2, d3, d4, d5, d6  
$end_zb-default-expectations  !  
!-----!
```

#### 4.2.29 \$zb-restrictions

Some restrictions apply for zinc blende materials.

|                               |                      |                 |
|-------------------------------|----------------------|-----------------|
| <b>\$zb-restrictions</b>      |                      | <b>required</b> |
| miller-size                   | <b>integer</b>       | <b>required</b> |
| miller-default-direction-of-x | <b>integer_array</b> | <b>required</b> |
| miller-default-direction-of-y | <b>integer_array</b> | <b>required</b> |
| direction-cosines             | <b>double_array</b>  | <b>required</b> |
| miller-direction-of-cx        | <b>integer_array</b> | <b>required</b> |
| miller-direction-of-cy        | <b>integer_array</b> | <b>required</b> |
| miller-direction-of-cz        | <b>integer_array</b> | <b>required</b> |
| lattice-constants-for-cxyz    | <b>double_array</b>  | <b>required</b> |
| gamma-is-cb-number            | <b>integer</b>       | <b>required</b> |
| l-is-cb-number                | <b>integer</b>       | <b>required</b> |
| x-is-cb-number                | <b>integer</b>       | <b>required</b> |
| <b>\$end_zb-restrictions</b>  |                      | <b>required</b> |

Explanations

**miller-size**

**type**  
integer

**presence**  
required

**value**  
3

There are three Miller indices altogether that define the (hkl) **plane**. Coincidentally, in zincblende the [hkl] vector is perpendicular to the (hkl) plane.

**miller-default-direction-of-x**

**type**  
integer\_array

**presence**  
required



Three-digit Miller indices of the (hkl) plane perpendicular to x axis of simulation coordinate system.

#### **miller-default-direction-of-y**

**type**  
integer\_array

**presence**  
required

Three-digit Miller indices of the (hkl) plane perpendicular to y axis of simulation coordinate system.

```
miller-default-direction-of-x = 1 0 0 ! (100 plane) ==> [100] direction
miller-default-direction-of-y = 0 1 0 ! (010 plane) ==> [010] direction
```

This corresponds to x axis and y axis, respectively, in simulation coordinate system, i.e. the x axis is perpendicular to the (1 0 0) plane, i.e. x axis is along the [100] direction in zinc blende, and the y axis is perpendicular to the (0 1 0) plane, i.e. y axis is along the [010] direction in zinc blende.

These value can be overwritten in *\$domain-coordinates* (hkl-x-direction, hkl-y-direction, hkl-z-direction).

Direction cosines

#### **direction-cosines**

**type**  
double\_array

**presence**  
required

**value**  
0.0 0.0 0.0

Direction cosines between lattice vectors.  $g_1 \cdot g_2$ ,  $g_2 \cdot g_3$ ,  $g_1 \cdot g_3$   $g_i \dots$  unit vectors in lattice directions.

Direction cosine refers to the cosine of the angle between any two vectors. Direction cosines are useful for forming direction cosine matrices that express one set of orthonormal basis vectors in terms of another set, or for expressing a known vector in a different basis. For zinc blende, we use:

```
g_ik = ( 1 0 0 )
      ( 0 1 0 )
      ( 0 0 1 )
```

Three-digit Miller indices of the (hkl) plane:

#### **miller-direction-of-cx**

**type**  
integer\_array

**value**  
1 0 0

Corresponds to x axis in crystal coordinate system, i.e. the x axis is perpendicular to the (1 0 0) plane, i.e. x axis is along the [100] direction in zinc blende.

#### **miller-direction-of-cy**

**type**  
integer\_array

**value**  
0 1 0

Corresponds to y axis in crystal coordinate system, i.e. the y axis is perpendicular to the (0 0 1) plane, i.e. y axis is along the [010] direction in zinc blende.

**miller-direction-of-cz**

**type**  
integer\_array

**value**  
0 0 1

Corresponds to z axis in crystal coordinate system, i.e. the z axis is perpendicular to the (0 0 1) plane, i.e. z axis is along the [001] direction in zinc blende.

```
miller-direction-of-cx = 1 0 0 ! (100) plane ("[100] direction")
miller-direction-of-cy = 0 1 0 ! (010) plane ("[010] direction")
miller-direction-of-cz = 0 0 1 ! (000) plane ("[001] direction")
```

These are the default orientations.

**lattice-constants-for-cxyz**

**type**  
double\_array

**value**  
1.0 1.0 1.0

Lattice constants to interpret the Miller indices.

Define how the bands are labeled.

**gamma-is-cb-number**

**type**  
integer

**value**  
1

The conduction band minimum at the *Gamma* point has the number 1.

**l-is-cb-number**

**type**  
integer

**value**  
2

The conduction band minimum at the *L* point has the number 2.

**x-is-cb-number**

**type**  
integer

**value**  
3

The conduction band minimum at the *X* point (or *Delta* in Si or Ge) has the number 3.

Example

```
!-----!
$zb-restrictions
miller-size = 3
miller-default-direction-of-x = 1 0 0
miller-default-direction-of-y = 0 1 0
direction-cosines = 0.0 0.0 0.0 ! g1*g2, g2*g3, g1*g3 gi
→... unit vectors in lattice directions
```

(continues on next page)

(continued from previous page)

```

miller-direction-of-cx      = 1 0 0
miller-direction-of-cy      = 0 1 0
miller-direction-of-cz      = 0 0 1
lattice-constants-for-cxyz  = 1.0  1.0  1.0
gamma-is-cb-number          = 1
l-is-cb-number              = 2
x-is-cb-number              = 3
$end_zb-restrictions
!-----!

```

### 4.2.30 \$wz-restrictions

Some restrictions apply for wurtzite materials.

|                               |                      |                 |
|-------------------------------|----------------------|-----------------|
| <b>\$wz-restrictions</b>      |                      | <b>required</b> |
| miller-size                   | <b>integer</b>       | <b>required</b> |
| miller-default-direction-of-x | <b>integer_array</b> | <b>required</b> |
| miller-default-direction-of-y | <b>integer_array</b> | <b>required</b> |
| direction-cosines             | <b>double_array</b>  | <b>required</b> |
| miller-direction-of-cx        | <b>integer_array</b> | <b>required</b> |
| miller-direction-of-cy        | <b>integer_array</b> | <b>required</b> |
| miller-direction-of-cz        | <b>integer_array</b> | <b>required</b> |
| lattice-constants-for-cxyz    | <b>double_array</b>  | <b>required</b> |
| <b>\$end_wz-restrictions</b>  |                      | <b>required</b> |

Explanations

#### miller-size

##### type

integer

##### presence

required

##### value

4

There are four Miller-Bravais indices altogether that define the (hkil) **plane**.

#### Note

They do not define the [hkil] direction.

Usually for wurtzite, the four-digit Miller-Bravais indices (h k i l) are used.

#### miller-default-direction-of-x

##### type

integer\_array

##### presence

required

Four-digit Miller indices of the (hkil) plane perpendicular to x axis of simulation coordinate system.

#### miller-default-direction-of-y

##### type

integer\_array

**presence**  
required

Four-digit Miller indices of the (hkil) plane perpendicular to y axis of simulation coordinate system.

```

miller-default-direction-of-x = 1 0 -1 0 ! ( 1 0 -1 0) plane
miller-default-direction-of-y = -1 2 -1 0 ! (-1 2 -1 0) plane

```

This corresponds to x axis and y axis, respectively, in simulation coordinate system, i.e. the x axis is perpendicular to the ( 1 0 -1 0) plane and the y axis is perpendicular to the (-1 2 -1 0) plane, respectively.

These value can be overwritten in *\$domain-coordinates* (hkil-x-direction, hkil-y-direction, hkil-z-direction).

#### **Note**

It holds for the four-digit Miller-Bravais indices (h k i l):  $i = -(h + k)$ , i.e. i is not independent.

Direction cosines

**direction-cosines**

**type**  
double\_array

**presence**  
required

**value**  
-0.5 0.0 0.0

Direction cosines between lattice vectors.  $g_1 \cdot g_2$ ,  $g_2 \cdot g_3$ ,  $g_1 \cdot g_3$   $g_i \dots$  unit vectors in lattice directions.

Direction cosine refers to the cosine of the angle between any two vectors. Direction cosines are useful for forming direction cosine matrices that express one set of orthonormal basis vectors in terms of another set, or for expressing a known vector in a different basis. For wurtzite, we use:

```

g_ik = ( 1  -0.5  0 )
      ( -0.5  1   0 )
      ( 0   0   1 )

```

Four-digit Miller indices of the (hkil) plane:

**miller-direction-of-cx**

**type**  
integer\_array

**value**  
1 0 -1 0

Corresponds to x axis in cartesian crystal coordinate system, i.e. the x axis is perpendicular to the ( 1 0 -1 0) plane.

**miller-direction-of-cy**

**type**  
integer\_array

**value**  
-1 2 -1 0

Corresponds to y axis in cartesian crystal coordinate system, i.e. the y axis is perpendicular to the (-1 2 -1 0) plane.

**miller-direction-of-cz**

**type**  
integer\_array

**value**

0 0 0 1

Corresponds to z axis in cartesian crystal coordinate system, i.e. the z axis is perpendicular to the (0 0 0 1) plane, i.e. axis parallel to sixfold rotational axis in wurtzite which is coincidentally also the [0001] direction.

```

miller-direction-of-cx = 1 0 -1 0 ! (10-10) plane
miller-direction-of-cy = -1 2 -1 0 ! (-12-10) plane
miller-direction-of-cz = 0 0 0 1 ! (0001) plane ("[0001] direction")

```

These are the default orientations.

**lattice-constants-for-cxyz****type**

double\_array

**value**

1.0 1.0 1.6329931618554520654648560498039

Lattice constants to interpret the Miller-Bravais indices: 1.0, 1.0,  $\sqrt{8/3}$ . Here, we take the ideal wurtzite ratio of  $c/a = \sqrt{8/3}$ .

In wurtzite, there are three coordinate axis in the basal plane,  $\mathbf{a}_1$ ,  $\mathbf{a}_2$ ,  $\mathbf{a}_3$ , and the  $\mathbf{c}$  direction perpendicular to it. There are different definitions for it.

- $\mathbf{a}_1 = [10-10]$ ,  $\mathbf{a}_2 = [-12-10]$ ,  $\mathbf{a}_3 = [\dots]$ ,  $\mathbf{c} = [0001]$  (used by *nextnano<sup>3</sup>*)  $\Rightarrow \mathbf{a}_1 = \sqrt{3}/2ax - a/2y$ ,  $\mathbf{a}_2 = ay$ ,  $\mathbf{c} = cz$
- $\mathbf{a}_1 = [2-1-10]$ ,  $\mathbf{a}_2 = [-12-10]$ ,  $\mathbf{a}_3 = [-1-120]$ ,  $\mathbf{c} = [0001]$   $\Rightarrow \mathbf{a}_1 = [2-1-10] a/\sqrt{6}$ ,  $\mathbf{a}_2 = [-12-10] a/\sqrt{6}$ ,  $\mathbf{c} = [0001] c = [0, 0, 0, 3 \lambda] a/\sqrt{6}$ , where  $\lambda = \sqrt{2/3}c/a$ .

Example

```

!-----!
$wz-restrictions
miller-size = 4
miller-default-direction-of-x = 1 0 -1 0
miller-default-direction-of-y = -1 2 -1 0
direction-cosines = -0.5 0.0 0.0 ! g1*g2, g2*g3, g1*g3 gi
→... unit vectors in lattice directions
miller-direction-of-cx = 1 0 -1 0
miller-direction-of-cy = -1 2 -1 0
miller-direction-of-cz = 0 0 0 1
lattice-constants-for-cxyz = 1.0 1.0 1.
→6329931618554520654648560498039
$end_wz-restrictions
!-----!

```

#### 4.2.31 \$region-default

Default priority in regions.

|                             |                |                 |
|-----------------------------|----------------|-----------------|
| <b>\$region-default</b>     |                | <b>required</b> |
| minimum-priority            | <b>integer</b> | <b>required</b> |
| <b>\$end_region-default</b> |                | <b>required</b> |

Minimum priority.

**minimum-priority****type**

integer

```
presence
    required
value
    0
```

Change only with care. 0 is a good idea. A negative value might also work.

Example

```
!-----!
$region-default
    minimum-priority = 0
$end_region-default
!-----!
```

### 4.2.32 \$known-doping-functions

Several doping functions are possible. For more information see *\$doping-function*.

|                                     |                  |                 |
|-------------------------------------|------------------|-----------------|
| <b>\$known-doping-functions</b>     |                  | <b>required</b> |
| doping-function-number              | <b>integer</b>   | <b>required</b> |
| doping-function-name                | <b>character</b> | <b>required</b> |
| <b>\$end_known-doping-functions</b> |                  | <b>required</b> |

Example

```
!-----!
$known-doping-functions

doping-function-number = 1
doping-function-name   = constant    ! constant doping profile

doping-function-number = 2
doping-function-name   = gauss-1d    ! Gaussian shaped doping profile

doping-function-number = 3
doping-function-name   = step-1d

doping-function-number = 4
doping-function-name   = well-1d

doping-function-number = 5
doping-function-name   = linear      ! linear doping profile

$end_known-doping-functions
!-----!
```

### 4.2.33 \$known-function-names

Several functions are possible. For more information see *\$alloy-function*.

|                                   |                  |                 |
|-----------------------------------|------------------|-----------------|
| <b>\$known-function-names</b>     |                  | <b>required</b> |
| function-number                   | <b>integer</b>   | <b>required</b> |
| function-name                     | <b>character</b> | <b>required</b> |
| <b>\$end_known-function-names</b> |                  | <b>required</b> |

Alloy functions.

```
function-number
```

**type**  
integer

**presence**  
required

**value**  
1, 2, ...

Dense numbering of integers, starting from 1, 2, 3, ...

#### **function-name**

**type**  
character

**presence**  
required

**value**  
constant, linear, import-alloy-profile,  
alloy-profile-defined-by-function, ... (see below)

Name of alloy profile.

Example

```
!-----!
$known-function-names
function-number = 1
function-name   = gaussian-3d

function-number = 2
function-name   = gaussian-2d

function-number = 3
function-name   = gaussian-1d

function-number = 4
function-name   = one-m-gaussian-3d

function-number = 5
function-name   = one-m-gaussian-2d

function-number = 6
function-name   = one-m-gaussian-1d

function-number = 7
function-name   = constant

function-number = 8
function-name   = linear

function-number = 9
function-name   = bilinear

function-number = 10
function-name   = xy-gaussian-3d

function-number = 11
function-name   = xy-gaussian-2d
```

(continues on next page)

(continued from previous page)

```

function-number = 12
function-name   = xy-gaussian-1d

function-number = 13
function-name   = xy-one-m-gaussian-3d

function-number = 14
function-name   = xy-one-m-gaussian-2d

function-number = 15
function-name   = xy-one-m-gaussian-1d

function-number = 16
function-name   = xy-constant

function-number = 17
function-name   = xy-linear

function-number = 18
function-name   = xy-bilinear

function-number = 19
function-name   = inv-triangle

function-number = 20
function-name   = trumpet

function-number = 21
function-name   = parabolic

function-number = 22
function-name   = biparabolic

function-number = 23
function-name   = triparabolic

function-number = 24
function-name   = Fermi-function

function-number = 25
function-name   = onion

function-number = 26
function-name   = import-alloy-profile

function-number = 27
function-name   = alloy-profile-defined-by-function

$end_known-function-names
!-----!

```

#### 4.2.34 \$known-impurity-types

Several impurity types are possible.



|                                   |                  |                 |
|-----------------------------------|------------------|-----------------|
| <b>\$known-impurity-types</b>     |                  | <b>required</b> |
| impurity-type-name                | <b>character</b> | <b>required</b> |
| impurity-type-number              | <b>integer</b>   | <b>required</b> |
| <b>\$end_known-impurity-types</b> |                  | <b>required</b> |

Example

```

!-----!
$known-impurity-types
→ !

impurity-type-name = n-type
impurity-type-number = 1

impurity-type-name = p-type
impurity-type-number = 2

impurity-type-name = trap
impurity-type-number = 3

$end_known-impurity-types
→ !
!-----!

```

#### 4.2.35 \$interface-state-limitations

Several doping functions are possible. For more information see *\$doping-function*.

|                                          |                  |                 |
|------------------------------------------|------------------|-----------------|
| <b>\$interface-state-limitations</b>     |                  | <b>required</b> |
| known-state-type-name                    | <b>character</b> | <b>required</b> |
| <b>\$end_interface-state-limitations</b> |                  | <b>required</b> |

Example

```

!-----!
$interface-state-limitations
known-state-type-name = trap      ! charge density due to trap states
known-state-type-name = fixed-charge ! fixed charge density
known-state-type-name = electrolyte ! charge density due to electrolyte
→ (site-binding model)
known-state-type-name = gas       ! charge density due to gas
→ adsorption
known-state-type-name = k.p       ! interface k.p Hamiltonian
$end_interface-state-limitations
!-----!

```

#### 4.2.36 \$quantum-model-electrons

Several quantum models for the electrons are possible.

|                                       |                  |                 |
|---------------------------------------|------------------|-----------------|
| <b>\$quantum-models-electrons</b>     |                  | <b>required</b> |
| model-name                            | <b>character</b> | <b>required</b> |
| model-type-number                     | <b>integer</b>   | <b>required</b> |
| <b>\$end_quantum-models-electrons</b> |                  | <b>required</b> |

Example

```
!-----!  
$quantum-models-electrons  
model-name      = effective-mass  
model-type-number = 1  
  
model-name      = 8x8kp  
model-type-number = 2  
$end_quantum-models-electrons  
!-----!
```

#### 4.2.37 \$quantum-model-holes

Several quantum models for the holes are possible.

|                            |           |          |
|----------------------------|-----------|----------|
| \$quantum-models-holes     |           | required |
| model-name                 | character | required |
| model-type-number          | integer   | required |
| \$end_quantum-models-holes |           | required |

Example

```
!-----!  
$quantum-models-holes  
model-name      = effective-mass  
model-type-number = 1  
  
model-name      = 8x8kp  
model-type-number = 2  
  
model-name      = 6x6kp  
model-type-number = 3  
$end_quantum-models-holes  
!-----!
```

#### 4.2.38 \$separation-models-electrons

More details about the separation model can be found in the [glossary](#).

|                                   |           |          |
|-----------------------------------|-----------|----------|
| \$separation-models-electrons     |           | required |
| model-name                        | character | required |
| model-type-number                 | integer   | required |
| \$end_separation-models-electrons |           | required |

Example

```
!-----!  
$separation-models-electrons  
model-name      = eigenvalue  
model-type-number = 1  
  
model-name      = energy  
model-type-number = 2  
  
model-name      = edge-model  
model-type-number = 3  
$end_separation-models-electrons  
!-----!
```

### 4.2.39 \$separation-models-holes

More details about the separation model can be found in the [glossary](#).

|                                      |                  |                 |
|--------------------------------------|------------------|-----------------|
| <b>\$separation-models-holes</b>     |                  | <b>required</b> |
| model-name                           | <b>character</b> | <b>required</b> |
| model-type-number                    | <b>integer</b>   | <b>required</b> |
| <b>\$end_separation-models-holes</b> |                  | <b>required</b> |

Example

```
!-----!
$separation-models-holes
model-name      = eigenvalue
model-type-number = 1

model-name      = energy
model-type-number = 2

model-name      = edge-model
model-type-number = 3
$end_separation-models-holes
!-----!
```

### 4.2.40 \$method-of-brillouin-zone-integration

More details about the method of Brillouin zone integration can be found in the [glossary](#).

|                                                   |                  |                 |
|---------------------------------------------------|------------------|-----------------|
| <b>\$method-of-brillouin-zone-integration</b>     |                  | <b>required</b> |
| model-name                                        | <b>character</b> | <b>required</b> |
| model-type-number                                 | <b>integer</b>   | <b>required</b> |
| <b>\$end_method-of-brillouin-zone-integration</b> |                  | <b>required</b> |

Example

```
!-----!
$method-of-brillouin-zone-integration
model-name      = special-axis
model-type-number = 1

model-name      = simple-integration
model-type-number = 2

model-name      = gen-dos
model-type-number = 3
$end_method-of-brillouin-zone-integration
!-----!
```

### 4.2.41 \$k-range-determination-methods

This makes only sense for  $\mathbf{k} \cdot \mathbf{p}$  calculations in 1D ( $k_{\parallel} = (k_x, k_y)$ ) and 2D ( $k_{\parallel} = (k_z)$ ) but not in 3D.

1. Solve Schrödinger equation for  $(k_x, k_y) = (0, 0)$ .
2. Define a set of  $k_{\parallel}$  that one needs and solve  $\mathbf{k} \cdot \mathbf{p}$  Schrödinger equation for every  $k_{\parallel}$ .

|                                            |                  |                 |
|--------------------------------------------|------------------|-----------------|
| <b>\$k-range-determination-methods</b>     |                  | <b>required</b> |
| model-name                                 | <b>character</b> | <b>required</b> |
| model-type-number                          | <b>integer</b>   | <b>required</b> |
| <b>\$end_k-range-determination-methods</b> |                  |                 |

Two models are supported.

```

model-name
    value
        bulk-dispersion-analysis

model-type-number
    value
        1

```

Here, the range for  $k_{||}$  is determined automatically by the program using the bulk energy dispersion  $E(k)$ . [More information...](#)

```

model-name
    value
        k-max-input

model-type-number
    value
        2

```

A maximum value  $k_{\max}$  of  $k_{||}$  has to be specified in the input file.

Example

```

!-----!
$k-range-determination-methods
  model-name      = bulk-dispersion-analysis
  model-type-number = 1

  model-name      = k-max-input
  model-type-number = 2
$end_k-range-determination-methods
!-----!

```

#### 4.2.42 \$tight-binding-parameters

This part is preliminary. It is not working yet.

```

$tight-binding-parameters                                optional !

  material-name                                             character    required !

  ! basis
  number-of-ions                                           integer      required !
  name-ion-1                                              character    required !
  name-ion-2                                              character    optional !

  ! states
  spin-orbit-ion-1-p                                       double      required ! ion 1 spin
  ↪orbit (p)
  spin-orbit-ion-1-d                                       double      required ! ion 1 spin
  ↪orbit (d)
  spin-orbit-ion-2-p                                       double      optional ! ion 2 spin
  ↪orbit (p)
  spin-orbit-ion-2-d                                       double      optional ! ion 2 spin
  ↪orbit (d)

  ! interaction between s and p orbitals

```

(continues on next page)

(continued from previous page)

```

! interaction between s and d orbitals (in our case: zero)

number-of-orbitals-per-ion-1      integer      required !
number-of-orbitals-per-ion-2      integer      optional !
onsite-ion-1-s                    double        required ! ion 1_
↪ onsite (s) [eV]
onsite-ion-1-p                    double        required ! ion 1_
↪ onsite (p) [eV]
onsite-ion-1-s*                   double        required ! ion 1_
↪ onsite (s*) [eV]
onsite-ion-1-d                    double        required ! ion 1_
↪ onsite (d) [eV]
onsite-ion-2-s                    double        optional ! ion 2_
↪ onsite (s) [eV]
onsite-ion-2-p                    double        optional ! ion 2_
↪ onsite (p) [eV]
onsite-ion-2-s*                   double        optional ! ion 2_
↪ onsite (s*) [eV]
onsite-ion-2-d                    double        optional ! ion 2_
↪ onsite (d) [eV]
offset-ion-1                      double        required ! ion 1 [eV]
offset-ion-2                      double        optional ! ion 2 [eV]

! Tight-binding couplings
number-of-pairs                   integer      required !
pair-1-ion-1                      character     required !
pair-1-ion-2                      character     optional !
pair-1-number-of-parameters       integer      required !
pair-1-nearest-neighbors          integer      required !
ion-ion-coupling-1               character     required !
ion-ion-coupling-1-sss            double_array required ! [eV],[]
ion-ion-coupling-1-sps            double_array required ! [eV],[]
ion-ion-coupling-1-ss*s           double_array required ! [eV],[]
ion-ion-coupling-1-sds            double_array required ! [eV],[]
ion-ion-coupling-1-pss            double_array required ! [eV],[]
ion-ion-coupling-1-pps            double_array required ! [eV],[]
ion-ion-coupling-1-ps*s           double_array required ! [eV],[]
ion-ion-coupling-1-pds            double_array required ! [eV],[]
ion-ion-coupling-1-s*ss           double_array required ! [eV],[]
ion-ion-coupling-1-s*ps           double_array required ! [eV],[]
ion-ion-coupling-1-s*s*s           double_array required ! [eV],[]
ion-ion-coupling-1-s*ds           double_array required ! [eV],[]
ion-ion-coupling-1-dss            double_array required ! [eV],[]
ion-ion-coupling-1-dps            double_array required ! [eV],[]
ion-ion-coupling-1-ds*s           double_array required ! [eV],[]
ion-ion-coupling-1-dds            double_array required ! [eV],[]
ion-ion-coupling-1-ppp            double_array required ! [eV],[]
ion-ion-coupling-1-pdp            double_array required ! [eV],[]
ion-ion-coupling-1-dpp            double_array required ! [eV],[]
ion-ion-coupling-1-ddp            double_array required ! [eV],[]
ion-ion-coupling-1-ddd            double_array required ! [eV],[]

ion-ion-pair-1-reference-distance-nn double      required ! [nm]

$end_tight-binding-parameters      optional !

```

Example

```

!-----!
$tight-binding-parameters

!-----
! Si tight-binding parameters (c) Jean Marc Jancu
!-----

material-name                = Si

! basis
number-of-ions               = 1          !
name-ion-1                   = Si

! states
spin-orbit-ion-1-p           = 0.0195    ! Si spin orbit (p)
spin-orbit-ion-1-d           = 0.0000    ! Si spin orbit (d)

! interaction between s and p orbitals
! interaction between s and d orbitals (in our case: zero)

number-of-orbitals-per-ion-1 = 4          ! Si
onsite-ion-1-s               = -2.0196    ! Si onsite (s)
onsite-ion-1-p               = 4.5448     ! Si onsite (p)
onsite-ion-1-s*              = 19.6748    ! Si onsite (s*)
onsite-ion-1-d               = 14.1836    ! Si onsite (d)
offset-ion-1                 = 0.0        ! Si eV

! Tight-binding couplings
number-of-pairs              = 1
pair-1-ion-1                 = Si
pair-1-ion-2                 = Si
pair-1-number-of-parameters = 21
pair-1-nearest-neighbors    = 1
ion-ion-coupling-1           = 'sss sps ss*s sds pss pps ps*s pds_
↪s*s ss s*ps s*s*s s*ds dss dps ds*s dds ppp pdp dpp ddp ddd'
ion-ion-coupling-1-sss       = -1.9413    3.672d0
ion-ion-coupling-1-sps       = 2.7836    2.488d0
ion-ion-coupling-1-ss*s      = -1.6933    0.000d0
ion-ion-coupling-1-sds       = -2.7998    1.869d0
ion-ion-coupling-1-pss       = 2.7836    2.488d0
ion-ion-coupling-1-pps       = 4.1068    2.187d0
ion-ion-coupling-1-ps*s      = 2.8428    1.919d0
ion-ion-coupling-1-pds       = -2.1073    1.830d0
ion-ion-coupling-1-s*ss      = -1.6933    0.000d0
ion-ion-coupling-1-s*ps      = 2.8428    1.919d0
ion-ion-coupling-1-s*s*s      = -3.3081    0.000d0
ion-ion-coupling-1-s*ds      = -0.7003    2.000d0
ion-ion-coupling-1-dss       = -2.7998    1.869d0
ion-ion-coupling-1-dps       = -2.1073    1.830d0
ion-ion-coupling-1-ds*s      = -0.7003    2.000d0
ion-ion-coupling-1-dds       = -1.2327    2.000d0
ion-ion-coupling-1-ppp       = -1.5934    3.711d0
ion-ion-coupling-1-pdp       = 1.9977    2.093d0
ion-ion-coupling-1-dpp       = 1.9977    2.093d0
ion-ion-coupling-1-ddp       = 2.5145    2.000d0
ion-ion-coupling-1-ddd       = -2.4734    2.000d0

```

(continues on next page)

(continued from previous page)

```

ion-ion-pair-1-reference-distance-nn = 2.351259e-10 ! m -> A ! do not
↪change! very sensitive

$end_tight-binding-parameters
!-----!

```

#### 4.2.43 \$warnings

Warnings can be switched on or off.

|                       |         |                 |
|-----------------------|---------|-----------------|
| <b>\$warnings</b>     |         | <b>required</b> |
| warnings              | logical | required        |
| <b>\$end_warnings</b> |         | <b>required</b> |

Specify here if warnings should be written to the .log file.

**warnings**

**type**

logical

**presence**

required

**options**

.FALSE.

.TRUE.

**default**

uses value specified in database\_nn3.in

Example

```

!-----!
$warnings                                     !
  warnings = .FALSE.                         ! switch off warnings
!warnings = .TRUE.                          ! switch on warnings
$end_warnings                               !
!-----!

```





## INPUT SYNTAX

### 5.1 Macro features

The Macro feature of *nextnano*<sup>3</sup> is also described here: <https://github.com/nextnano/FortranInputParser>. The Fortran Input Parser source code is open-source.

#### Comment signs

Everything in the *same* line following after such a symbol is treated as a *comment*.

- #

Note: This is the same comment sign as used in *nextnano++*.

- !

- //

- /\* This is a comment. \*/

The end tag `*/` is optional.

Note: No line breaks are allowed for the comments `/* ... */`.

- < This is a comment. >

#### Note

However, the following are exceptions and are treated differently.

- #IF
- !IF
- #WHEN
- !WHEN

These are special statements for conditional lines (see below).

#### Note

XML tags can be used in the input file, .e.g. to provide additional meta information. They are ignored by *nextnano*<sup>3</sup> but could be used in e.g. *nextnanomat*. Everything that is between these brackets `< ... >` is replaced with blanks. This can be used to define XML tags such as:

```
<variables>
# comment
%TEMPERATURE = 300.0                # (DisplayUnit:K)
%WIDTH       = 10.0                # (DisplayUnit:nm)
%HEIGHT      = 20.0 < unit = "nm" > # (DisplayUnit:nm)
</variables>
<variables> %TEMPERATURE = 300.0 </variables> # (DisplayUnit:K)
```

### 5.1.1 Macro 1: Variables

The % symbol is used to define a variable such as %VARIABLE = that should be replaced with another string or value. Variables are case-sensitive.

A variable is only defined and initialized if the first character in a line starts with the % symbol.

This macro distinguishes between *strings* (default), i.e. simple string replacements where 1 and 1.0 are treated as a *string* and *functions* (i.e. arithmetic operations if %FunctionParser = yes) where both 1 or 1.0 are treated as a *real* number. Both can be combined but it is important to understand the differences to avoid errors. For instance, for some *specifiers*, the input parser expects an *integer* number and not a *real* number. So real numbers have to be converted back to integers in some cases. This macro is supported by *nextnanomat*.

#### String feature

*Find & Replace* of variables

Example: %WIDTH = 10.0

The *nextnano<sup>3</sup>* tool supports a simple *Find & Replace* feature where *strings* in the input file can be replaced with another *string*.

#### How to use it

Just put something like the following lines anywhere into your input file. We recommend that you put it at the beginning of the input file for better readability.

```
#-----
# By default, we do not evaluate functions.
# Everything is just a simple "Find & Replace String" operation.
#-----
%width          = 10.0          # quantum well width_
↪(DisplayUnit:nm)
%length         = 10           # quantum well length_
↪(DisplayUnit:nm)
%size           = 10.0         # quantum well size (DisplayUnit:nm)
%z_coordinates  = -21.0 101.0  # from z = -21 to z = 101_
↪(DisplayUnit:nm)
%well_material   = GaAs        # quantum well material: GaAs
%barrier_material = "AlAs"     # quantum barrier material: AlAs
%zmin           = -10.0        # zmin = - 10 nm (DisplayUnit:nm)
%zmax           = 10.0         # zmax = 10 nm (DisplayUnit:nm)

# derived variable: quantum_region = zmin zmax = -10.0 10.0
%quantum_region  = %zmin %zmax  # (DisplayUnit:nm)

%hkl_z_direction = 0 1 1       # [0 1 1] growth direction

# The whole string including blanks after the first '=' sign is used.
%Region1         = region-number = 1 base-geometry = line region-
↪priority = 1
```

The % sign indicates that you define a *variable*. When the input file is processed, all occurrences of %width, %z-coordinates, %well\_material, ... are replaced by the according strings.

A *comment* sign (e.g. #, !) is allowed in this line. Blanks are allowed within the string which is useful for reading in arrays of numbers. Almost everything is allowed.

#### Restrictions

- The variables and their definitions are *case-sensitive*.

- A variable name must not contain a - sign, e.g.

```
%cb-mass = 0.067 # (DisplayUnit:m0)
```

is not allowed. Use underscores \_ instead.

- In one line, only *one* variable can be initialized.

## Function feature

*Find, Evaluate & Replace*

Example: %WIDTH = %LENGTH / 2.0

The statement %FunctionParser = yes switches the function parser on (default: off).

The following operators

|                                     |     |
|-------------------------------------|-----|
| round arithmetic brackets           | ( ) |
| power (exponentiation)              | ^   |
| arithmetic multiplication, division | * / |
| arithmetic plus and minus           | + - |

and functions are supported:

|         |                                          |
|---------|------------------------------------------|
| sqrt()  | square root $\sqrt{\phantom{x}}$         |
| exp()   | exponential function $\exp(\phantom{x})$ |
| log()   | natural logarithm $\log$                 |
| log10() | decadic logarithm (base 10) $\log_{10}$  |
| sin()   | sine $\sin(\phantom{x})$                 |
| cos()   | cosine $\cos(\phantom{x})$               |
| tan()   | tangent $\tan(\phantom{x})$              |
| asin()  | acrsine $\sin^{-1}(\phantom{x})$         |
| acos()  | arccosine $\cos^{-1}(\phantom{x})$       |
| atan()  | arctangent $\tan^{-1}(\phantom{x})$      |
| sinh()  | hyperbolic sine $\sinh(\phantom{x})$     |
| cosh()  | hyperbolic cosine $\cosh(\phantom{x})$   |
| tanh()  | hyperbolic tangent $\tanh(\phantom{x})$  |
| abs()   | absolute value $  $                      |

Example

```
#-----
# If '%FunctionParser = yes', then from now on we evaluate all functions.
# If '%FunctionParser = no' (or not present), no functions are evaluated.
# "Find & Evaluate & Replace String" operation.
#-----

%FunctionParser = yes # This is an important line!

%a      = 4.0          # a = 4.0
%b      = 3            # b = 3
%function1 = Sqrt(%a) - %b / 2 # Evaluate: sqrt(a) - b / 2 = 2.0 - 1.5 = 0.5
↪5
...
%xalloy = %function1    # Insert evaluated value for %function1.

# Strings containing blanks are also possible.
```

(continues on next page)

(continued from previous page)

```
# They must be in quotation marks "" or apostrophes '.
%string1 = "GaAs"
%string2 = "GaAs"
%string3 = "zero-potential = yes"

# Logical variables that are .TRUE. or .FALSE. can be defined
# and used in conditional #IF statements.
%Logical1 = .TRUE.
%Logical2 = .FALSE.
```

We have to be careful about the value of a variable. Should it be regarded as a *string* (default), or as a *function* (%FunctionParser = yes)?

```
#-----
#-
# Treat variables as strings (default):
#-----
#   e.g. %variable1 = %zmin %zmax # '%zmin %zmax' is treated as a string
#       %variable2 = 2.5e0        # '2.5e0' is treated as a string
#       %variable3 = 2.5          # '2.5' is treated as a string
#-without exponent 'e0'
#-----
#-
#%radius      = 5e0                # quantum dot radius = 5 nm
#(DisplayUnit:nm)
%grid_spacing = 0.5                # grid spacing
#(DisplayUnit:nm)
#-----
#-
# From now on, evaluate all functions:
#-----
#   e.g. %variable1 = %zmin %zmax # '%zmin %zmax' is treated as a formula
#                               # (which will produce an error in this
#-case)
#       %variable2 = 2.5          # '2.5' is converted to a real
#-number including exponent 'e0'
#-----
#-
%FunctionParser = yes              # use function parser
%radius          = 5               # quantum dot radius = 5 nm
#(DisplayUnit:nm)
%boundary        = %radius + 1     # device_width/2 = 5 nm + 1 nm
#(DisplayUnit:nm)
```

### Special treatment of integers

The parser distinguishes between integers and real numbers. Thus we have to be careful when using functions. This means that we have to convert the real values back to integers. Integers will be rounded down up  $\geq .5$  or rounded down if  $< .5$ , i.e. they will be rounded to the nearest integer.

- Option a) (recommended)

```
#-----
# If you need an integer, then use INT(...).
# %nodes contains an integer value.
#-----
%nodes = INT( %width / %grid_spacing - 1 )
```

- Option b) (deprecated)

```
#-----
# If you need an integer, then simply call the
# variable '%INT(<name>)'
# %INT(nodes) contains an integer value.
#-----
%INT(nodes) = %width / %grid_spacing - 1
```

- Option c) (deprecated)

```
#-----
# If you need an integer, then simply call the
# variable '%INT(<name>)'
# %nodes is real number.
# %INT(nodes) contains an integer value.
#-----
%nodes = %width / %grid_spacing - 1
%INT(nodes) = %nodes
```

Integer numbers can be defined using quotation marks `"`´`"` or `INT()`:

```
%NumberOfEigenvalues = "10"      # 10 eigenvalues
%NumberOfEigenvalues = INT(10)    # 10 eigenvalues
```

Integer arrays can be defined using quotation marks `"`´`"`:

```
%hkl_z_direction = "0 1 1"      # [0 1 1] growth direction
```

## conditional lines

`#IF`, `!IF`, `#WHEN` or `!WHEN` statements can also be used to define conditional lines, i.e. variables can be used to define conditional lines.

`#IF`, `!IF`, `#WHEN` or `!WHEN` are not *case-sensitive*, i.e.

`#if`, `!if`, `#when` or `!when` also work.

```
#IF %TemperatureDependentBandGap varshni-parameters-on = yes
```

Such a statement has the following meaning:

- If the value of `%TemperatureDependentBandGap` is either `.TRUE.` or `1` or any other *nonzero value*, then the statement is *included* in the input file.
- If the value of `%TemperatureDependentBandGap` is either `.FALSE.` or `0` or *undefined*, then the statement is simply *ignored* as if it were a comment.

In this example, the text is always commented out, unless `%IncludeHoles` is defined with a value `%IncludeHoles /= 0` or `/.FALSE.`.

```
%IncludeHoles = .TRUE.      # or %IncludeHoles = 1

!IF %IncludeHoles $quantum-model-holes
!IF %IncludeHoles model-number           = 1
!IF %IncludeHoles model-name             = effective-mass
!IF %IncludeHoles cluster-numbers        = 1
!IF %IncludeHoles valence-band-numbers    = 1 2 3
!IF %IncludeHoles number-of-eigenvalues-per-band = 10
!IF %IncludeHoles $end_quantum-model-holes
```

In the following, the evaluated value of %IncludeElectrons is 0, therefore the three lines starting with !IF %IncludeElectrons are treated as comments.

```
%IncludeHoles      = 1
%IncludeElectrons = 1 - %IncludeHoles  # evaluated to: 1 - 1 = 0
!IF %IncludeElectrons $quantum-model-electrons # treated as comment
!IF %IncludeElectrons model-number = 1        # treated as comment
!IF %IncludeElectrons ...                     # treated as comment
```

If the variable definition

```
%IncludeHoles = .TRUE.
```

appears after the line

```
%FunctionParser = yes
```

quotation marks " or apostrophes ' must be used for the logical strings .TRUE./.FALSE. to indicate that they have to be treated as strings:

```
%IncludeHoles = ".TRUE." # or
%IncludeHoles = '.TRUE.'
```

### Variable replacement

- Once the input file has been processed, the input file (\*.in) is copied to the output folder. Additionally, this output folder contains a file called \*.in.mo\_macro which contains the input file with all variables having been replaced with their values.
- All variables and their values that are used in a simulation are written to the output folder into a file called variables\_input.txt.

### 5.1.2 Macro 2: Namelist (deprecated)

Alternatively, namelists are supported for macros. The input syntax for this macro is implemented as a NAMELIST which is a standard Fortran feature.

The nextnano<sup>3</sup> tool supports a simple *Find & Replace* feature where strings in the input file can be replaced with another string. The first lines of the input file can contain the following additional entries.

```
&macro filename = 'macro.in' /
```

Here, macro.in is an ASCII file containing the following content:

```
&macro Find_and_Replace_1 = '$WIDTH' , '0d0 10.0d0'
Find_and_Replace_2 = 'x-nodes = 9' , 'x-nodes = 29'
Find_and_Replace_3 = 'GaAs' , 'InP'
/
```

These three lines have the following meaning:

- Find \$WIDTH and replace it with 0d0 10.0d0.
- Find x-nodes = 9 and replace it with x-nodes = 29.
- Find GaAs and replace it with InP.

Download example file: [macro.in](#)

Example 1

```

!
! *****
!
! Macro: 'Find & Replace'
!
! *****
!
!
! A macro can be used to simply search for a specific string and to replace
! it with another string.
!
! Example: Replace the material 'GaAs' with 'InAs'.
!     material-name = GaAs
!     ==> material-name = InAs
!
! Replace the quantum well width '$WIDTH' with '10d0':
!     x-coordinates = -$WIDTH $WIDTH
!     ==> x-coordinates = -10d0 10d0
!
! Replace the string '2*$WIDTH' with '20d0':      ! (Mathematical operations
! are not supported yet. They are treated as strings.)
!     x-coordinates = 0d0 2*$WIDTH
!     ==> x-coordinates = 0d0 20d0
!
! Replace the string '$WIDTH+$SIZE' with '30d0': ! (Mathematical operations
! are not supported yet. They are treated as strings.)
!     x-coordinates = 0d0 $WIDTH+$SIZE
!     ==> x-coordinates = 0d0 30d0
!
! Replace the quantum well width '$WIDTH' with '0d0 10d0':
!     x-coordinates = $WIDTH
!     ==> x-coordinates = 0d0 10d0
!
! Simulate along x, y or z direction:
!     Find_and_Replace_1 = '$DIRECTION' , 'x'      ! simulation along x
! direction
!     Find_and_Replace_2 = '$DOMAINTYPE', '1 0 0'   ! simulation along x
! direction
!
!     Find_and_Replace_1 = '$DIRECTION' , 'y'      ! simulation along y
! direction
!     Find_and_Replace_2 = '$DOMAINTYPE', '0 1 0'   ! simulation along y
! direction
!
! a) The macro can be defined in the input file: filename = '' (or)
! b) The macro can be read in from another file: filename = 'macro.in'
!
! -----
!
!
!&macro filename = 'macro.in' ! (optional)
!&macro file to be read in, use '' for no macro input file
!     Find_and_Replace_1 = '$WIDTH' , '0d0 10.0d0' ! is ignored if
!&macro file is read in
!     Find_and_Replace_2 = 'x-nodes = 9', 'x-nodes = 29' ! is ignored if
!&macro file is read in
!     Find_and_Replace_3 = 'GaAs' , 'InP' ! is ignored if
!&macro file is read in

```

(continues on next page)

(continued from previous page)

```
/
```

Example 2

```
&macro filename = '' ! (optional)␣
↪macro file to be read in, use '' for no macro input file
    Find_and_Replace_1 = '$WIDTH' , '0d0 10.0d0' ! find string,␣
↪replace string
    Find_and_Replace_2 = 'x-nodes = 9' , 'x-nodes = 29' ! find string,␣
↪replace string
    Find_and_Replace_3 = 'GaAs' , 'InP' ! find string,␣
↪replace string
/
```

Example 3

```
&macro Find_and_Replace_1 = '$WIDTH' , '0d0 10.0d0' ! find string,␣
↪replace string
    Find_and_Replace_2 = 'x-nodes = 9' , 'x-nodes = 29' ! find string,␣
↪replace string
    Find_and_Replace_3 = 'GaAs' , 'InP' ! find string,␣
↪replace string
    ...
    Find_and_Replace_40 = 'GaAs' , 'InP' ! find string,␣
↪replace string
/
```

Currently up to 40 *Find & Replace* definitions can be specified.

## 5.2 The input file keywords

Here you can find detailed descriptions about the keywords and specifiers for the input file. The general definition file is `keywords.val` where valid keywords are defined.

Keywords for setting up the device geometry, material and general settings

- `$input_filename`
- `$global-parameters` (temperature)
- `$simulation-dimension`
- `$simulation-flow-control`
- `$strain-minimization-model`
- `$electric-field`
- `$magnetic-field`
- `$numeric-control`

Keywords for setting up the device geometry

- `$domain-coordinates`
- `$regions`
- `$region-cluster`
- `$grid-specification`

Keywords for setting up material properties

- `$material`



- \$binary-zb-default
- \$binary-wz-default
- \$sternary-zb-default
- \$sternary-wz-default
- \$alloy-function
- \$import-data-on-material-grid

#### Keywords for doping

- \$doping-function
- \$impurity-parameters
- \$material-interfaces
- \$interface-states

#### Keywords for output

- \$global-settings (output directory)
- \$output-bandstructure
- \$output-densities
- \$output-strain
- \$output-1-band-schroedinger
- \$output-kp-data
- \$output-current-data
- \$output-raw-data
- \$output-grid
- \$output-geometry
- \$output-material
- \$output-file-format
- \$output-section

#### Keywords for quantum calculations

- \$quantum-regions
- \$quantum-cluster
- \$quantum-model-electrons
- \$quantum-model-holes
- \$quantum-bound-states
- \$quantum-dot-layer-density
- \$optical-absorption
- \$tighten

#### Keywords for current calculations

- \$poisson-boundary-conditions
- \$voltage-sweep
- \$current-regions
- \$current-cluster

- \$current-models
- \$simple-drift-models
- \$Monte-Carlo (not available yet)

Keywords for recombination

- \$Auger-recombination
- \$direct-recombination (radiative)
- \$SRH-recombination (Shockley-Read-Hall)
- \$quantumstate-recombination-rates

Keywords for mobility models

- \$mobility-model-constant
- \$mobility-model-arora
- \$mobility-model-dar (Darwish)
- \$mobility-model-lom (Lombardi)
- \$mobility-model-masetti
- \$mobility-model-minimos
- \$mobility-model-simba

Keywords for electrolytes

- \$electrolyte
- \$electrolyte-ion-content
- \$buffer-solutions
- \$buffer-constant-A(T)

Keywords for NEGF

- \$CBR-current (ballistic)
- \$global-parameters-NEGF
- \$scattering-mechanisms
- \$contact-type
- \$damping-parameters
- \$roughness-profile
- \$left-contact-potential-profile
- \$right-contact-potential-profile
- \$potential-profile (deprecated)
- \$mass-profile (deprecated)
- \$alloy-profile (deprecated)
- \$nonparabolicity-profile (deprecated)
- \$dielectric-profile (deprecated)
- \$doping-function-NEGF (deprecated)
- \$NEGF-spintronics (for NEGF spintransport code)

Other keywords

- \$warnings

The Fortran Input Parser source code of *nextnano*<sup>3</sup> is open-source: <https://github.com/nextnano/FortranInputParser>

## 5.3 Keywords

### 5.3.1 \$simulation-dimension

#### Simulation dimension and orientation

##### Simulation coordinate system

In general, the underlying coordinate system for the simulation is a three-dimensional cartesian coordinate system (x,y,z). Dependent on the type of simulation, i.e. whether a one-, two- or three-dimensional simulation has to be performed, the corresponding number of axes has to be selected out of this triplet of coordinate directions. For these specifications, the keyword `$simulation-dimension` together with its specifiers `dimension` and `orientation` have to be used in the input file.

|                                         |                      |                 |
|-----------------------------------------|----------------------|-----------------|
| <code>\$simulation-dimension</code>     |                      | <b>required</b> |
| <code>dimension</code>                  | <b>integer</b>       | <b>required</b> |
| <code>orientation</code>                | <b>integer_array</b> | <b>required</b> |
| <code>\$end_simulation-dimension</code> |                      | <b>required</b> |

Specification of the simulation domain dimension and its orientation

#### **dimension**

##### **type**

integer

##### **presence**

required

##### **options**

1 (1D)

2 (2D)

3 (3D)

Specify here if a 1D, 2D or 3D simulation is performed.

#### **orientation**

##### **type**

integer array of dimension 3

##### **presence**

required

##### **options**

for 1D simulation: 1 0 0, 0 1 0, 0 0 1

for 2D simulation: 1 1 0, 1 0 1, 0 1 1

for 3D simulation: 1 1 1

The specifier `orientation` defines the orientation of the simulation domain relative to the (x,y,z) coordinate system. A three-dimensional array containing 0 and/or 1 is expected. The relevant axes in this array are selected using 1.

Specify here if the simulation is performed along a direction, in a plane or in a volume.

- simulation direction (1D: x, y or z axis)
- simulation plane (2D: (x,y), (x,z) or (y,z) plane)
- simulation volume (3D)

Example 1 (1D)

```

!-----!
$simulation-dimension      !
dimension    = 1           ! 1D simulation
orientation  = 0 0 1       ! z axis
$end_simulation-dimension  !
!-----!

```

Example 2 (2D)

```

!-----!
$simulation-dimension      !
dimension    = 2           ! 2D simulation
orientation  = 1 1 0       ! (x,y) plane
$end_simulation-dimension  !
!-----!

```

Example 3 (3D)

```

!-----!
$simulation-dimension      !
dimension    = 3           ! 3D simulation
orientation  = 1 1 1       !
$end_simulation-dimension  !
!-----!

```

### 5.3.2 \$regions

- *General specifiers*
- *Details of specification*
  - *1-dimensional objects (only possible in 1D simulations)*
  - *2-dimensional objects*
  - *3-dimensional objects*

#### General specifiers

To build up a geometry, there are - **dependent on the dimension** of the simulation to be performed - various basic geometry elements available. These geometry elements are specified within the keyword `$regions` and can be clustered to a bigger object later on.

| <b>\$regions</b> |                     |                 |
|------------------|---------------------|-----------------|
| region-number    | <b>integer</b>      | <b>optional</b> |
| region-priority  | <b>integer</b>      | <b>required</b> |
| base-geometry    | <b>character</b>    | <b>required</b> |
| x-coordinates    | <b>double_array</b> | <b>optional</b> |
| y-coordinates    | <b>double_array</b> | <b>optional</b> |
| z-coordinates    | <b>double_array</b> | <b>optional</b> |
| center           | <b>double_array</b> | <b>optional</b> |
| radius           | <b>double</b>       | <b>optional</b> |

(continues on next page)

(continued from previous page)

|                      |              |          |
|----------------------|--------------|----------|
| base-coordinates     | double_array | optional |
| top-coordinates      | double_array | optional |
| corner-coordinates   | double_array | optional |
| semi-ellipse-base    | double_array | optional |
| semi-ellipse-top     | double_array | optional |
| <b>\$end_regions</b> |              | optional |

**region-number****type**integer ( $\geq 1$ )**presence**

required

**example**

4

An integer number to refer to geometry element. Numbering must be unique. All region numbers together must form a dense set  $1, 2, 3, \dots, \text{maxnumber}$ .

**region-priority****type**integer ( $\geq 1$ )**presence**

required

**example**

2

A positive integer to set overwriting priority. In case of overlapping regions, the region with higher priority (= higher numerical value) overwrites the region with lower priority.

**base-geometry****type**

character

**presence**

required

**example**

rectangle

type of geometry object.

- 3D: cuboid, sphere, ...
- 2D: rectangle, circle, ...
- 1D: line

These are specified in detail later.

**x-coordinates****type**

double\_array

**presence**

optional

**unit**

[nm]

**example**

-10.0 10.0

This is for cuboid, rectangle, line. The values represent xmin, xmax from left.

**y-coordinate****type**

double\_array

**presence**

optional

**unit**

[nm]

**example**

-10.0 10.0

This is for cuboid, rectangle, line. The values represent ymin, ymax from left.

**z-coordinates****type**

double\_array

**presence**

optional

**unit**

[nm]

**example**

-10.0 10.0

This is for cuboid, rectangle, line. The values represent zmin, zmax from left.

**center****type**

double\_array

**presence**

optional

**unit**

[nm]

**example**

2.0

This is for circle (2D), sphere (3D).

**radius****type**

double

**unit**

[nm]

**presence**

optional

**example**

3.0

This is for circle (2D), sphere (3D).

**base-coordinates**

**type**  
double\_array

**presence**  
optional

**unit**  
[nm]

**example**  
10.0 20.0 10.0 20.0 15.0 15.0

This is for obelisk, cone, truncated-cone, trapezoid. The values represent xmin xmax ymin ymax zmin zmax of obelisk (cone) base plane. (one pair must be equal)

#### top-coordinates

**type**  
double\_array

**presence**  
optional

**unit**  
[nm]

**example**  
10.0 20.0 10.0 20.0 30.0 30.0

This is for obelisk, cone, truncated-cone, trapezoid, semiellipsoid: The values represents xmin xmax ymin ymax zmin zmax of obelisk (cone) base plane (one pair must be equal) or x y z for semiellipsoid.

#### corner-coordinates

**type**  
double\_array

**presence**  
optional

**unit**  
[nm]

**example**  
10.0 10.0 10.0 10.0 30.0 10.0 20.0 20.0 10.0

Triangle corner coordinates, interpreted via orientation and similar for polygon. For triangular prism: x1 y1 z1 x2 y2 z2 x3 y3 z3

#### semi-ellipse-base

**type**  
double\_array

**presence**  
optional

**unit**  
[nm]

**example**  
60.0 120.0 40.0 40.0

For semiellipse: base line ellipse. These are two pairs of xmin, xmax ymin, ymax zmin, zmax - one pair equal numbers

#### semi-ellipse-top

**type**  
double\_array  
**presence**  
optional  
**unit**  
[nm]  
**example**  
100.0 20.0

For semiellipse: top coordinate pair e.g. for coordinate orientation (101) -> (x,z)

### Details of specification

- *1-dimensional objects (only possible in 1D simulations)*
  - *line*
- *2-dimensional objects*
  - *rectangle*
  - *circle*
  - *triangle*
  - *trapezoid*
  - *polygon*
  - regular-polygon (not implemented yet but could be added -> use *polygon* instead)
  - hexagon (not implemented yet but could be added -> use *polygon* instead)
  - *semiellipse*
- *3-dimensional objects*
  - *cuboid*
  - *sphere*
  - cylinder (not implemented yet but could be added -> use *truncated-cone* instead)
  - *obelisk*
  - *hexagonal-obelisk*
  - *cone*
  - *truncated-cone*
  - *semiellipsoid*
  - regular-prism (not implemented yet but could be added)
  - hexagonal-prism (not implemented yet but could be added -> use *hexagonal-obelisk* instead)
  - *triangular-prism*
  - polygonal-prism (not implemented yet but could be added)
  - pyramid (not implemented yet but could be added -> use *obelisk* instead)
  - regular\_pyramid (not implemented yet but could be added -> use *obelisk* instead)
  - hexagonal\_pyramid (not implemented yet but could be added -> use *hexagonal-obelisk* instead)
  - polygonal\_pyramid (not implemented yet but could be added)



## 1-dimensional objects (only possible in 1D simulations)

### line

```

$regions
  region-number      = 1
  base-geometry       = line
  region-priority     = 1
  x-coordinates       = xmin  xmax
$end_regions

```

- Chosen coordinates must be consistent with simulation orientation.

## 2-dimensional objects

### rectangle

```

$regions
  region-number      = 1
  base-geometry       = rectangle
  region-priority     = 1
  x-coordinates       = xmin  xmax
  y-coordinates       = ymin  ymax
$end_regions

```

- Two pairs of delimiting coordinates are required. Whether these have to be **x-coordinates** and **y-coordinates** as in the example above, or another combination (e.g. **x, z**) depends on the simulation orientation which is specified already.

### circle

```

$regions
  region-number      = 1
  base-geometry       = circle
  region-priority     = 1
  center             = x      y
  radius             = r
$end_regions

```

- The circle is defined by a center with coordinates (x,y) and a radius r.

### triangle (can also be specified using polygon)

```

$regions
  region-number      = 1
  base-geometry       = triangle
  region-priority     = 1
  corner-coordinates = x1 y1    x2 y2    x3 y3
$end_regions

```

- The corner coordinates refer to the plane, specified by the simulation orientation.

### polygon

```

$regions
  region-number      = 1

```

(continues on next page)

(continued from previous page)

```

base-geometry      = polygon
region-priority    = 1
corner-coordinates = x1 y1    x2 y2    x3 y3    x4 y4    x5 y5    ...
$end_regions

```

- The corner coordinates of a **polygon** refer to the plane, specified by the simulation orientation.
- The vertices may be listed clockwise or anticlockwise. The first point could be repeated, i.e. the first point and the last point may be identical but this is not required.
- The input polygon may be a compound polygon consisting of several separate subpolygons.

If so, the first vertex of each subpolygon must be repeated.

### trapezoid (can also be specified using polygon)

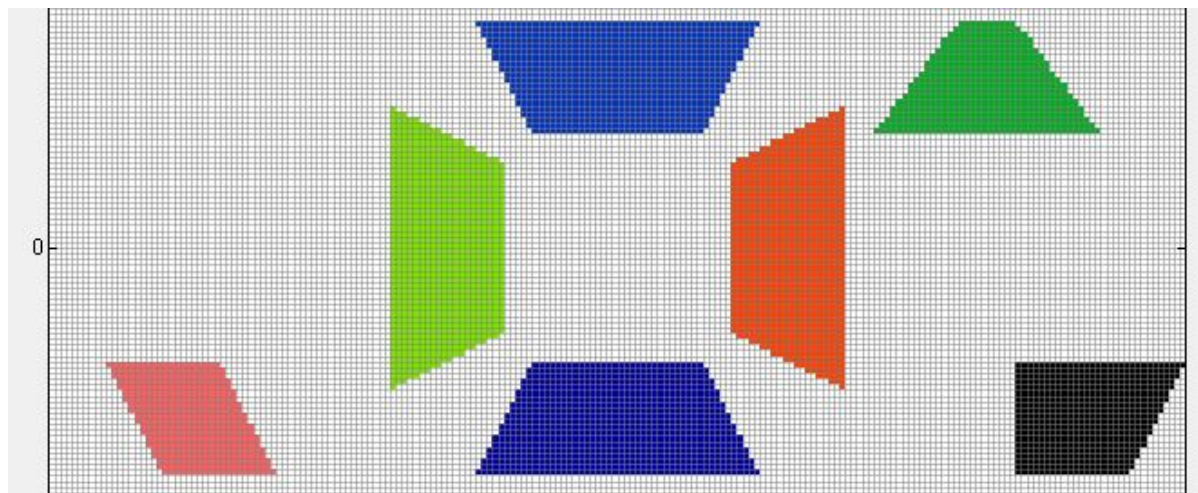
```

$regions
region-number      = 1
base-geometry       = trapezoid
region-priority     = 1
top-coordinates     = -15.0 15.0    -20.0 -20.0    ! xmin xmax  ymin ymax
base-coordinates    = -25.0 25.0    -40.0 -40.0    ! xmin xmax  ymin ymax
$end_regions

```

- For each, **base-coordinates** and **top-coordinates**, 2 values must be equal (meaning **only** 2 values can be equal), e.g.  $y_{\min}=y_{\max}=-40.0$ . Then one boundary of the object is a line at  $y=-40$  nm. Base and top planes of the trapezoid must be in a coordinate plane. This plane is identified by the implicit rule, that a pair of coordinate values (e.g.  $y_{\min}$   $y_{\max}$ ) has identical values ( $y_{\min} = y_{\max}$ ). This also implies that base and top lines are parallel to either the x or the y axis.

The example given above shows the blue trapezoid at the bottom of this figure. The top line extends in x direction from -15 to 15 nm. The top line has a constant y coordinate of  $y = -20$  nm. The base line extends in x direction from -25 to 25 nm. The base line has a constant y coordinate of  $y = -40$  nm.



### semiellipse

```

$regions
region-number      = 1
base-geometry       = semiellipse
region-priority     = 1
semi-ellipse-base   = xmin xmax  ymin ymax

```

(continues on next page)

(continued from previous page)

```
semi-ellipse-top    =    x1        y1
$end_regions
```

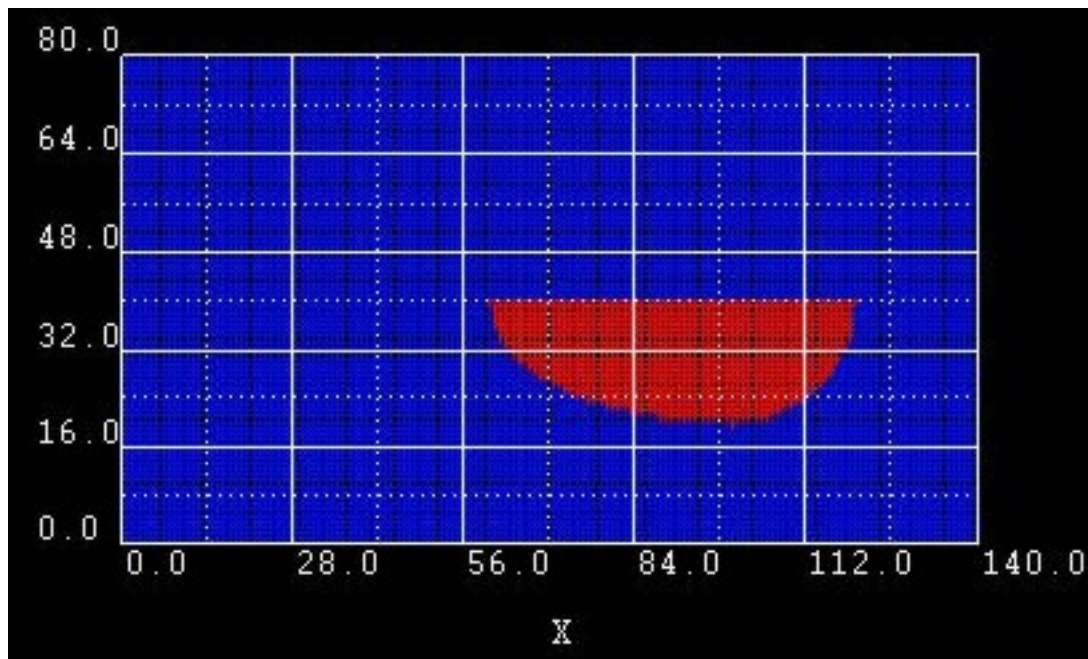
- **semi-ellipse-base:** Here, 2 values must be equal (meaning only 2 values can be equal), e.g.  $y_{min}=y_{max}=40.0$ . Then one boundary of the object is a line at  $y=40$  nm. Base plane of the semiellipse must be in a coordinate plane. This plane is identified by the implicit rule, that a pair of coordinate values (e.g.  $y_{min}$   $y_{max}$ ) has identical values ( $y_{min} = y_{max}$ ).
- **semi-ellipse-top:** This defines a point which determines the height of the semiellipse. In our example ( $y_{min} = y_{max}$ ), the plane is in the  $(x,z)$ -coordinate plane. Top coordinates specify an arbitrary point “above” the ellipse, representing the base of the semiellipse.
- Top coordinate must be lower than upper coordinate of base line border in direction of axis mentioned above.
- Top coordinate must be higher than lower coordinate of base line border in direction of axis mentioned above.
- e.g.) in case of  $y_{min} = y_{max}$ ,  $x_{min} < x_1 < x_{max}$  and ( $y_1 < y_{min}=y_{max}$  or  $y_1 > y_{min}=y_{max}$ )

**Example1**

```
semi-ellipse-base = 60.0 120.0 40.0 40.0 semi-ellipse-top = 100.0
20.0
```

From these data, the following points are extracted: (point: (x,y))

- base: ( 60,40) (120,40)
- top: (100,20)

**Example2**

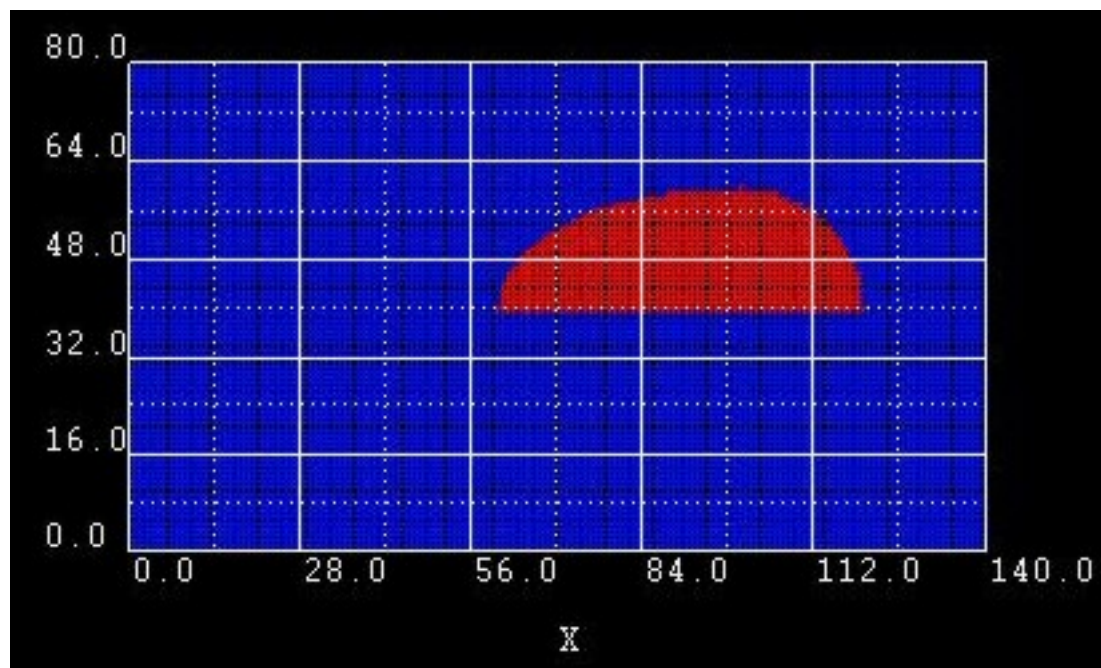
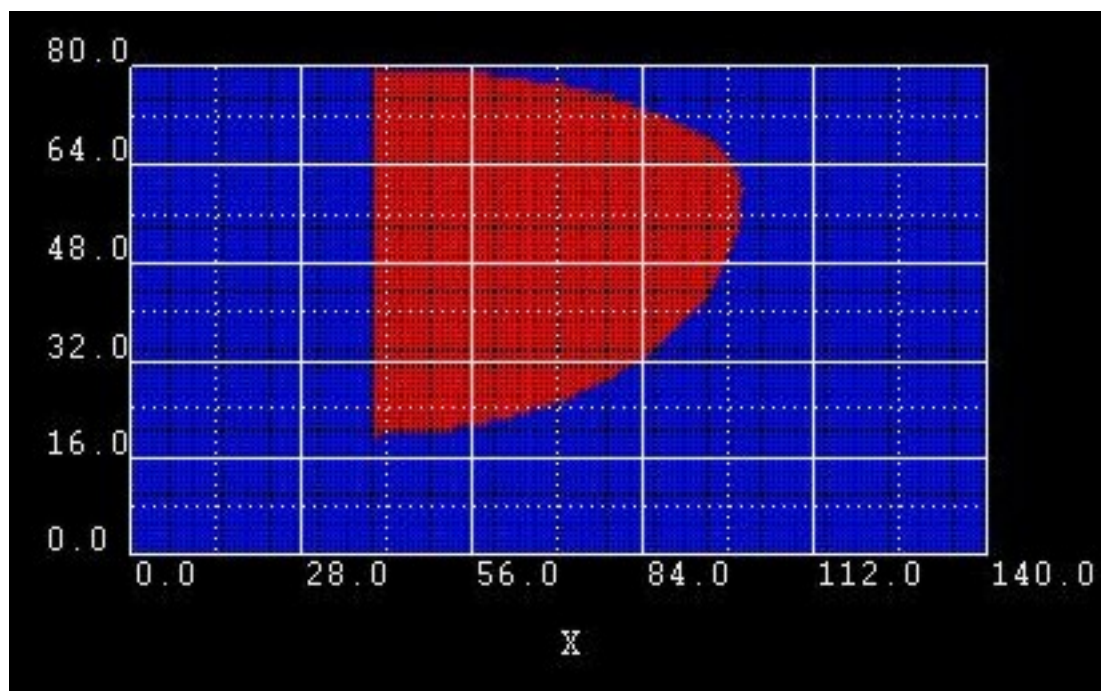
```
semi-ellipse-base = 60.0 120.0 40.0 40.0 semi-ellipse-top = 100.0
60.0
```

We changed the **y1** coordinate of semi-ellipse-top from 20.0 to 60.0.

**Example3**

```
semi-ellipse-base = 40.0 40.0 20.0 80.0 semi-ellipse-top = 100.0
60.0
```

Here we changed semi-ellipse-base: Now the two **x** coordinates have identical values.

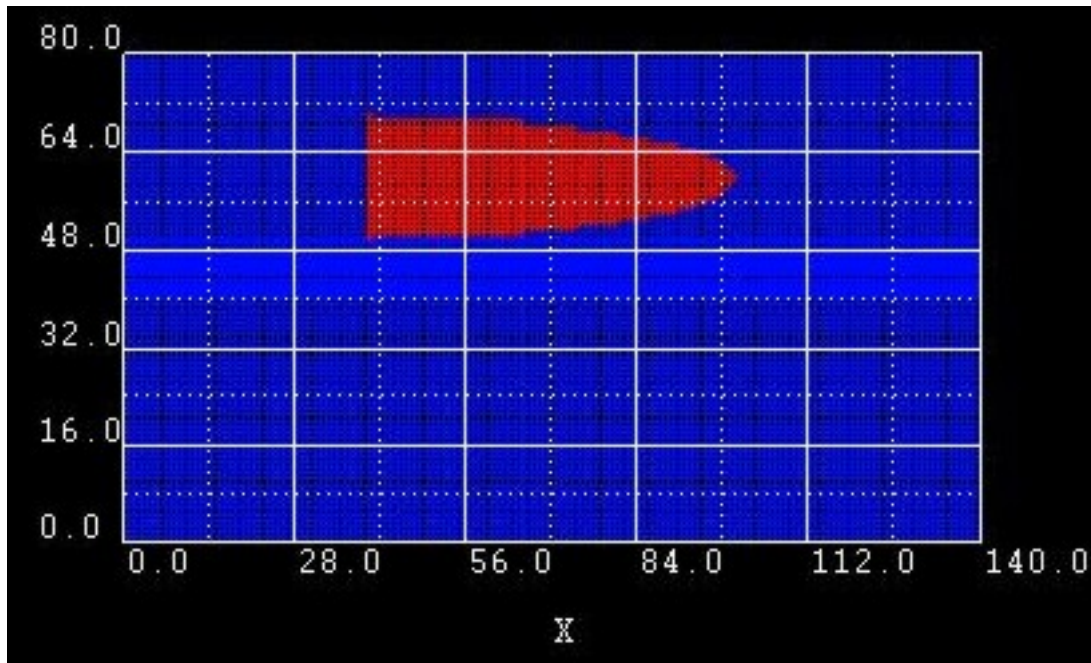




**Example4**

```
semi-ellipse-base = 40.0 40.0 50.0 70.0 semi-ellipse-top = 100.0
60.0
```

Here we changed the **ymin** and **ymax** coordinates of **semi-ellipse-base**: Now the **y** extension of the semiellipse is restricted from **ymin** = 50 nm to **ymax** = 70 nm. The baseline is at the fixed value for **x** = 40 nm.

**Example5**

```
semi-ellipse-base = 40.0 40.0 40.0 80.0 semi-ellipse-top = 60.0
60.0
```

```
semi-ellipse-base = 40.0 40.0 40.0 80.0 semi-ellipse-top = 20.0
60.0
```

Here we built a circle out of 2 semi-ellipses. However, it is obviously easier to use **circle** instead.

**Example6**

```
semi-ellipse-base = 42.0 42.0 40.0 80.0 semi-ellipse-top = 60.0
60.0
```

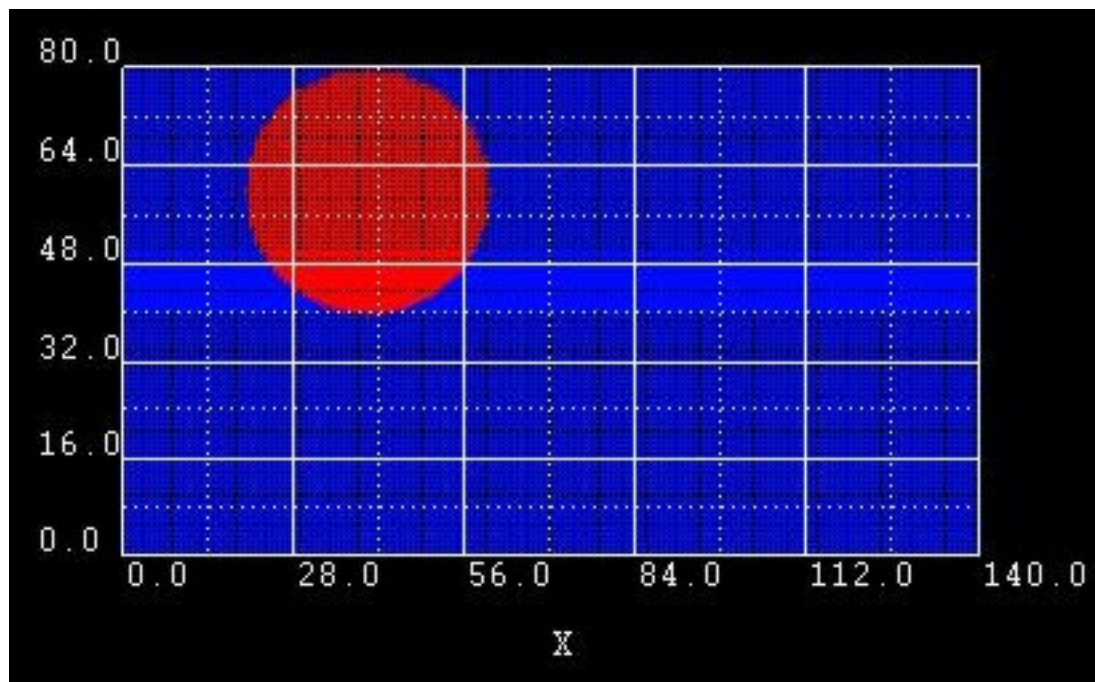
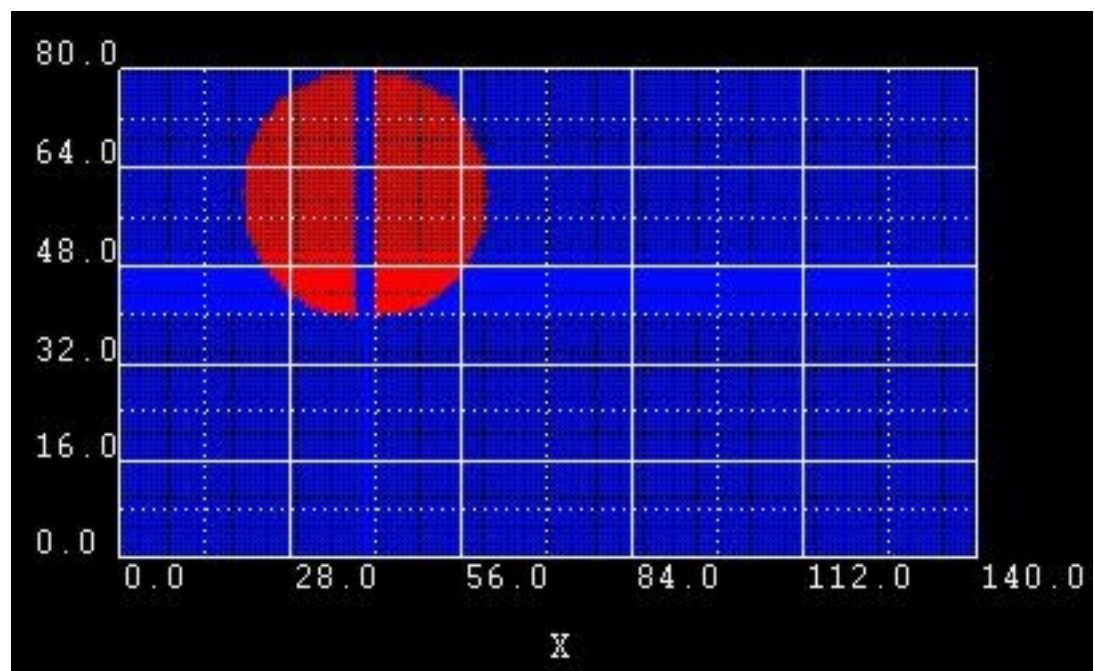
```
semi-ellipse-base = 38.0 38.0 40.0 80.0 semi-ellipse-top = 20.0
60.0
```

Same as example 5 but this time, we moved the baselines a little bit apart from each other to make example 5 easier to understand.

**3-dimensional objects****cuboid****\$regions**

```
region-number      = 1
base-geometry      = cuboid
region-priority    = 1
x-coordinates      = xmin  xmax
y-coordinates      = ymin  ymax
```

(continues on next page)



(continued from previous page)

```

    z-coordinates      = xmin  xmax
$end_regions

```

- The surfaces of the **cuboid** are assumed to be in coordinate planes of the simulation coordinate system. The coordinates above specify the six coordinate planes which limit the cuboid.

### sphere

```

$regions
  region-number      = 1
  base-geometry      = sphere
  region-priority    = 1
  center             = x      y      z      ! [nm]
  radius             = r              ! [nm]
$end_regions

```

- The sphere is defined by a center with coordinates (x,y,z) and a radius r.

### obelisk

```

$regions
  region-number      = 1
  base-geometry      = obelisk
  region-priority    = 1
  base-coordinates   = xmin xmax  ymin ymax  zmin zmax
  top-coordinates    = xmin xmax  ymin ymax  zmin zmax
$end_regions

```

- Base and top plane of the obelisk have to be in parallel coordinate planes. These planes are identified by the implicit rule, that a pair of coordinate values (e.g. `ymin ymax`) has the same value (`ymin = ymax`).
- In this example, the plane is in the (x,z)-coordinate plane. The remaining four coordinates specify a rectangle in the corresponding plane.

### truncated-cone and cylinder

```

$regions
  region-number      = 1
  base-geometry      = truncated-cone           ! can be used to
→ specify a cylinder
  region-priority    = 1
  base-coordinates   = xmin xmax  ymin ymax  zmin zmax
  top-coordinates    = xmin xmax  ymin ymax  zmin zmax
$end_regions

```

- A cone with its apex cut off by a plane is called a truncated cone. In our implementation, the truncated cone is bounded by two ellipses of different size that are aligned parallel to each other.
- Base and top plane of the truncated cone have to be in parallel coordinate planes. This plane is identified by the implicit rule, that a pair of coordinate values (e.g. `xmin xmax`) has the same value (`xmin = xmax`).
- In this example, the plane is in the (y,z)-coordinate plane. `ymin ymax` and `zmin zmax` specify the diameter of the truncated cone top and base in the y and z direction, respectively. This corresponds to the specification of ellipses in the base and top plane.

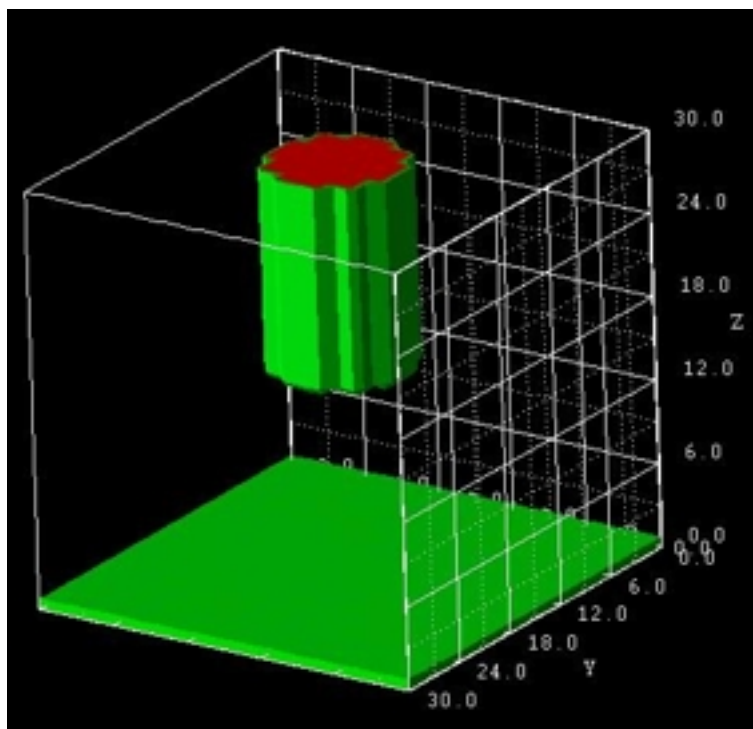
*How to specify a cylinder?*

A cylinder is specified as a special case of a truncated-cone where the two boundary planes are circles. For a truncated cone, one specifies base and top coordinates.

Let us assume we have a spherical cylinder of diameter 10 nm and height 15 nm. Then the base and top coordinates would be, for example, `base-coordinates = 10.0 20.0 10.0 20.0 15.0 15.0` (`xmin,xmax,ymin,ymax,zmin,zmax`) = (10,20,10,20,15,15)

`top-coordinates = 10.0 20.0 10.0 20.0 30.0 30.0` (`xmin,xmax,ymin,ymax,zmin,zmax`) = (10,20,10,20,30,30)

In other words, the x and y-coordinates specify the principal axes of the bottom and top ellipse (or circle) of the cylinder, respectively, and the z-coordinates specify the planes in which these two ellipses lie.



## cone

```
$regions
  region-number      = 1
  base-geometry       = cone
  region-priority     = 1
  base-coordinates    = xmin xmax  ymin ymax  zmin zmax
  top-coordinates     = x y z
$end_regions
```

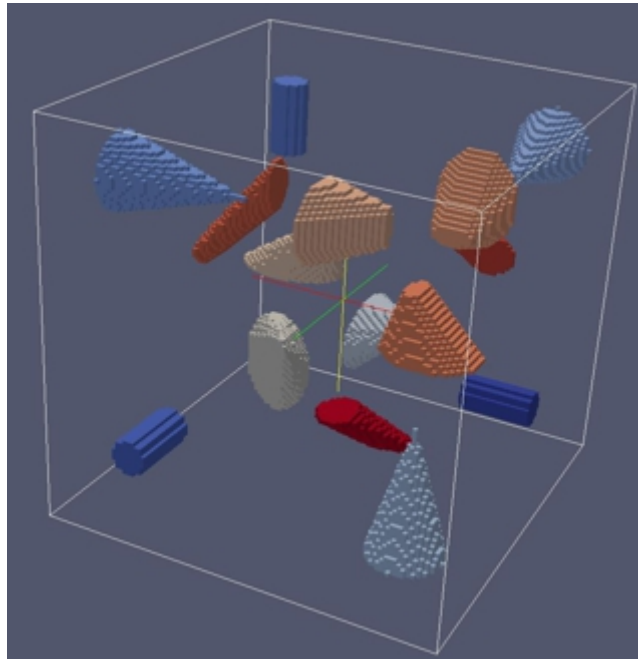
Base plane of the cone has to be in parallel to the coordinate system planes. This plane is identified by the implicit rule, that a pair of coordinate values (e.g. `xmin xmax`) has the same value (`xmin = xmax`). In this example, the plane is in the (y,z)-coordinate plane. `ymin ymax` and `zmin zmax` specify the diameter of the cone base in the y and z direction, respectively. This corresponds to the specification of an ellipse in the base plane. The top of the cone, the apex, is defined by the point (x,y,z).

The cone does not have to be *right circular* (where *circular* means that the base is a circle and *right* means that the axis passes through the center of the base at right angles to its plane).

*Oblique* cones are allowed, in which the axis does not pass perpendicularly through the center of the base.



The following figure shows several **cones** and **truncated-cones** (including the special case of **cylinders**).



The cones are defined as follows:

```
!-----
!-----
! This is a cylinder. The base and top planes are parallel to the (y,z)
! plane.
!-----
!-----
region-number    = 1    base-geometry = truncated-cone    region-priority
! = 1            ! cylinder
base-coordinates = 80.0 80.0    -80.0 -60.0    -80.0 -60.0
!               ! xmin xmax ymin ymax zmin zmax
top-coordinates  = 40.0 40.0    -80.0 -60.0    -80.0 -60.0
!               ! xmin xmax ymin ymax zmin zmax

!-----
!-----
! This is a right circular cone. The base plane is parallel to the (y,z)
! plane.
!-----
!-----
region-number    = 2    base-geometry = cone                region-priority
! = 1            ! cone (right circular)
base-coordinates = -80.0 -80.0    50.0 90.0    50.0 90.0
!               ! xmin xmax ymin ymax zmin zmax
top-coordinates  = -10.0 70.0 70.0
!               ! x y z

!-----
!-----
! This is a cone where the projection of the apex onto the base plane is
! located outside the base plane.
```

(continues on next page)

(continued from previous page)

```

! The base plane is parallel to the (y,z) plane.
!-----
↳-----
region-number      = 3      base-geometry = cone                region-priority_
↳= 2              ! cone
base-coordinates = -10.0 -10.0      -60.0 -20.0      -80.0 -20.0
↳              ! xmin xmax ymin ymax zmin zmax
top-coordinates  =  40.0 -10.0 -10.0
↳              ! x y z

!-----
! This is a truncated cone. The base plane is parallel to the (y,z) plane.
!-----
region-number      = 4      base-geometry = truncated-cone      region-priority =
↳3                ! truncated-cone
base-coordinates =  20.0 20.0      50.0 90.0      50.0 100.0
↳              ! xmin xmax ymin ymax zmin zmax
top-coordinates  =  50.0 50.0      60.0 80.0      60.0  70.0
↳              ! xmin xmax ymin ymax zmin zmax

!-----
↳-----
! These are three truncated cones where the projection of the top plane on_
↳the base plane is outside the base plane.
! The base plane is parallel to the (y,z) plane.
!-----
↳-----
region-number      = 5      base-geometry = truncated-cone      region-priority_
↳= 4              ! truncated-cone
base-coordinates = -50.0 -50.0      20.0 40.0      20.0  60.0
↳              ! xmin xmax ymin ymax zmin zmax
top-coordinates  = -20.0 -20.0      50.0 80.0      20.0  40.0
↳              ! xmin xmax ymin ymax zmin zmax

```

## semiellipsoid

```

$regions
  region-number      = 1
  base-geometry       = semiellipsoid
  region-priority    = 1
  base-coordinates   = xmin xmax  ymin ymax  zmin zmax
  top-coordinates    = xtop ytop ztop
$end_regions

```

Base plane of the semiellipsoid must be in a coordinate plane. This plane is identified by the implicit rule, that a pair of coordinate values (e.g. `ymin ymax`) has identical values (`ymin = ymax`).

In this example, the plane is in the (x,z)-coordinate plane. Top coordinates specify an arbitrary point “above” the ellipse, representing the base of the semiellipsoid.

### Example

A 3D sphere can be constructed from two semiellipsoids. However, it is obviously easier to use `sphere` instead.

In this example, the bottom planes of the two half-spheres are at  $z = 5$  nm. The upper half-sphere extends from 5 nm to 6 nm, the lower half-sphere from 5 nm to 4 nm. The extensions in x and y directions for both half-spheres are from 4 nm to 6 nm.

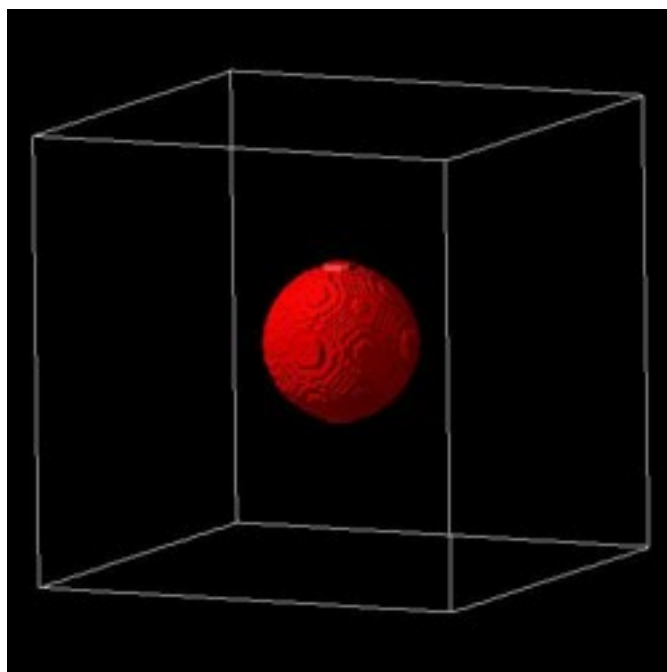
Consequently, the sphere has a diameter of 2 nm.

```

region-number      = 1
base-geometry      = semiellipsoid           ! (upper half-
↳sphere)
base-coordinates   = 4.0 6.0   4.0 6.0   5.0 5.0   ! xmin xmax,
↳ymin ymax zmin=zmax
top-coordinates    = 5.0 5.0 6.0           ! xtop ytop,
↳ztop

region-number      = 2
base-geometry      = semiellipsoid           ! (lower half-
↳sphere)
base-coordinates   = 4.0 6.0   4.0 6.0   5.0 5.0   ! xmin xmax,
↳ymin ymax zmin=zmax
top-coordinates    = 5.0 5.0 4.0           ! xtop ytop,
↳ztop

```



### hexagonal-obelisk

```

$regions
  region-number      = 1
  base-geometry      = hexagonal-obelisk
  region-priority    = 1
  base-coordinates   = xmin xmax   ymin ymax   zmin zmax
  top-coordinates    = xmin xmax   ymin ymax   zmin zmax
$end_regions

```

Base and top plane of the hexagonal-obelisk have to be in parallel coordinate planes. These planes are identified by the implicit rule, that a pair of coordinate values (e.g. `zmin zmax`) has the same value (`zmin = zmax`). In this example, the plane is in the (x,y)-coordinate plane. The remaining four coordinates specify a rectangle in the corresponding plane.

This geometry element is useful for wurtzite.

Many thanks to Lu Fu-Fa (Institute of Technology (CCIT), Taiwan, R.O.C.) for useful suggestions regarding the implementation of this geometry element.

Two hexagonal-obelisk shapes are possible:

- **‘hexagonal-cylinder’ with 6-fold rotational symmetry axis oriented along the z direction (i.e.  $z_{min} = z_{max}$ ).**

Requirements: -  $x_{min}(base) = x_{min}(top)$  -  $x_{max}(base) = x_{max}(top)$  -  $y_{min}(base) = y_{min}(top)$  -  $y_{max}(base) = y_{max}(top)$

**Note**

This condition should be fulfilled:

$$- y_{max}(base) - y_{min}(base) > (x_{max}(base) - x_{min}(base)) / 0.866$$

where  $0.866 = \cos\pi/6$

- **‘hexagonal-pyramid’ with 6-fold rotational symmetry axis oriented along the z direction (i.e.  $z_{min} = z_{max}$ ).**

Requirements: -  $x_{min}(top)$ ,  $x_{max}(top)$ ,  $y_{min}(top)$ ,  $y_{max}(top)$  arbitrary

**Note**

This condition should be fulfilled:

$$- y_{max} - y_{min} > (x_{max} - x_{min}) / 0.866$$

For both shapes it holds:

- height of pyramid/cylinder:  $z_{max}(top) - z_{max}(base)$  ( $z_{min} = z_{max}$ )
- width in x direction (distance between two parallel planes):  $x_{max}(base) - x_{min}(base)$
- width in y direction (distance between two corners) if the condition  $y_{max}(base) - y_{min}(base) > (x_{max}(base) - x_{min}(base)) / 0.866$  is fulfilled:
- If the above cited condition is not fulfilled, then the width is:  $y_{max}(base) - y_{min}(base)$
- width along y > width along x if  $y_{max}(base) - y_{min}(base) > x_{max}(base) - x_{min}(base)$
- center of hexagonal base plane on x axis:  $0.5 \cdot (x_{min}(base) + x_{max}(base))$
- center of hexagonal base plane on y axis:  $0.5 \cdot (y_{min}(base) + y_{max}(base))$ :  $y_{min}(base)$  and  $y_{max}(base)$  can be used to shift the hexagon along the y axis.

Two sides of the hexagonal base plane are aligned parallel to the y axis. To rotate the hexagonal base plane by 30 degrees, the user has to specify values for  $x_{min}(top)$ ,  $x_{max}(top)$ ,  $y_{min}(top)$ ,  $y_{max}(top)$  so that it holds:

$$x_{max}(top) - x_{min}(top) > y_{max}(top) - y_{min}(top)$$

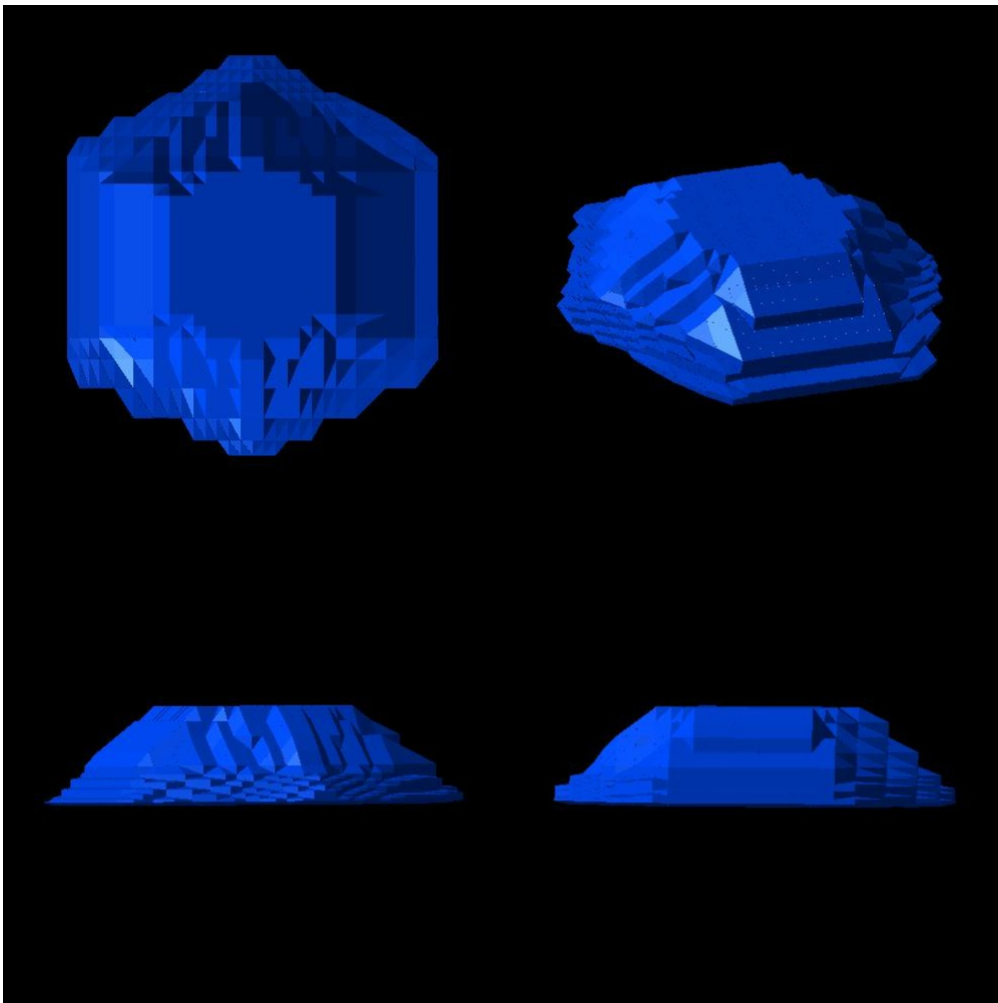
In this case it holds:

- width in x direction (distance between two corners) if the condition  $x_{max}(base) - x_{min}(base) > (y_{max}(base) - y_{min}(base)) / 0.866$  is fulfilled:
- width in y direction (distance between two parallel planes):  $y_{max}(base) - y_{min}(base)$
- width along y < width along x if  $x_{max}(base) - x_{min}(base) > y_{max}(base) - y_{min}(base)$
- center of hexagonal base plane on x axis:  $0.5 \cdot (x_{min}(base) + x_{max}(base))$ :  $x_{min}(base)$  and  $x_{max}(base)$  can be used to shift the hexagon along the x axis.

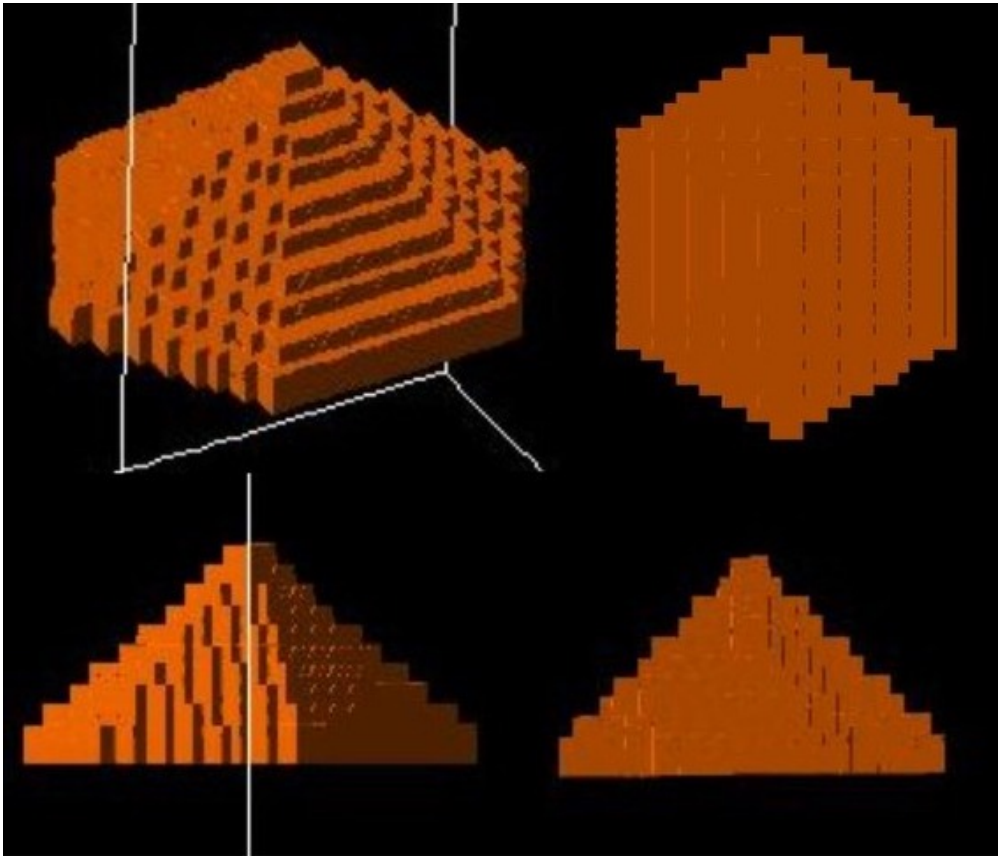
If the 6-fold rotational axis is oriented along the x (i.e.  $x_{min} = x_{max}$ ) or y directions (i.e.  $y_{min} = y_{max}$ ), cyclic permutations hold for the above statements.

**Example**

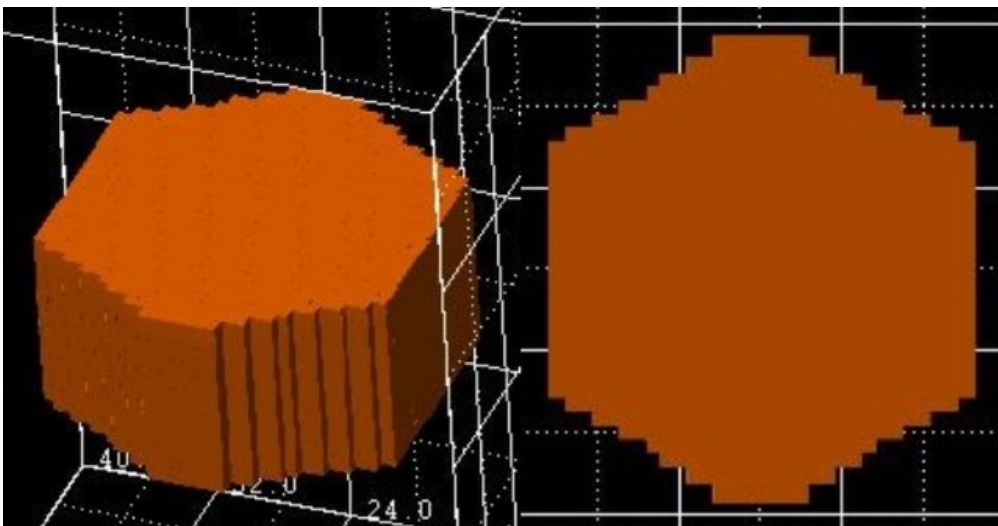
- Hexagonal shaped pyramid with flat top plane:



- Hexagonal shaped pyramid:



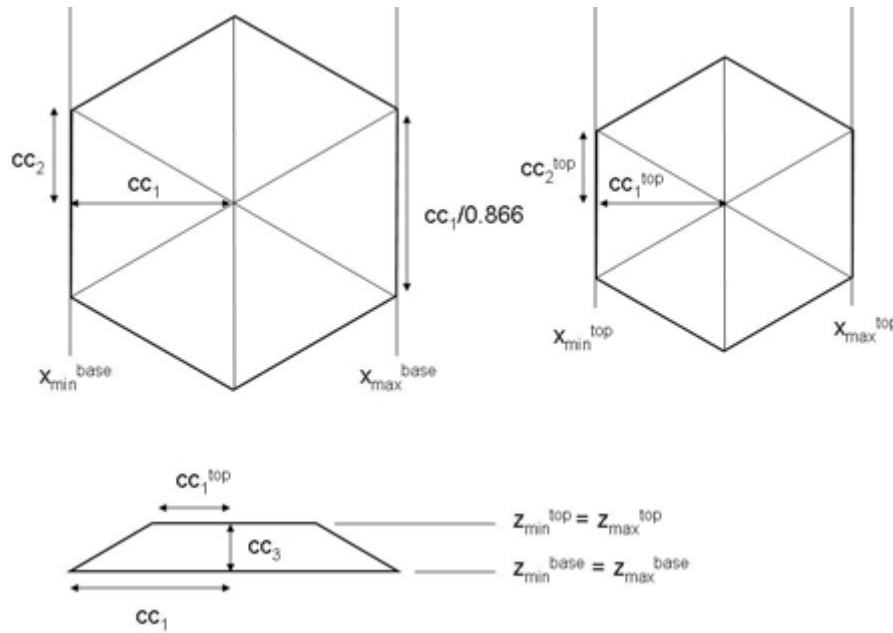
- Hexagonal shaped “cylinder”:



### triangular-prism

```
$regions
  region-number      = 1
  base-geometry       = triangular-prism
  region-priority     = 1
  corner-coordinates = x1 y1 z1    x2 y2 z2    x3 y3 z3
```

(continues on next page)



(continued from previous page)

**\$end\_regions**

x4 y4 z4    x5 y5 z5    x6 y6 z6

Restrictions: **triangular-prism** must be oriented so that the triangles are perpendicular to either the x, y or z directions.

Example: Triangles perpendicular to z direction. Then it must hold:

corner-coordinates

- $z1 = z2 = z3$
- $4 = z5 = z6$

In addition it holds:

- $x1 = x4, y1 = y4$
- $x2 = x5, y2 = y5$
- $x3 = x6, y3 = y6$

**Example**

```

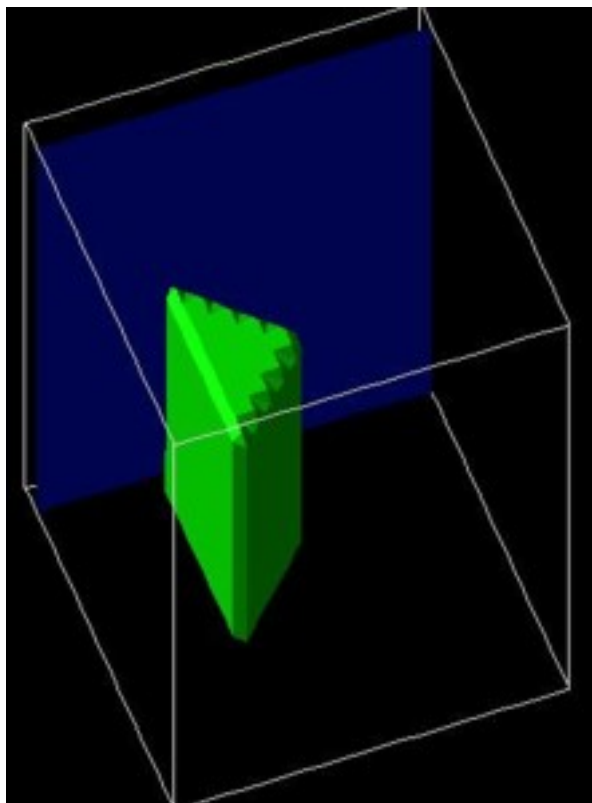
$regions
  region-number      = 1
  base-geometry      = triangular-prism
  region-priority    = 1
  corner-coordinates = x1 y1 z1   x2 y2 z2   x3 y3 z3
                      x4 y4 z4   x5 y5 z5   x6 y6 z6
$end_regions

```

### 5.3.3 \$region-cluster

Regions can be clustered to bigger objects by the keyword **\$region-cluster**. Any region must be assigned to a cluster which is labeled by its **cluster-number**.

Also the so-called *default region* must be assigned to a cluster. The default region is the rest of the simulation domain which is not filled out by defined regions. It might happen that the default region is an *empty region*. Nevertheless, it must be assigned to a cluster.



In any case it holds:

default region number = maximum region number defined + 1

The regions which have to be clustered to one cluster are specified by their numbers like this `region-number = 2 3 5`.

|                               |                      |                 |
|-------------------------------|----------------------|-----------------|
| <b>\$region-cluster</b>       |                      | <b>required</b> |
| cluster-number                | <b>integer</b>       | <b>required</b> |
| region-numbers                | <b>integer_array</b> | <b>required</b> |
| apply-constant-el-Fermi-level | <b>double</b>        | <b>optional</b> |
| apply-constant-hl-Fermi-level | <b>double</b>        | <b>optional</b> |
| <b>\$end_region-cluster</b>   |                      | <b>required</b> |

Explanation of specifiers

**cluster-number**

**type**

integer  $\geq 1$

**presence**

required

An integer number to refer to clustered geometry element.

**region-numbers**

**type**

integer array containing integers  $\geq 1$

**presence**

required

Region numbers belonging to this cluster `cluster-number`.

**apply-constant-el-Fermi-level**



**type**  
double  
**presence**  
optional  
**unit**  
(eV)  
**example**  
0.2

Applies a constant electron Fermi level  $E_{F,n}$  to this cluster. The energetic distance of the conduction band edges with respect to this Fermi level determines the electron density.

#### apply-constant-hl-Fermi-level

**type**  
double  
**presence**  
optional  
**unit**  
(eV)  
**example**  
-0.1

Applies a constant hole Fermi level  $E_{F,p}$  to this cluster. The energetic distance of the valence band edges with respect to this Fermi level determines the hole density.

#### Example

We have defined 5 regions under keyword `$regions`, region 2 and 4 are clustered to form `cluster-number 2` (i.e. region 2 and 4 now will consist of the same *material* as a material kind is assigned to a *cluster* and not to a *region*). `region-number 6` is an undefined *default region* (Assumption: We have defined only 5 regions.) and must be the default rest. Now there are two possibilities:

##### Possibility 1:

```
!-----!
$region-cluster      !
cluster-number = 1    region-numbers = 1    ! e.g. GaAs
cluster-number = 2    region-numbers = 2  4    ! e.g. AlAs
cluster-number = 3    region-numbers = 3    ! e.g. GaAs
cluster-number = 4    region-numbers = 5  6    ! e.g. InP, 6 = default_
↪region
$end_region-cluster  !
!-----!
```

Here, the default region has the same material properties as `cluster-number 4`. The default region can consist of some undefined region (the *remaining part* of the simulation area which is not filled yet) or it can be *empty* if the *whole* simulation area is already filled with other objects.

##### Possibility 2:

```
!-----!
$region-cluster      !
cluster-number = 1    region-numbers = 1    ! e.g. GaAs
cluster-number = 2    region-numbers = 2  4    ! e.g. AlAs
cluster-number = 3    region-numbers = 3    ! e.g. GaAs
cluster-number = 4    region-numbers = 5    ! e.g. InP
cluster-number = 5    region-numbers = 6    ! e.g. InP, 6 = default_
↪region
```

(continues on next page)

(continued from previous page)

```
$end_region-cluster
!-----!
```

Here, the default region has its *own* material properties which have to be defined later under keyword *\$material*. Again, the default region can consist of some undefined region (the *remaining part* of the simulation area which is not filled yet) or it can be *empty* if the *whole* simulation area is already filled with other objects.

#### Useful application for a default region

If one has a complicated geometry such as a quantum dot which is surrounded by a GaAs cap layer, one can define the complicated geometry as usual and for the (complicated) surrounding material one just uses the *default* option.

#### **i** Note

Be careful in 3D when you specify for strain calculation *strain-minimization* (see *\$strain-minimization-model*) to define your substrate as a separate cluster.

### 5.3.4 \$quantum-cluster

For quantum mechanical calculations, you have to define quantum regions (*\$quantum-regions*) and quantum clusters on which certain quantum models (*\$quantum-model-electrons*, *\$quantum-model-holes*) are applied. As for the *regions* and *current-regions*, *quantum-regions* can be *clustered*. At least cluster number 1 has to be specified if the quantum mechanical properties should be calculated at all.

The syntax of the *region-cluster*, *quantum-cluster*, and *current-cluster* is very similar. For more information on these specifiers see *\$region-cluster* and *\$current-cluster*.

|                               |                      |                 |
|-------------------------------|----------------------|-----------------|
| <b>\$quantum-cluster</b>      |                      | <b>optional</b> |
| cluster-number                | <b>integer</b>       | <b>required</b> |
| region-numbers                | <b>integer_array</b> | <b>required</b> |
| deactivate-cluster            | <b>character</b>     | <b>optional</b> |
| apply-constant-el-Fermi-level | <b>double</b>        | <b>optional</b> |
| apply-constant-hl-Fermi-level | <b>double</b>        | <b>optional</b> |
| <b>\$end_quantum-cluster</b>  |                      | <b>optional</b> |

Explanation of specifiers

#### **cluster-number**

##### **type**

integer >= 1

##### **presence**

required

An integer number to refer to clustered geometry element.

#### **region-numbers**

##### **type**

integer array containing integers >= 1

##### **presence**

required

Region numbers belonging to the cluster *cluster-number*.

#### **deactivate-cluster**

##### **type**

character

**presence**  
optional

**value**  
yes or no

**default**  
no

Flag to switch off quantum cluster. Switching off quantum calculation means classical simulation. This is a very convenient way to turn the quantum calculation off and on. For a first test, it is recommended to a classical calculation as it is much faster. Then you can check whether the geometry and the doping regions are defined correctly, and if the strain has the expected influence on the conduction and valence band edges. Once you are convinced that you have set up the structure correctly, you can do a quantum mechanical simulation.

#### apply-constant-el-Fermi-level

**type**  
double

**presence**  
optional

**unit**  
(eV)

**example**  
0.2

Applies a constant electron Fermi level  $E_{F,n}$  to this quantum cluster. The energetic distance of the conduction band edges with respect to this Fermi level determines the electron density.

#### apply-constant-hl-Fermi-level

**type**  
double

**presence**  
optional

**unit**  
(eV)

**example**  
-0.1

Applies a constant hole Fermi level  $E_{F,p}$  to this quantum cluster. The energetic distance of the valence band edges with respect to this Fermi level determines the hole density.

#### Examples

Here, only one quantum cluster is defined.

```
!-----
$quantum-cluster
  cluster-number = 1      region-numbers = 1
  deactivate-cluster = no
!deactivate-cluster = yes
$end_quantum-cluster
!-----
```

Here, two quantum regions are combined to form a common quantum cluster.

```
!-----
$quantum-cluster
```

(continues on next page)

(continued from previous page)

```

cluster-number = 1      region-numbers = 1 2
$end_quantum-cluster
!-----

```

Here, three separate quantum clusters are defined that are treated independently, i.e. three different Schrödinger equations are solved. Typically, it is best to have only one quantum cluster. However, in some cases, using separate clusters can lead to faster calculations. But this should be used with care, as often the wave functions with higher energies are distributed over the whole device.

```

!-----
$quantum-cluster
cluster-number = 1      region-numbers = 1
cluster-number = 2      region-numbers = 2
cluster-number = 3      region-numbers = 3
$end_quantum-cluster
!-----

```

### 5.3.5 \$current-cluster

For current calculations, you have to define current regions (*\$current-regions*) and current clusters on which certain current models (*\$current-models*) are applied. As for the *regions* and *quantum-regions*, *current-regions* can be *clustered*. At least cluster number 1 has to be specified if the current has to be calculated at all.

The syntax of the *region-cluster*, *quantum-cluster*, and *current-cluster* is very similar. For more information on these specifiers see *\$region-cluster* and *\$quantum-cluster*.

|                          |                      |                 |
|--------------------------|----------------------|-----------------|
| <b>\$current-cluster</b> |                      |                 |
| cluster-number           | <b>integer</b>       | <b>optional</b> |
| region-numbers           | <b>integer_array</b> | <b>required</b> |
| deactivate-cluster       | <b>character</b>     | <b>optional</b> |
| \$end_current-cluster    |                      | <b>optional</b> |

Explanation of specifiers

#### cluster-number

##### type

integer >= 1

##### presence

required

An integer number to refer to clustered geometry element.

#### region-numbers

##### type

integer array containing integers >= 1

##### presence

required

Region numbers belonging to cluster cluster-number.

#### deactivate-cluster

##### type

character

##### presence

optional

##### value

yes or no

**default**  
no

Flag to switch off current clusters. Switching off current calculation means equilibrium simulation. This is a very convenient way to turn the current calculation off and on.

Example

```
!-----
$current-cluster
  cluster-number = 1      region-numbers = 1
  deactivate-cluster = no
!deactivate-cluster = yes
$end_current-cluster
!-----
```

### 5.3.6 \$current-regions

In order to make it possible to combine several different modes of current calculation in one device, there is an extra set of regions and clusters for the calculation of the current. The syntax is the same as for the other regions (*\$regions*, *\$quantum-regions*).

If no current region is specified the whole device is numbered as current region 1, i.e. this keyword does not have to be present.

|                              |                     |                 |
|------------------------------|---------------------|-----------------|
| <b>\$current-regions</b>     |                     | <b>optional</b> |
| region-number                | <b>integer</b>      | <b>required</b> |
| base-geometry                | <b>character</b>    | <b>required</b> |
| region-priority              | <b>integer</b>      | <b>required</b> |
|                              |                     |                 |
| x-coordinates                | <b>double_array</b> | <b>optional</b> |
| y-coordinates                | <b>double_array</b> | <b>optional</b> |
| z-coordinates                | <b>double_array</b> | <b>optional</b> |
|                              |                     |                 |
| base-coordinates             | <b>double_array</b> | <b>optional</b> |
| top-coordinates              | <b>double_array</b> | <b>optional</b> |
|                              |                     |                 |
| corner-coordinates           | <b>double_array</b> | <b>optional</b> |
|                              |                     |                 |
| semi-ellipse-base            | <b>double_array</b> | <b>optional</b> |
| semi-ellipse-top             | <b>double_array</b> | <b>optional</b> |
| <b>\$end_current-regions</b> |                     | <b>optional</b> |

### 5.3.7 \$current-models

So far only the drift-diffusion model has been implemented but you never know what comes next.

|                             |                      |                 |
|-----------------------------|----------------------|-----------------|
| <b>\$current-models</b>     |                      | <b>optional</b> |
| model-number                | <b>integer</b>       | <b>required</b> |
| transport-model-name        | <b>character</b>     | <b>required</b> |
| cluster-numbers             | <b>integer_array</b> | <b>required</b> |
| <b>\$end_current-models</b> |                      | <b>optional</b> |

Explanation of specifiers

**model-number**

**type**

integer >= 1

**presence**  
required  
**value**  
1

An integer number to refer to current model. Currently, only one model is implemented, so you have to choose 1.

**transport-model-name**

**type**  
character  
**presence**  
required  
**value**  
simple-drift-model

This is the only option currently.

**cluster-numbers**

**type**  
integer array  
**presence**  
required  
**value**  
1

Currently, only one current cluster supported so far.

Example

```
!-----  
$current-models  
model-number      = 1  
transport-model-name = simple-drift-model  
cluster-numbers   = 1  
$end_current-models  
!-----
```

### 5.3.8 \$impurity-parameters

To specify the properties of impurities used in the simulation.

|                                  |               |                 |
|----------------------------------|---------------|-----------------|
| <b>\$impurity-parameters</b>     |               | <b>optional</b> |
| impurity-number                  | integer       | required        |
| impurity-name                    | character     | optional        |
| impurity-type                    | character     | required        |
| number-of-energy-levels          | integer       | required        |
| energy-levels-relative           | double_array  | required        |
| degeneracy-of-energy-levels      | integer_array | required        |
| incomplete-ionization-N-ref      | double_array  | optional        |
| incomplete-ionization-c          | double_array  | optional        |
| incomplete-ionization-N-b        | double_array  | optional        |
| incomplete-ionization-d          | double_array  | optional        |
| transition-times-cb-to-levels    | double_array  | optional        |
| transition-times-levels-to-vb    | double_array  | optional        |
| <b>\$end_impurity-parameters</b> |               | <b>optional</b> |

Specify here the properties of the impurities used for doping.

**impurity-number****type**

integer

**presence**

required

**example**

1

1, 2, 3, ... are unique impurity numbers labeled in *\$doping-function*.

Optionally, a name for the impurity can be provided.

**impurity-name****type**

character

**presence**

optional

**example**

n-As-in-Si

**example** $0.010 \cdot x + 0.050 \cdot (1-x)$  (unit: eV)**example**

$0.010 \cdot x + 0.050 \cdot (1-x) - 0.020 \cdot x \cdot (1-x)$  (unit: eV) Here, a bowing factor of 0.020 eV is included.

Specify an arbitrary name. The name will be part of some output files. (for later use; it is planned to read in impurity parameters from the database.)

Special feature: If **impurity-name** contains the character **x**, this string is then interpreted as an equation where **x** is the alloy content. This way, a position dependent impurity level can be simulated, e.g. for graded Al(x)Ga(1-x)N layers. In this example, the impurity level would be 10 meV for AlN (x=1) and 50 meV for GaN (x=0). Note that this has influence on CPU time, i.e. the calculation will take longer because the result of the equation has to be evaluated during the calculation.

(For later use: It is planned to read in impurity parameters from the database.)

Several types of impurities are supported.

**impurity-type****type**

character

**presence**

required

**options**

n-type, p-type, trap

Specifies the type of an impurity. **n-type** means that the impurity is treated as a donor, **p-type** as an acceptor. Option **trap** is not supported so far.

Each impurity can have several different energy levels.

**number-of-energy-levels****type**

integer

**presence**

required

**example**

1

**example**

1 2

Number of different energy levels for this impurity. Each energy level is specified in `energy-levels-relative`.

Energy levels

**energy-levels-relative****type**

double array

**presence**

required

**unit**

(eV)

**example**

0.005

**example**

0.005 0.015

**example**

1000.0

When impurity levels are relatively deep compared to the thermal energy  $k_B T$  at room temperature, **incomplete ionization** must be considered.

Energy levels relative to “nearest” band edge (n-type -> conduction band, else valence band) in units of eV. As many energies as energy levels. These energies are meant as **ionization energies**, e.g. a donor with an energy level right below the conduction band edge would be specified by a small positive energy level  $E_D^{\text{ion}}$ .

A large negative value, e.g. -1000.0 eV implies **full ionization**.

**Note**

‘Cheat’ parameter: `energy-levels-relative = -1000.0`, for instance, that means, all electrons are fully ionized from the donors (similar for holes/acceptors). This might be useful for low temperatures like 4 K where usually the degree of ionization is very small. By using `-1000.0` one can force them to be completely ionized.

The energy levels of the donors and acceptors relative to the lowest conduction band edge and highest valence band edge can be output using `dopant-energy-levels = yes` (see [\\$output-densities](#)).

See also our tutorial on [Doped semiconductors](#) to learn more about partial ionization.

Degeneracy of the specified energy levels

**degeneracy-of-energy-levels****type**

integer array

**presence**

required

**example**

2



**example**

4 4

**Shallow donors** have degeneracy factor 2: Outer s orbital is onefold occupied (neutral state). There is one possibility to get rid of one electron but there are two to incorporate one (spin up, spin down).

**Shallow acceptors** have degeneracy factor 4: The  $sp^3$  orbital is threefold occupied. Thus, one possibility to incorporate an electron, four possibilities to get rid of one.

More details on degenerate impurity levels can be found in e.g. [\[ChuangOpto1995\]](#). Note that in nitride semiconductors crystallizing in the wurtzite structure the degeneracy factor may vary from 4 to 6 because of a small valence band splitting.

If **full ionization** is assumed, i.e. `energy-levels-relative = -1000.0`, then the degeneracy factor effectively becomes irrelevant. This can be seen from eqs. (1.4) - (1.7) in [\[BirnerPhD2011\]](#).

**Altermatt model for incomplete ionization**

The Altermatt model for incomplete ionization is described in [\[AltermattI2006\]](#) and [\[AltermattII2006\]](#). We need 4 parameters ( $N_{\text{ref}}$ ,  $c$ ,  $N_b$ ,  $d$ ) to calculate the incomplete ionization. If all of these 4 parameters are specified, then the Altermatt model is used. If these parameters are not specified, the default model for incomplete ionization is used (Eqs. (1.4) - (1.7) in [\[BirnerPhD2011\]](#)).

**incomplete-ionization-N-ref****type**

double array

**presence**

optional

**unit** $10^{18} \text{ cm}^{-3}$ **example**

2.2

Parameter  $N_{\text{ref}}$ **incomplete-ionization-c****type**

double array

**presence**

optional

**example**

2

Parameter  $c$ **incomplete-ionization-N-b****type**

double array

**presence**

optional

**unit** $10^{18} \text{ cm}^{-3}$ **example**

6.0

Parameter  $N_b$ **incomplete-ionization-d**

**type**

double array

**presence**

optional

**example**

2.3

Parameter  $d$ 

Our default model assumes that

$$E_D = E_c - E_D^{\text{ion}}$$

is the energetic position of the donor level, and

$$E_A = E_v + E_A^{\text{ion}}$$

is the energetic position of the acceptor level. In the Altermatt model, these energies are modified,

$$E_D^{\text{Alt}} = E_c - E_D^{\text{ion,Alt}},$$

$$E_A^{\text{Alt}} = E_v + E_A^{\text{ion,Alt}},$$

because we assume that the dopant density of states (DOS) peaks at the energy  $E_D^{\text{ion,Alt}}$  (for donors) and at  $E_A^{\text{ion,Alt}}$  (for acceptors). To calculate this modified energy of the dopant level, we use

$$E_D^{\text{ion,Alt}} = \frac{E_D^{\text{ion}}}{1 + (N_D/N_{\text{ref}})^c}, \text{ and}$$

$$E_A^{\text{ion,Alt}} = \frac{E_A^{\text{ion}}}{1 + (N_A/N_{\text{ref}})^c},$$

where  $E_D^{\text{ion}}$  and  $E_A^{\text{ion}}$  are the (usual) dopant energy at low doping concentration  $N_D$  for donors (or  $N_A$  for acceptors) specified via `doping-concentration`, i.e. the activation energy of the donor level  $E_D^{\text{ion}}$  or acceptor level  $E_A^{\text{ion}}$  specified via `energy-levels-relative`.

The decreasing fraction of the bound states is described by

$$b = \frac{1}{1 + (N_{\text{dop}}/N_b)^d},$$

where  $N_{\text{dop}}$  is the dopant concentration ( $N_D$  for donors and  $N_A$  for acceptors).

Then the ionized donors and acceptors are calculated as follows,

$$N_D^+ = N_D \left( 1 - \frac{b}{1 + g' e^{-(E_F - E_D^{\text{Alt}})/kT}} \right),$$

$$N_A^+ = N_A \left( 1 - \frac{b}{1 + g' e^{-(E_A^{\text{Alt}} - E_F)/kT}} \right),$$

where  $E_D^{\text{Alt}}$  and  $E_A^{\text{Alt}}$  are the (modified) energetic (peak) positions of the donor or acceptor level.

In these equations, the factor  $g' = 1/g$  is the inverse of the degeneracy  $g$  (`degeneracy-of-energy-levels`), i.e. 1/2 for donors and 1/4 for acceptors.

For comparison, our default incomplete ionization model uses

$$N_D^+ = N_D \frac{1}{1 + g e^{(E_F - E_D)/kT}},$$

$$N_A^+ = N_A \frac{1}{1 + g e^{(E_A - E_F)/kT}},$$

where the degeneracy factor  $g$  (`degeneracy-of-energy-levels`) is 2 for donors and 4 for acceptors.

**Interlevel transition times** (*not implemented yet*)

**transition-times-cb-to-levels****type**

double array

**presence**  
optional

**unit**  
?

Transition times tau1, tau2, tau3, ... from conduction band(s) to energy levels; required in case of trap: times from conduction band to discrete levels.

#### transition-times-levels-to-vb

**type**  
double array

**presence**  
optional

**unit**  
?

Transition times tau1, tau2, tau3, ... from energy levels to valence bands band(s); required in case of trap: times from discrete levels to valence bands.

#### Note

Currently no interlevel transition times implemented. Can be added provided there are also models which can handle such things.

#### Example

```
$impurity-parameters

!-----
! n-type in GaAs
!-----
impurity-number      = 1
impurity-name        = n-Si-in-GaAs
impurity-type        = n-type
number-of-energy-levels = 1
energy-levels-relative = 0.0058
degeneracy-of-energy-levels = 2      ! 2 for n-type

!-----
! n-type in GaAs (fully ionized)
!-----
impurity-number      = 2
impurity-name        = n-fully-ionized
impurity-type        = n-type
number-of-energy-levels = 1
energy-levels-relative = -1000.0    ! fully ionized
degeneracy-of-energy-levels = 2      ! 2 for n-type

!-----
! p-type in GaAs
!-----
impurity-number      = 3
impurity-name        = p-C-in-GaAs
impurity-type        = p-type
number-of-energy-levels = 1
energy-levels-relative = 0.027
```

(continues on next page)

(continued from previous page)

```

degeneracy-of-energy-levels = 4          ! 4 for p-type

!-----
! p-type in GaAs (fully ionized)
!-----
impurity-number             = 2
impurity-name               = p-fully-ionized
impurity-type               = p-type
number-of-energy-levels     = 1
energy-levels-relative      = -1000.0   ! fully ionized
degeneracy-of-energy-levels = 4          ! 4 for n-type

$end_impurity-parameters

```

Database values

```

energy-levels-relative = ...           ! energy in units of [eV]
                        = -1000.0      ! a large negative value implies full_
↪ ionization
                        = 0.054        ! n-As-in-Si
                        = 0.045        ! n-P -in-Si
                        = 0.039        ! n-Sb-in-Si
                        = 0.045        ! n-N -in-Si
                        = 0.006        ! n-Si-in-Al0.27Ga0.73As
                        = 0.0058       ! n-Si-in-GaAs
                        = 0.007        ! n-Si-in-AlAs
                        = 0.10         ! n-N -in-SiC
                        = 0.20         ! p-Al-in-SiC
                        = 0.045        ! p-B -in-Si
                        = 0.16         ! p-In-in-Si
                        = 0.027        ! p-C -in-GaAs

```

More parameters can be found in the database file `database_nn3.in` or at this website: <http://www.ioffe.ru/SVA/NSM/Semicond/>

### 5.3.9 \$material-interfaces

To specify additional charges at material interfaces, one has to specify

- material interfaces
- interface state properties.

See also documentation for keyword *\$interface-states*.

|                                  |                      |                 |
|----------------------------------|----------------------|-----------------|
| <b>\$material-interfaces</b>     |                      | <b>optional</b> |
| interface-number                 | <b>integer</b>       | <b>required</b> |
| apply-between-material-numbers   | <b>integer_array</b> | <b>required</b> |
| state-numbers                    | <b>integer_array</b> | <b>required</b> |
| <b>\$end_material-interfaces</b> |                      | <b>optional</b> |

Explanation of specifiers

**interface-number**

**type**

integer >= 1

**presence**

required

An integer number to refer to interface number. Dense numbering (1, 2, ...) as usual.

#### **apply-between-material-numbers**

**type**  
integer >= 1

**presence**  
required

Two integer numbers to refer to interface between geometry clusters, i.e. it contains the *material numbers* of the adjoining regions.

Is 3 4 equivalent to 4 3 in 1D? This becomes relevant for 2D and 3D simulations.

#### **state-numbers**

**type**  
integer >= 1

**presence**  
required

Identification numbers of interface states (e.g. fixed charge, trap or electrolyte) defined under keyword *\$interface-states*.

#### **Note**

There can only be *one* unique interface-number for the the interface between integer1 and integer2.

```
interface-number          = 1
apply-between-material-numbers = integer1 integer2
```

Here is an example which does not make sense (!):

```
interface-number          = 1
apply-between-material-numbers = 3 4

interface-number          = 2
apply-between-material-numbers = 3 4 ! '3 4' has already been assigned
                                   ! in 'interface-number = 1'
```

### 5.3.10 \$magnetic-field

One can apply a magnetic field only for 2D and 3D simulations. For 1D simulations this is not possible.

|                                      |                      |                 |
|--------------------------------------|----------------------|-----------------|
| <b>\$magnetic-field</b>              |                      | <b>optional</b> |
| magnetic-field-on                    | <b>character</b>     | <b>required</b> |
| magnetic-field-strength              | <b>double</b>        | <b>required</b> |
| magnetic-field-direction             | <b>integer_array</b> | <b>required</b> |
| magnetic-field-sweep-active          | <b>character</b>     | <b>optional</b> |
| magnetic-field-sweep-step-size       | <b>double</b>        | <b>optional</b> |
| magnetic-field-sweep-number-of-steps | <b>integer</b>       | <b>optional</b> |
| output-magnetic-vector-potential     | <b>character</b>     | <b>optional</b> |
| exclude-region-cluster-numbers       | <b>integer_array</b> | <b>optional</b> |
| <b>\$end_magnetic-field</b>          |                      | <b>optional</b> |

Example

```

!-----!
$magnetic-field
magnetic-field-on           = yes      ! yes/no
magnetic-field-strength     = 4.0      ! [T]
magnetic-field-direction    = 0 0 1    ! along z direction
!
output-magnetic-vector-potential = yes  ! yes/no
exclude-region-cluster-numbers = 2 3    !
$end_magnetic-field
!-----!

```

Here, a magnetic field of 4 T is applied.

#### magnetic-field-on

**type**  
character

**presence**  
required

**options**  
yes or no

**default**  
no

Flag for switching magnetic field *on* or *off*.

#### magnetic-field-strength

**type**  
double

**unit**  
(T)

**example**  
2.5

The magnetic field strength refers to the magnetic flux density **B** which has the SI unit (T) = (Vs/m<sup>2</sup>). Note that the SI unit for the magnetic field strength **H** is (A/m). It is possible to specify a negative value which inverts the vector specified here: `magnetic-field-direction`

#### magnetic-field-direction

**type**  
integer array of dimension 3

**example**  
0 0 1

The 3 (integer) indices  $x, y, z$  of a vector  $\mathbf{x} = (x, y, z)$  which is parallel to the magnetic field vector.

The direction of the magnetic field must be perpendicular to simulation orientation, e.g. if

```

!-----!
$simulation-dimension
dimension           = 2      !
orientation         = 0 1 1  ! simulation in (y,z) plane
$end_simulation-dimension
!-----!

```

then the magnetic field direction must be

```
magnetic-field-direction = 1 0 0 ! along x direction
```

For a 3D simulation, the magnetic field can have any direction. The vector  $\mathbf{B}$  must be specified with respect to the xyz simulation system, i.e. you cannot specify the four-digit Miller-Bravais indices as in the case for wurtzite. It is with respect to the x, y and z coordinate axes that were specified under *\$domain-coordinates*: If not specified, default values specified in database\_nn3.in are taken.

```
x y z := hkl-x-direction-zb, hkl-y-direction-zb, hkl-z-direction-zb

or

x y z := hkil-x-direction,  hkil-y-direction,  hkil-z-direction
```

(optional, only needed for magnetic field sweep)

#### output-magnetic-vector-potential

**type**  
character

**options**  
yes or no

**default**  
no

The magnetic vector potential  $\mathbf{A}(x, y, z)$  is defined with respect to the *symmetric gauge*:  $\mathbf{A}(\mathbf{r}) = -1/2(\mathbf{r} \times \mathbf{B})$ .

#### exclude-region-cluster-numbers

**type**  
integer array

**example**  
2 4

Here, cluster numbers 2 and 4 are exempted from magnetic field, i.e.  $\mathbf{B}=0$ . This is useful for Aharonov–Bohm effect (*not fully implemented yet*).

### Magnetic field sweep

It is possible to sweep over the magnetic field strength, i.e. to vary the strength of the magnetic field stepwise. This is similar to electric field sweeps (*\$electric-field*), voltage sweeps (*\$voltage-sweep*) and doping concentration sweeps (*\$doping-function*).

#### magnetic-field-sweep-active

**type**  
character

**options**  
yes or no

**default**  
no

Specify *yes* if you want to sweep over several values of the magnetic field.

#### magnetic-field-sweep-size

**type**  
double

**example**  
0.5

**example**  
(T)

Here, the magnetic field increases in steps of 0.5 (T). A negative value can be used to decrease the magnetic field.

#### magnetic-field-sweep-number-of-steps

**type**  
integer

**example**  
10

Specify here the number of steps for the magnetic field sweep.

Example

```
!-----!
$Magnetic-field
magnetic-field-on           = yes      ! yes/no
magnetic-field-strength     = 0.0      ! [T]
magnetic-field-direction    = 0 0 1    ! along z direction
!
magnetic-field-sweep-active  = yes      ! yes/no
magnetic-field-sweep-step-size = 0.5    ! [T]
magnetic-field-sweep-number-of-steps = 10
$end_magnetic-field
!-----!
```

Here, the magnetic field increases from 0 (T) to 5 (T) in steps of 0.5 (T), i.e. 11 simulations are performed.

The output is labeled with `_ind000.dat`, `_ind001.dat`, `_ind002.dat`, ... where the index refers to the number of the magnetic field sweep step.

The output for the eigenvalues as a function of applied magnetic field can be found here: `magnetic_ev_vb1_qc1_sg1_deg1.dat`.

In this particular example, the heavy hole valence band edge energies (vb1) have been written out as a function of magnetic field. The first column contains the strength of the magnetic field in units of (T). The second column contains the 1<sup>st</sup> eigenvalue for the specified electric field in units of (eV), the third column contains the 2<sup>nd</sup> eigenvalue for the specified electric field in units of (eV), ...

#### **Note**

The magnetic field is only implemented for the single-band effective mass model but not for the multi-band  $\mathbf{k} \cdot \mathbf{p}$  model. For the combination of  $\mathbf{k} \cdot \mathbf{p}$  and magnetic field, please use *nextnano++*.

### 5.3.11 \$Auger-recombination

Any values specified here in the input file overwrite the values specified in the material database.

Please refer to the database section of *\$Auger-recombination* for more details on the meaning of these parameters.

| \$Auger-recombination |           | optional |                                      |
|-----------------------|-----------|----------|--------------------------------------|
| material-name         | character | required |                                      |
| number-of-parameters  | integer   | required |                                      |
| n-C                   | double    | optional | ! [cm <sup>6</sup> /s] for electrons |
| p-C                   | double    | optional | ! [cm <sup>6</sup> /s] for holes     |
| n-bow-C               | double    | optional | ! [cm <sup>6</sup> /s] for electrons |

(continues on next page)



(continued from previous page)

|                                  |        |          |                                  |
|----------------------------------|--------|----------|----------------------------------|
| p-bow-C                          | double | optional | ! [cm <sup>6</sup> /s] for holes |
| <b>\$end_Auger-recombination</b> |        | optional |                                  |

### 5.3.12 \$direct-recombination

Any values specified here in the input file overwrite the values specified in the material database.

Please refer to the database section of *\$direct-recombination* for more details on the meaning of these parameters.

|                                   |           |          |                        |
|-----------------------------------|-----------|----------|------------------------|
| <b>\$direct-recombination</b>     |           | optional |                        |
| material-name                     | character | required |                        |
| number-of-parameters              | integer   | required |                        |
| C-opt                             | double    | optional | ! [cm <sup>3</sup> /s] |
| bow-C-opt                         | double    | optional | ! [cm <sup>3</sup> /s] |
| <b>\$end_direct-recombination</b> |           | optional |                        |

### 5.3.13 \$SRH-recombination

Any values specified here in the input file overwrite the values specified in the material database.

Please refer to the database section of *\$SRH-recombination* for more details on the meaning of these parameters.

|                                |           |          |                                     |
|--------------------------------|-----------|----------|-------------------------------------|
| <b>\$SRH-recombination</b>     |           | optional |                                     |
| material-name                  | character | required |                                     |
| number-of-parameters           | integer   | required |                                     |
| n-N-ref                        | double    | optional | ! [cm <sup>-3</sup> ] for electrons |
| n-tau                          | double    | optional | ! [s] for electrons                 |
| p-N-ref                        | double    | optional | ! [cm <sup>-3</sup> ] for holes     |
| p-tau                          | double    | optional | ! [s] for holes                     |
| n-bow-N-ref                    | double    | optional | ! [cm <sup>-3</sup> ] for electrons |
| n-bow-tau                      | double    | optional | ! [s] for electrons                 |
| p-bow-N-ref                    | double    | optional | ! [cm <sup>-3</sup> ] for holes     |
| p-bow-tau                      | double    | optional | ! [s] for holes                     |
| <b>\$end_SRH-recombination</b> |           | optional |                                     |

### 5.3.14 \$mobility-model-constant

See also explanations under section database ==> *\$mobility-model-constant*.

The only difference is that number-of-parameters is optional instead of required.

|                                      |         |          |  |
|--------------------------------------|---------|----------|--|
| <b>\$mobility-model-constant</b>     |         | optional |  |
| ...                                  |         |          |  |
| number-of-parameters                 | integer | optional |  |
| ...                                  |         |          |  |
| <b>\$end_mobility-model-constant</b> |         | optional |  |

### 5.3.15 \$mobility-model-minimos

See also explanations under section database ==> *\$mobility-model-minimos*.

The only difference is that `number-of-parameters` is optional instead of required.

|                                     |                |                 |
|-------------------------------------|----------------|-----------------|
| <b>\$mobility-model-minimos</b>     |                | <b>optional</b> |
| ...                                 |                |                 |
| number-of-parameters                | <b>integer</b> | <b>optional</b> |
| ...                                 |                |                 |
| <b>\$end_mobility-model-minimos</b> |                | <b>optional</b> |

### 5.3.16 \$mobility-model-simba

See also explanations under section database ==> *\$mobility-model-simba*.

The only difference is that `number-of-parameters` is optional instead of required.

|                                   |                |                 |
|-----------------------------------|----------------|-----------------|
| <b>\$mobility-model-simba</b>     |                | <b>optional</b> |
| ...                               |                |                 |
| number-of-parameters              | <b>integer</b> | <b>optional</b> |
| ...                               |                |                 |
| <b>\$end_mobility-model-simba</b> |                | <b>optional</b> |

### 5.3.17 \$mobility-model-arora

See also explanations under section database ==> *\$mobility-model-arora*.

The only difference is that `number-of-parameters` is optional instead of required.

|                                   |                |                 |
|-----------------------------------|----------------|-----------------|
| <b>\$mobility-model-arora</b>     |                | <b>optional</b> |
| ...                               |                |                 |
| number-of-parameters              | <b>integer</b> | <b>optional</b> |
| ...                               |                |                 |
| <b>\$end_mobility-model-arora</b> |                | <b>optional</b> |

### 5.3.18 \$mobility-model-dar

See also explanations under section database ==> *\$mobility-model-dar*.

The only difference is that `number-of-parameters` is optional instead of required.

|                                 |                |                 |
|---------------------------------|----------------|-----------------|
| <b>\$mobility-model-dar</b>     |                | <b>optional</b> |
| ...                             |                |                 |
| number-of-parameters            | <b>integer</b> | <b>optional</b> |
| ...                             |                |                 |
| <b>\$end_mobility-model-dar</b> |                | <b>optional</b> |

### 5.3.19 \$mobility-model-lom

See also explanations under section database ==> *\$mobility-model-lom*.

The only difference is that `number-of-parameters` is optional instead of required.

|                                 |                |                 |
|---------------------------------|----------------|-----------------|
| <b>\$mobility-model-lom</b>     |                | <b>optional</b> |
| ...                             |                |                 |
| number-of-parameters            | <b>integer</b> | <b>optional</b> |
| ...                             |                |                 |
| <b>\$end_mobility-model-lom</b> |                | <b>optional</b> |

### 5.3.20 \$mobility-model-masetti

See also explanations under section database ==> *\$mobility-model-masetti*.

The only difference is that `number-of-parameters` is optional instead of required.

|                                     |                |                 |
|-------------------------------------|----------------|-----------------|
| <b>\$mobility-model-masetti</b>     |                | <b>optional</b> |
| ...                                 |                |                 |
| number-of-parameters                | <b>integer</b> | <b>optional</b> |
| ...                                 |                |                 |
| <b>\$end_mobility-model-masetti</b> |                | <b>optional</b> |

### 5.3.21 \$Optical-absorption

- *Optical absorption*
- *Solar cells*
  - *Solar spectrum*
  - *Absorption Spectra*
  - *Reflection coefficient*
  - *Transmission coefficient*
  - *Solar cell output*
- *Black body spectrum*
- *Intraband absorption in the single-band case*

This keyword allows calculating optical absorption coefficient and solar cells.

|                             |                     |                 |
|-----------------------------|---------------------|-----------------|
| <b>\$Optical-absorption</b> |                     | <b>optional</b> |
| destination-directory       | <b>character</b>    | <b>required</b> |
| calculate-optics            | <b>character</b>    | <b>optional</b> |
| kind-of-absorption          | <b>character</b>    | <b>optional</b> |
| read-in-k-points            | <b>character</b>    | <b>optional</b> |
| num-quantum-cluster         | <b>integer</b>      | <b>optional</b> |
| e-min-state                 | <b>double</b>       | <b>optional</b> |
| e-max-state                 | <b>double</b>       | <b>optional</b> |
| e-min-photon                | <b>double</b>       | <b>optional</b> |
| e-max-photon                | <b>double</b>       | <b>optional</b> |
| num-energy-steps            | <b>integer</b>      | <b>optional</b> |
| smoothing-of-curve          | <b>character</b>    | <b>optional</b> |
| smoothing-damping-parameter | <b>double</b>       | <b>optional</b> |
| E-P                         | <b>double</b>       | <b>optional</b> |
| polarization-vector-1       | <b>double_array</b> | <b>optional</b> |
| polarization-vector-2       | <b>double_array</b> | <b>optional</b> |
| magnitude-relation-1-2      | <b>double</b>       | <b>optional</b> |
| phase                       | <b>double</b>       | <b>optional</b> |
| fermi_in_el                 | <b>double</b>       | <b>optional</b> |
| fermi_in_hl                 | <b>double</b>       | <b>optional</b> |
| device_thickness_in         | <b>double</b>       | <b>optional</b> |
| k-space-symmetry            | <b>character</b>    | <b>optional</b> |

(continues on next page)

(continued from previous page)

```

!-----
! The following are only relevant for solar cell simulations.
!-----

incident-light-along-direction    character    optional

import-absorption-spectrum       character    optional
file-absorption-spectrum        character    optional

import-reflectivity-spectrum     character    optional
file-reflectivity-spectrum      character    optional

import-transmission-spectrum     character    optional
file-transmission-spectrum      character    optional

import-solar-spectrum           character    optional
file-solar-spectrum            character    optional

number-of-suns                  double        optional

calculate-black-body-spectrum   character    optional
$end_optical-absorption         optional

```

A tutorial is available that describes this keyword: [Optical absorption of an InGaAs quantum well](#)

Detailed description about the Physics: [Absorption, Matrix elements, Inter-band transitions, Intra-band transitions](#) (pdf).

#### **destination-directory**

##### **type**

character

##### **presence**

required

##### **example**

optics/

Directory for output of data files.

### **Optical absorption**

#### **calculate-optics**

##### **type**

character

##### **options**

yes or no

##### **default**

no

Choose yes if you want to calculate the optical absorption spectrum (Step 3).

This flag can be set to no for Step 1 and Step 2, and yes for Step 3 (*see below*).

#### **num-quantum-cluster**

##### **type**

integer >= 1

**default**

1

Number of quantum cluster for which absorption spectrum is calculated. If this specifier is not present, the quantum cluster 1 is taken.

**kind-of-absorption****type**

character

**options**

interband-only   intra-vb-only   intra-cb-only   intra-sg-only  
inter-sg-only

• **interband-only**

Considers only interband transitions between holes and electrons.

- heavy hole <==> Gamma band
- light hole <==> Gamma band
- split-off hole <==> Gamma band

• **intra-vb-only**

Considers only intraband transitions within the valence bands.

- heavy hole <==> light hole
- heavy hole <==> split-off hole
- light hole <==> split-off hole
- heavy hole <==> heavy hole
- light hole <==> light hole
- split-off hole <==> split-off hole

• **intra-cb-only**

Considers only intraband transitions within the conduction band (Gamma band).

- Gamma band <==> Gamma band

• **intra-sg-only**

Considers only intraband transitions within the same band (single-band for Gamma, L, X, heavy hole, light hole, split-off hole band)

- Gamma band <==> Gamma band
- L band <==> L band
- X band <==> X band
- heavy hole <==> heavy hole
- light hole <==> light hole
- split-off hole <==> split-off hole

This is a simple algorithm taking only account the energy levels and wave functions at  $k_{\parallel} = 0$  (for single-band case). It only works for 1D and 2D simulations so far. It can also be used for the  $\mathbf{k} \cdot \mathbf{p}$  wave functions as shown in this tutorial: *Intersubband transitions in InGaAs/AlInAs multiple quantum well systems*. In this case, the correct  $\mathbf{k} \cdot \mathbf{p}$  density and k-dependent matrix elements and nonparabolicity and anisotropy of the energy dispersion  $E(k_x, k_y)$  is taken into account.

In order for this flag to work, the following must be present in the input file (*\$output-1-band-schroedinger*):

```
$output-1-band-schroedinger
...
intraband-matrixelements = yes ! or any other value apart from 'no'
```

The equation used is described here: *Intraband absorption in the single-band case*

- **inter-sg-only**

Similar as **intra-sg-only** but for interband transitions. It currently does not work for  $\mathbf{k} \cdot \mathbf{p}$  wave functions.

Specifications for energy range of absorption spectrum: lower/upper boundary for photon energy interval

#### **e-min-photon**

```
type
    double

unit
    (eV)

example
    1.0
```

lower boundary for photon energy

#### **e-max-photon**

```
type
    double

unit
    (eV)

example
    2.0
```

upper boundary for photon energy

#### **num-energy-steps**

```
type
    integer

example
    1000
```

Number of energy steps between **e-min-photon** and **e-max-photon**.

This number determines the resolution of the absorption spectrum curve  $\alpha(E)$  where  $E$  is the energy in units of (eV).

#### **Note**

The number of

energy grid points = **num-energy-steps** + 1

because the first grid point (**e-min-photon**) is also included.

The energy grid spacing is

$$\Delta_E = (\text{e-max-photon} - \text{e-min-photon}) / \text{num-energy-steps}.$$

Distinguish between calculating and reading in  $k_{\parallel}$  points.

#### **read-in-k-points**

**type**  
character

**options**  
yes or no

**default**  
no

Flag to distinguish between Step 2 and Step 3. For Step 3, in order to avoid calculating the  $k_{\parallel}$  points again, one can simply read them in from a previous simulation, and then calculate and output the optical absorption spectrum.

Energy of lowest/highest eigenvalue considered for calculation, i.e. energies are calculated in the interval [e-min-state,e-max-state]. Here, the eigenvalue solver is called with these energy values. Alternatively, the eigenvalue solver can be called with a certain number of eigenvalues requested, i.e. one either has to specify an energy interval or the number of eigenvalues. Depending on the eigenvalue solver used, different options are possible.

**e-min-state**

**type**  
double

**unit**  
(eV)

**default**  
-5.0

**example**  
-1.7

lowest eigenvalue

**e-max-state**

**type**  
double

**unit**  
(eV)

**default**  
5.0

**example**  
0.3

highest eigenvalue

### Broadening of absorption curve

This is only relevant for the calculation of the absorption spectrum.

**smoothing-of-curve**

**type**  
character

**options**  
Lorentzian, Gaussian, yes, no

**default**  
yes

For Lorentzian or Gaussian, we introduce an artificial broadening (smoothing) of the curve (*Lorentzian* or *Gaussian* broadening). If yes, both *Lorentzian* and *Gaussian* broadening will be calculated and written out. If no, no broadening (smoothing) of the curve is assumed.

**smoothing-damping-parameter**

**type**  
double > 0.0

**unit**  
(eV)

**default**  
0.005

The artificial parameter for smoothing of absorption spectrum is `smoothing-damping-parameter`. It is usually denoted as  $\Gamma$  and is the *Full Width at Half Maximum (FWHM)*.

#### *Lorentzian lineshape*

The Lorentzian function is given by

$$L(E) = \frac{1}{\pi} \frac{\Gamma/2}{(E-E_{ij})^2 + (\Gamma/2)^2}$$

where  $\Gamma/2$  is the scale parameter *Lorentzian half-width*, i.e. Half Width at Half Maximum (**HWHM**). It describes the shape of certain types of spectral lines (lineshape). Note that the definition of the Lorentzian function includes a factor  $1/\pi$ .

- $E_{ij}$  is the transition energy between the states  $i$  and  $j$  and specifies the location of the peak in the Lorentzian function.
- $\Gamma$  is specified in the input file via `smoothing-damping-parameter`. It is the *Full Width at Half Maximum (FWHM)*.
- $\alpha = \Gamma/2$  is the *Half Width at Half Maximum (HWHM)*.

```
!-----  
! for k.p algorithm only:  
! First, the absorption spectrum is calculated.  
! Then the broadening is applied.  
!-----  
  
absorption_NoSmoothingV = absorptionV  
absorptionV = 0.0  
DO i=1,num-energy-steps+1 ! Loop over all energy grid points E(i) and_  
↪determine absorption coefficients alpha(i)=alpha(E).  
  DO j=1,num-energy-steps+1 ! This loop is essentially an integration_  
↪over energy dE.  
  
    E_weight = (Lorentzian( E_gridV(j), E_gridV(i), smoothing-damping-  
↪parameter ) - &  
                Lorentzian( E_gridV(j), - E_gridV(i), smoothing-damping-  
↪parameter ) ) * DeltaEnergy  
  
    absorptionV(i) = absorptionV(i) + absorption_NoSmoothingV(j) * E_  
↪weight  
  
  END DO  
END DO
```

The following specifiers are only used for the  $\mathbf{k} \cdot \mathbf{p}$  optical absorption spectrum but not for the simple single-band intersubband absorption algorithm.

#### **E\_P**

**type**  
double

**example**  
20.0



**unit**

(eV)

**status***currently not implemented, value from the database is used*

$E_P$  is Kane's matrix element  $E_P = |\langle S|p|X \rangle|^2$ . It should be around 20 eV and depends on the material. The  $E_P$  parameter is given in the database by the specifier `8x8kp-parameters`.  $E_P$  can be converted into the  $P$  parameter by the following equation:  $E_P = \frac{2m_0}{\hbar^2} P^2$ . In our model the  $E_P$  parameter is only relevant for interband transitions. It enters into the matrix element prefactor (*matrix\_element\_prefac*) which is described in Section 1.1.1 *Inter-band transitions* of the documentation: [Absorption spectrum](#), [Matrix elements](#), [Inter-band transitions](#), [Intra-band transitions](#) (pdf).  $E_P$  has the same value for all materials in this implementation. In principle it could have been read in from the database rather than specifying it within the keyword `$optical-absorption`.

**Polarization**

$m$  is equivalent to  $\tan(\theta)$  or  $\tan^{-1}(\theta)$  depending whether  $P_1/P_2$  is  $x$  or  $y$ . so instead of  $\theta$  one can directly use the formula below:

$$P = m \cdot P_1 + e^{i\omega\pi} \cdot P_2$$

This is in fact more general, as it also describes circular polarization which lead to complex coefficient.

**polarization-vector-1****type**

double array

**example**

1.0 0.0 0.0

x y z coordinates (in simulation system) for first in-plane vector  $P_1$

**polarization-vector-2****type**

double array

**example**

0.0 1.0 0.0

x y z coordinates (in simulation system) for second in-plane vector  $P_2$

**magnitude-relation-1-2****type**

double

**default**

0.5

**example**

1.0

relation of magnitudes  $m = |E_1|/|E_2|$

**phase****type**

double

**default**

0.0

**example**

0.5

phase  $\omega$ :  $E_2 \Rightarrow \exp(i\omega\pi)E_2$

$$P = m \cdot P_1 + e^{i\omega\pi} \cdot P_2$$

## Examples

- x-polarized light

```
polarization-vector-1 = 0.0 1.0 0.0
polarization-vector-2 = 1.0 0.0 0.0
magnitude-relation-1-2 = 0.0
```

In this case, polarization-vector-1 is ignored as  $|E_1|$  is set to be zero.

- z-polarized light

```
polarization-vector-1 = 1.0 0.0 0.0
polarization-vector-2 = 0.0 0.0 1.0
magnitude-relation-1-2 = 0.0
```

In this case, polarization-vector-1 is ignored as  $|E_1|$  is set to be zero.

- circularly polarized light in the (x,y) plane

```
polarization-vector-1 = 1.0 0.0 0.0
polarization-vector-2 = 0.0 0.0 1.0
magnitude-relation-1-2 = 1.0
```

In this case, polarization-vector-1 is *not* ignored as  $|E_1| = |E_2|$ .

- quantum well, interband absorption

```
polarization-vector-1 = 1.0 0.0 0.0
polarization-vector-2 = 0.0 1.0 0.0
magnitude-relation-1-2 = 1.0
```

- quantum well, intraband absorption

```
polarization-vector-1 = 0.0 0.0 1.0
polarization-vector-2 = 0.0 0.0 1.0
magnitude-relation-1-2 = 0.0
```

Note: Intraband absorption spectrum only for z-polarized light.

## Fermi levels

**fermi\_in\_el**

**type**  
double

**unit**  
(eV)

**default**  
0.0

**example**  
0.1

Optional input for Fermi level of electrons (default: calculated quasi-Fermi level for electrons)

**fermi\_in\_hl**

**type**  
double

**unit**  
(eV)

**default**  
0.0

**example**  
-1.0

Optional input for Fermi level of holes (default: calculated quasi-Fermi level for holes)

**device-thickness**

**type**  
double

**unit**  
(m)

**default**  
thickness of device or quantum cluster? (*Check this!*)

**example**  
1e-6

Optional input of device thickness for normalization of absorption spectrum

**k-space-symmetry**

**type**  
character

**options**  
default, none, four-fold

**default**  
default

Symmetry of  $k_{||} = (k_x, k_y)$  space to be discretized. If any symmetry is present, less  $k_{||}$  points have to be calculated. By default, the appropriate symmetry is chosen taking into account any crystal rotations with respect to the simulation axes, as well as nonsymmetric strains.

#### Note

In order to save CPU time, we recommend the following procedure:

Instead of calculating

- $\mathbf{k} \cdot \mathbf{p}$  eigenstates and
- optical absorption spectrum

within one simulation, it is more efficient to divide this into 3 steps.

1. Step 1: Calculate eigenstates for  $k_{||} = 0$ .

```
calculate-optics = no
```

This is very quick. Solve  $\mathbf{k} \cdot \mathbf{p}$  to determine lowest and highest eigenvalue so that we know what to specify for e-min-state and e-max-state in Step 2.

2. Step 2: Calculate eigenstates for all  $k_{||}$  vectors and save all of the wave functions to file so that they can be read in and used many times in Step 3.

```
raw-potential-in    = yes
raw-fermi-levels-in = yes
strain-calculation  = raw-strain-in
```

```
num-kp-parallel     = 1700 ! STEP 2/3    ! total number of k_|| points for_
↳ Brillouin zone discretization
```

```
calculate-optics    = yes
num-quantum-cluster = 1
```

**5.3. Keywords**

```
read-in-k-points    = no
e-min-state         = -1.7 ! Choose a reasonable value for E_min
e-max-state         = 0.3  ! Choose a reasonable value for E_min
```

Read in raw data (potential, quasi-Fermi levels, strain (if applicable) and all  $\mathbf{k} \cdot \mathbf{p}$  wave functions) and output  $k_{\parallel}$  points. The strain calculation is very fast for a 1D simulation. Instead of reading it in, it could be directly calculated (*recommended*).

The user specifies the total number of  $k_{\parallel}$  points that are present in  $k_{\parallel}$  space. However, internally the code modifies this number according to the following algorithm:

- number of k points in positive x direction (without Gamma point):  $N_{kx}$
- number of k points in positive y direction (without Gamma point):  $N_{ky} = N_{kx}$

==> Thus the actual, total number of  $k_{\parallel}$  points is:

$$\text{total\_number\_of\_k} = (2 * N_{kx} + 1) * (2 * N_{ky} + 1)$$

In this example (num-kp-parallel = 1700):

$$N_{kx} = N_{ky} = 20$$

$$\Rightarrow \text{total\_number\_of\_k} = 41 * 41 = 1681$$

### 3. Step 3: Calculate optical absorption spectrum.

```
calculate-optics = yes
read-in-k-points = yes
```

Read in  $k_{\parallel}$  points, calculate and output optical absorption spectrum for specific polarization of incident light.

If one wants to repeat the calculation for another polarization, one only needs to change the polarization vector and repeat Step 3. It is not necessary in this case to recalculate Step 1 or Step 2. Step 3 also outputs the energy dispersion  $E(k_{\parallel}) = E(k_x, k_y)$ . (*Check: Why not Step 2?*)

## Output

### Results

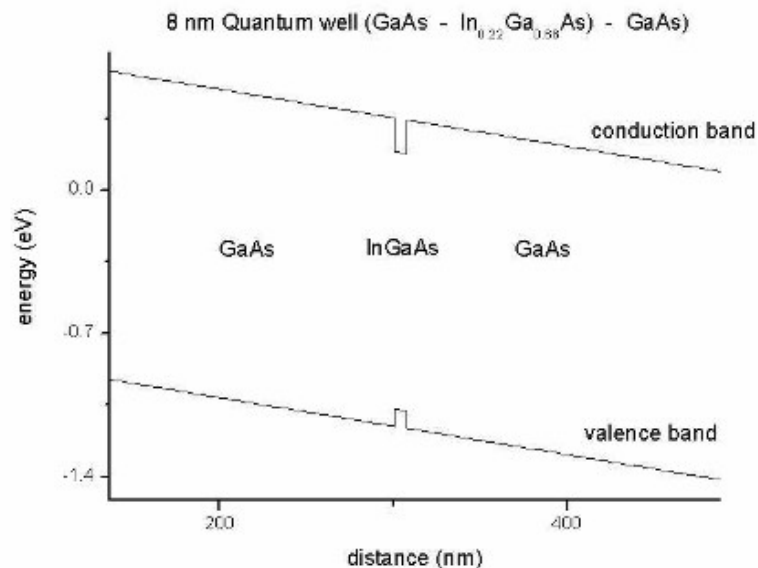


Figure 5.3.21.1: Conduction and valence band edges of the quantum well

The unit of the optical absorption coefficient is  $\text{m}^{-1}$  and not *arbitrary units* as indicated in the figure.

The *electric susceptibility tensor*  $\chi$  is contained in the file `susceptibility_tensor.dat`:

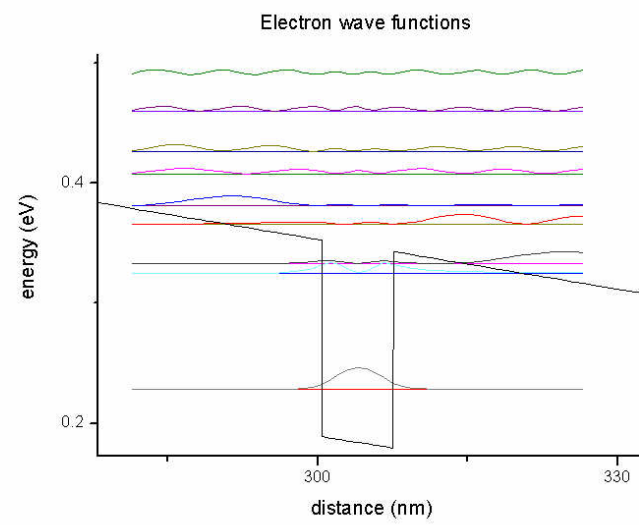


Figure 5.3.21.2: Electron states in the quantum well

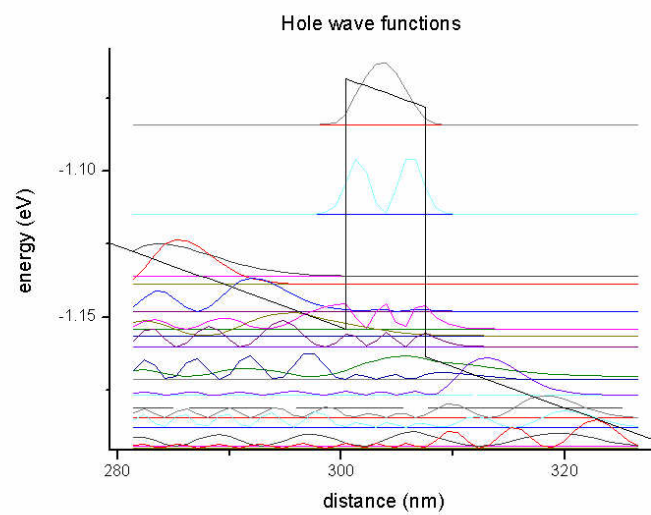


Figure 5.3.21.3: Hole states in the quantum well

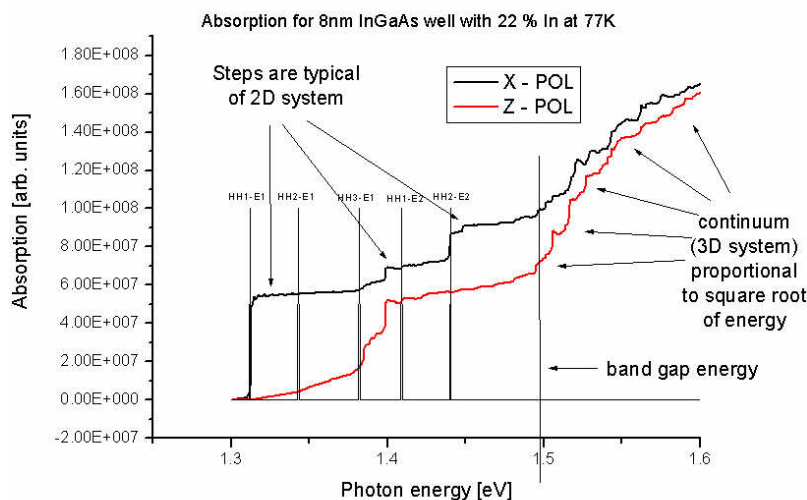


Figure 5.3.21.4: Optical absorption spectrum of the quantum well

```
chi11re chi11im chi22re chi22im chi33re chi33im chi12re chi12im ↵
↵chi13re chi13im chi23re chi23im
```

Note: As this tensor is complex, for each component, two values are written out.

- re: real part
- im: imaginary part

The relevant part for the absorption spectrum is only the *imaginary part*.

The units of the  $k_{\parallel}$  space grid coordinates  $k_x$  and  $k_y$  are ( $\text{\AA}^{-1}$ ) and the energy units are (eV).

The files

- el\_dispersion\_100.dat
- el\_dispersion\_110.dat
- hl\_dispersion\_100.dat
- hl\_dispersion\_110.dat

show the same data but with slices along the

- [10] (i.e.  $k_{\parallel} = (k_x, k_y = 0)$  and
- [11] (i.e.  $k_{\parallel} = (k_x = k_y)$  directions in  $k_{\parallel}$  space.

Here all electron and all hole eigenvalues are contained in one file, respectively.

### Restrictions

- Only *Dirichlet* boundary conditions are supported so far.
- Step 2 and Step 3 only work if:

```
raw-potential-in = yes
```

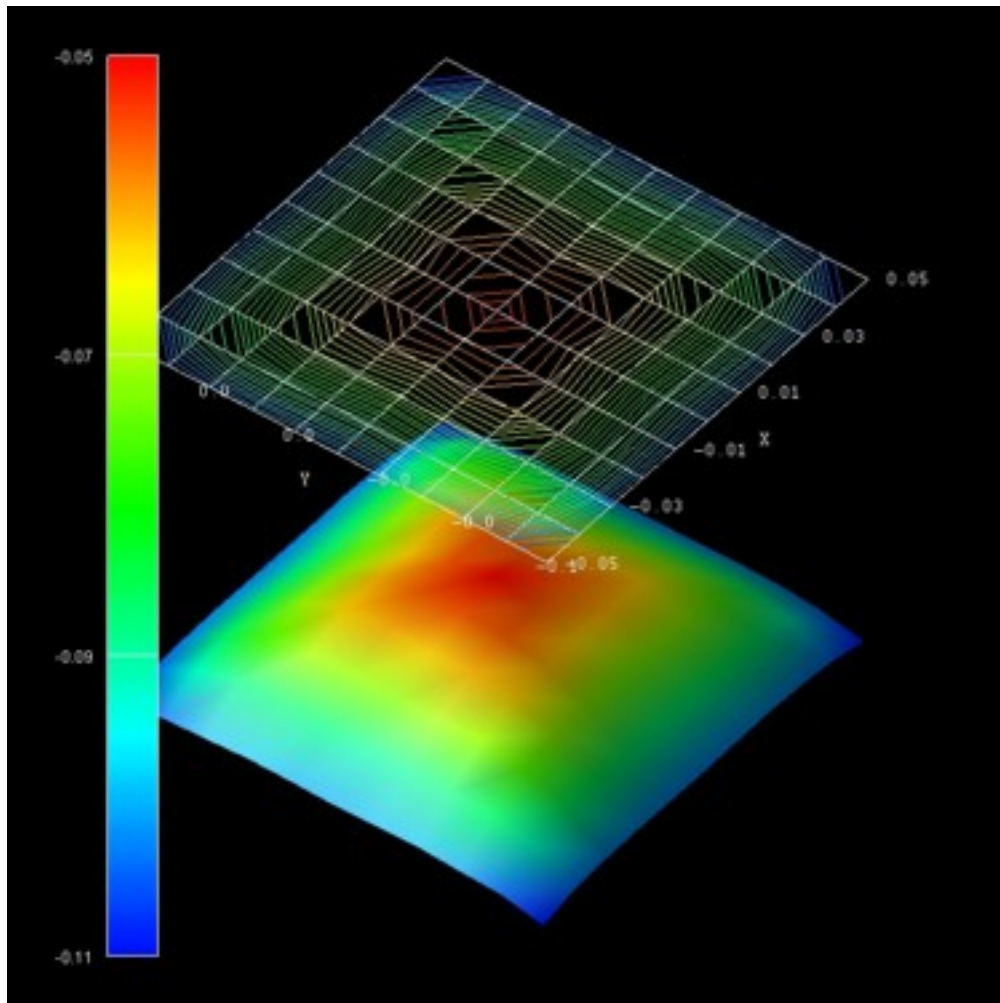


Figure 5.3.21.5: Energy dispersion  $E(k_x, k_y)$  of the highest hole eigenstate (ground state).

## Solar cells

For solar cells, we have this tutorial: *GaAs Solar Cell*

Example files for solar spectra, absorption coefficient, transmission and reflectivity coefficients can be found in the installation folder:

- C:\Program Files\nextnano\nextnano3\Syntax\Solar cell files\absorption\
- C:\Program Files\nextnano\nextnano3\Syntax\Solar cell files\reflectivity
- C:\Program Files\nextnano\nextnano3\Syntax\Solar cell files\solar spectra\

The following specifiers are relevant for solar cell simulations (photovoltaics).

### **incident-light-along-direction**

#### **type**

character

#### **options**

x, y, z, -x, -y, -z

#### **default**

along simulation direction in 1D

In a 1D simulation, this specifier is optional. For 2D and 3D, a direction must be specified.

## Solar spectrum

### **import-solar-spectrum**

#### **type**

character

#### **options**

yes or no

#### **default**

no

For a solar cell simulation, one has to read in a solar spectrum, e.g. AM 1.5, or AM 1.0 (AM = air mass). They can be obtained from NREL website, e.g. ASTM-E490: <https://www.nrel.gov/grid/solar-resource/spectra-astm-e490.html> (AMST = American Society for Testing and Materials)

### **file-solar-spectrum**

#### **type**

character

#### **example**

H:\solar\_cells\ASTMG173\_AM10.dat AM 1.0 spectrum (extraterrestrial)

#### **example**

H:\solar\_cells\ASTMG173\_AM15.dat AM 1.5 spectrum

#### **example**

H:\solar\_cells\ASTMG173\_AM15G.dat AM 1.5G spectrum (G = global, i.e. including diffuse light)

The file must consist of two columns (wavelength and spectrum), the units are (nm) and ( $\text{W}/\text{m}^2 \cdot \text{nm}^{-1}$ ).

|                |                                                      |
|----------------|------------------------------------------------------|
| wavelength[nm] | AM1.5 [ $\text{W}/\text{m}^2 \cdot \text{nm}^{-1}$ ] |
| ...            | ...                                                  |

Concentration of sun light

### **number-of-suns**



```

type
    integer
default
    1.0 our sun
example
    0.0 (no sun, dark)
example
    2.5 2.5 suns
example
    300.0 300 suns
example
    1000.0 100 suns

```

The number of suns can be set to increase the power of the solar spectrum in order to model concentrator solar cells.

## Absorption Spectra

```

import-absorption-spectrum
    type
        character
    options
        yes or no
    default
        no
file-absorption-spectrum
    type
        character
    example
        AbsorptionCoefficient_GaAs_300K.dat

```

The file must consist of two columns (wavelength and absorption coefficient), the units are (nm) and (cm<sup>-1</sup>).

|                |                  |
|----------------|------------------|
| wavelength[nm] | absorption[1/cm] |
| ...            | ...              |

## Reflection coefficient

Fraction of incident photons that are reflected from surface for a particular wavelength.

```

import-reflectivity-spectrum
    type
        character
    options
        yes or no
    default
        no
file-reflectivity-spectrum
    type
        character

```

**example**`ReflectionCoefficient_GaAs_300K.dat`

The file must consist of two columns, wavelength (nm) and reflection coefficient.

```
wavelength[nm]    reflectivity[]  
...               ...
```

**Transmission coefficient**

Fraction of incident photons that are transmitted through the device for a particular wavelength (relevant for very thin devices).

**import-transmission-spectrum****type**

character

**options**

yes or no

**default**

no

**file-transmission-spectrum****type**

character

**example**`TransmissionCoefficient.dat`

The file must consist of two columns, wavelength (nm) and transmission coefficient.

```
wavelength[nm]    transmission[]  
...               ...
```

**Solar cell output**

All output is twofold:

- one is with respect to wavelength in units of (nm)
- one is with respect to photon energy in units of (eV) (indicated by `_eV*.dat`)

The files are:

- Absorption coefficient
  - `optics/Absorption_coefficient.dat` (as read in from file but now in units of  $(m^{-1})$ )
  - `optics/Absorption_coefficient_interpolated.dat` (interpolated on wavelength grid of solar spectrum but now in units of  $(m^{-1})$ )
- Reflectivity
  - `optics/Reflectivity.dat` (as read in from file)
  - `optics/Reflectivity_interpolated.dat` (interpolated on wavelength grid of solar spectrum)
- Transmission
  - `optics/Transmission.dat` (as read in from file)
  - `optics/Transmission_interpolated.dat` (interpolated on wavelength grid of solar spectrum)
- Solar spectrum
  - `optics/SolarSpectralIrradiance_sun0001.dat` (as read in from file)

- optics/PhotonFlux\_sun0001.dat (photon flux density calculated from solar spectrum)
- Total number of photons in the solar spectrum above an energy value contributing to the maximum photocurrent for a solar cell made with a specific band gap:
  - optics/PhotonFlux\_BandGap\_eV\_sun0001.dat (calculated from solar spectrum)
  - optics/PhotoCurrent\_BandGap\_eV\_sun0001.dat (calculated from solar spectrum)
- Spectral response
  - optics/SpectralResponse\_sun0001.dat external and internal spectral response
- Quantum efficiency
  - optics/QuantumEfficiency\_sun0001.dat external and internal quantum efficiency
- Generation rate
  - optics/GenerationRateLight\_AVS\_sun0001.fld 2D plots  $G(x, \lambda)$  and  $G(x, E)$
  - optics/GenerationRateLight\_sun0001.dat 1D plot  $G(x)$
  - optics/GenerationRate\_eV\_sun0001.dat 1D plot  $G(E)$  where  $E$  is the energy
  - optics/GenerationRate\_Wavelength\_sun0001.dat 1D plot  $G(\lambda)$
- Current-voltage characteristics
  - current/IV\_characteristics\_new.dat

```
voltage[V]   current[A/m^2] ... power[W/m^2] powersolar[W/m^2] efficiency[%]
```

- The following information can be found in the .log file, such as
  - short-circuit current  $I_{sc}$
  - open-circuit voltage  $U_{oc}$
  - ideal conversion efficiency  $\eta$
  - ...

```
*****
Solar cell results
*****
short-circuit current:      I_sc      = 281.473346 [A/m^2]
↳(photo current: It increases with smaller band gap.)
open-circuit voltage:      U_oc      = -1.012500 [V]          (U_oc
↳<= built-in potential ~ band gap)
current at maximum power:  I_max     = 273.089897 [A/m^2]
voltage at maximum power:  U_max     = -0.925000 [V]
maximum power output:      P_max     = U_max * I_max = -252.608155 [W/m^2]
↳(condition for maximum power output: dP/dV = 0)
maximum extracted power:   P_solar   = - P_max          = 252.608155 [W/m^2]
incident power:            P_in      = 0.000000 [W/m^2]
ideal conversion efficiency: eta      = P_max / P_in = Infinity %
fill factor:               FF        = 0.886370
In practice, a good fill factor is around 0.8.
All these results are approximations.
They are only correct if a lot of voltage steps have been used (i.e. a high
↳resolution).
*****
```

Example for a solar cell simulation

```

!-----
$optical-absorption
destination-directory      = optics/

import-absorption-spectrum = yes
file-absorption-spectrum   = "..\Syntax\Solar cell files\absorption\
↪AbsorptionCoefficient_GaAs_300K.dat"

import-reflectivity-spectrum = yes
file-reflectivity-spectrum  = "..\Syntax\Solar cell files\reflectivity\
↪Reflectivity_Al0.80Ga0.20As.dat"

import-solar-spectrum      = yes
file-solar-spectrum        = "..\Syntax\Solar cell files\solar spectra\
↪ASTMG173_AM15G.dat"

number-of-suns             = 1

$end_optical-absorption
!-----

```

## Black body spectrum

### calculate-black-body-spectrum

#### type

character

#### options

yes or no

#### default

no

Flag for calculating *black body spectrum* according to [Planck's law](#), e.g. to compare the solar spectrum to the spectrum of a black body at  $T = 5778$  K.

- The spectral *energy density*
- the spectral *radiance* (which is *emitted* per  $\text{m}^2$  and per unit solid angle sr (sr = steradian)) and
- the spectral *irradiance* (which is *received* per  $\text{m}^2$ )

is calculated.

### Note

spectral irradiance = spectral radiance  $\cdot \pi$

spectral energy density = spectral radiance  $\cdot 4\pi/c$

There are several output files, i.e. output with respect to

- *wavelength*  $\lambda$  in units of (nm),
  - BlackBody\_SpectralEnergyDensity\_wavelength.dat

|                |                                             |
|----------------|---------------------------------------------|
| Wavelength[nm] | SpectralEnergyDensity[kJ/m <sup>3</sup> /m] |
|----------------|---------------------------------------------|

- BlackBody\_SpectralRadiance\_wavelength.dat

|                |                                            |
|----------------|--------------------------------------------|
| Wavelength[nm] | SpectralRadiance[kW/m <sup>2</sup> /nm/sr] |
|----------------|--------------------------------------------|

– BlackBody\_SpectralIrradiance\_wavelength.dat

|                |                                           |
|----------------|-------------------------------------------|
| Wavelength[nm] | SpectralIrradiance[kW/m <sup>2</sup> /nm] |
|----------------|-------------------------------------------|

- *angular frequency*  $\omega = 2\pi\nu$  in units of (1/s),

– BlackBody\_SpectralEnergyDensity\_angular\_frequency.dat

|                                             |                                                                             |
|---------------------------------------------|-----------------------------------------------------------------------------|
| AngularFrequency_omega[10 <sup>15</sup> /s] | SpectralEnergyDensity[10 <sup>-15</sup> J/m <sup>3</sup> /s <sup>-1</sup> ] |
|---------------------------------------------|-----------------------------------------------------------------------------|

– BlackBody\_SpectralRadiance\_angular\_frequency.dat

|                                             |                                                                           |
|---------------------------------------------|---------------------------------------------------------------------------|
| AngularFrequency_omega[10 <sup>15</sup> /s] | SpectralRadiance[10 <sup>-12</sup> W/m <sup>2</sup> /s <sup>-1</sup> /sr] |
|---------------------------------------------|---------------------------------------------------------------------------|

– BlackBody\_SpectralIrradiance\_angular\_frequency.dat

|                                             |                                                                          |
|---------------------------------------------|--------------------------------------------------------------------------|
| AngularFrequency_omega[10 <sup>15</sup> /s] | SpectralIrradiance[10 <sup>-12</sup> W/m <sup>2</sup> /s <sup>-1</sup> ] |
|---------------------------------------------|--------------------------------------------------------------------------|

- *frequency*  $\nu$  in units of (Hz),

– BlackBody\_SpectralEnergyDensity\_frequency.dat

|                |                                                               |
|----------------|---------------------------------------------------------------|
| Frequency[THz] | SpectralEnergyDensity[10 <sup>-15</sup> J/m <sup>3</sup> /Hz] |
|----------------|---------------------------------------------------------------|

– BlackBody\_SpectralRadiance\_frequency.dat

|                |                                                          |
|----------------|----------------------------------------------------------|
| Frequency[THz] | SpectralRadiance[10 <sup>-12</sup> W/m <sup>2</sup> /sr] |
|----------------|----------------------------------------------------------|

– BlackBody\_SpectralIrradiance\_frequency.dat

|                |                                                            |
|----------------|------------------------------------------------------------|
| Frequency[THz] | SpectralIrradiance[10 <sup>-12</sup> W/m <sup>2</sup> /Hz] |
|----------------|------------------------------------------------------------|

- *photon energy*  $E = h\nu$  in units of (eV).

– BlackBody\_SpectralEnergyDensity\_energy.dat

|                                             |                                              |
|---------------------------------------------|----------------------------------------------|
| AngularFrequency_omega[10 <sup>15</sup> /s] | SpectralEnergyDensity[kJ/m <sup>3</sup> /eV] |
|---------------------------------------------|----------------------------------------------|

– BlackBody\_SpectralRadiance\_energy.dat

|                                             |                                            |
|---------------------------------------------|--------------------------------------------|
| AngularFrequency_omega[10 <sup>15</sup> /s] | SpectralRadiance[kW/m <sup>2</sup> /eV/sr] |
|---------------------------------------------|--------------------------------------------|

– BlackBody\_SpectralIrradiance\_energy.dat

|                                             |                                           |
|---------------------------------------------|-------------------------------------------|
| AngularFrequency_omega[10 <sup>15</sup> /s] | SpectralIrradiance[kW/m <sup>2</sup> /eV] |
|---------------------------------------------|-------------------------------------------|

The file BlackBody\_Info.txt contains some additional information about the calculated black body spectrum.

### Intraband absorption in the single-band case

In the following we assume a single band with a parabolic energy band dispersion.

Tutorials showing results are available here:

- *Intersubband absorption of an infinite quantum well*
- *Intersubband transitions in InGaAs/AlInAs multiple quantum well systems*

For a 1D heterostructure grown along the  $x$  direction, formula for the *absorption coefficient*  $\alpha$  reads (see e.g. [ChuangOpto1995] or p. 53 in [FaistQCL2013])

$$\alpha(\omega) = \frac{e^2 \omega}{\epsilon_0 n_r c} \sum_i \sum_j (\bar{n}_i - \bar{n}_j) x_{ij}^2 \frac{\Gamma/2}{(E_j - E_i - \hbar\omega)^2 + (\Gamma/2)^2}$$

or, equivalently in energy,

$$\alpha(E) = \frac{e^2 E}{\hbar \varepsilon_0 n_r c} \sum_i \sum_j (\bar{n}_i - \bar{n}_j) x_{ij}^2 \frac{\Gamma/2}{(E_j - E_i - \hbar\omega)^2 + (\Gamma/2)^2}$$

where

- $\omega = E/\hbar$  is the frequency in units of (s<sup>-1</sup>)
- $E$  the energy in units of (J)
- $e$  is the elementary charge in units of (As)
- $\varepsilon_0$  is the vacuum permittivity in units of (As/Vm)
- $c$  is speed of light in vacuum in units of (m/s)
- $n_r = \sqrt{\varepsilon_r}$  is the refractive index assumed to be homogeneous. So we take the average of the quantum region (*check this*).
- $\bar{n}_i = \frac{1}{L} \sigma_i = \frac{1}{L} \int n_i(x) dx$  is the *averaged* electron density of subband  $i$  in units of (m<sup>-3</sup>), where  $L$  is the length of the quantum region and  $\sigma_i = \int n_i(x) dx$  is the subband density in units of (m<sup>-2</sup>)
- $x_{ij} = \langle i|x|j \rangle$  is the dipole moment between initial state  $i$  and final state  $j$  in units of (m)
- $\Gamma$  is the energy linewidth (broadening) in units of (J) in terms of full-width at half maximum (FWHM).

This equation includes a *Lorentzian broadening* which includes a factor of  $1/\pi$ .

We can also define the *position dependent absorption coefficient*

$$\alpha(\omega, x) = \frac{e^2 \omega}{\varepsilon_0 n_r c} \sum_i \sum_j (n_i(x) - n_j(x)) x_{ij}^2 \frac{\Gamma/2}{(E_j - E_i - \hbar\omega)^2 + (\Gamma/2)^2}$$

where

- $n_i(x)$  is the electron density of state  $i$  at position  $x$  in units of (m<sup>-3</sup>).

The units of both  $\alpha(\omega)$  and  $\alpha(\omega, x)$  are (m<sup>-1</sup>). In plots, typically (cm<sup>-1</sup>) is used.

If we integrate  $\alpha(\omega, x)$  over position  $x$  in the whole quantum region of length  $L$ , and divide by the length of the quantum region  $L$ , we obtain  $\alpha(\omega)$  as defined above,

$$\alpha(\omega) = \frac{1}{L} \int \alpha(\omega, x) dx.$$

So  $\alpha(\omega)$  as defined in the beginning of this section, where we averaged the density  $\bar{n}_i$ , is the averaged absorption coefficient in the quantum region and equivalent to the definition given here.

Finally, we note that this also works for the  $\mathbf{k} \cdot \mathbf{p}$  wave functions:

$$\alpha(\omega, x, k_{\parallel}) = \frac{e^2 \omega}{\varepsilon_0 n_r c} \sum_i \sum_j \sum_{k_{\parallel}} (n_i(x, k_{\parallel}) - n_j(x, k_{\parallel})) (x_{ij}(k_{\parallel}))^2 \frac{\Gamma/2}{(E_j(k_{\parallel}) - E_i(k_{\parallel}) - \hbar\omega)^2 + (\Gamma/2)^2}$$

where

- $n_i(x, k_{\parallel})$  is the electron density of state  $i$  at position  $x$  and vector  $k_{\parallel} = (k_x, k_y)$  (in 1D) or  $k_{\parallel} = k_z$  (in 2D).

### 5.3.22 \$buffer-solutions

Buffer solutions: To control pH values in electrolytes.

It is possible to overwrite the database entries of *\$buffer-solutions* by this keyword and its specifiers in the input file. You can also define a completely new buffer which is not contained in the database. If you overwrite a buffer that is contained in the database, it is necessary that you include *all* specifiers, including the ones that you do not want to overwrite. The program then uses all the values given in the input file and ignores all database entries.

| \$buffer-solutions |              | optional |
|--------------------|--------------|----------|
| buffer-name        | character    | required |
| number-of-ions     | integer      | required |
| ion-valency        | double_array | required |
| ion-name-1         | character    | required |
| ion-name-2         | character    | required |

(continues on next page)

(continued from previous page)

|                        |              |          |                                |
|------------------------|--------------|----------|--------------------------------|
| ion-name-3             | character    | optional | ! only necessary for some      |
| ↪ buffers, e.g. PBS    |              |          |                                |
| ion-name-4             | character    | optional | ! only necessary for some      |
| ↪ buffers, e.g. PBS    |              |          |                                |
| pKa                    | double_array | required | ! pKa at 25° C ( = 298.15 K)   |
| dpKa_dT                | double_array | required | ! d pKa / d T                  |
| z_acid                 | double_array | required | ! charge on the conjugate acid |
| ↪ species              |              |          |                                |
| \$end_buffer-solutions |              | optional |                                |

$pK_a$  and  $d pK_a / d T$ .

### Example 1: MOPS

```

!-----!
$buffer-solutions
buffer-name      = MOPS                ! MOPS (C7H15NO4S) + NaOH
number-of-ions   = 2                   !
ion-valency      = -1.0                +1.0 !
                  ! (C7H14NO4S)^-   Na^+ !
ion-name-1       = Mops^-              ! C7H15NO4S <==> (C7H14NO4S)^- + H^+
ion-name-2       = Na^+                !
pKa              = 7.31                ! pKa at 25° C ( = 298.15 K)
                  ! Note: This pKa is thermodynamic value.
                  ! The working pKa' is 7.20.
dpKa_dT          = -0.011              ! d pKa / d T
z_acid           = 0.0                 ! charge on the conjugate acid species (0 =
↪ C7H15NO4S)
$end_buffer-solutions
!-----!

```

Involved ions and molecules:

- buffer-name: MOPS ( $C_7H_{15}NO_4S$ ) + NaOH
- ion-valency: ( $C_7H_{14}NO_4S$ )<sup>-</sup> and Na<sup>+</sup>
- ion-name-1:  $C_7H_{15}NO_4S \rightleftharpoons (C_7H_{14}NO_4S)^- + H^+$
- ion-name-2: Na<sup>+</sup>
- z\_acid: 0 =  $C_7H_{15}NO_4S$

### Example 2: PBS (phosphate buffer)

The phosphate buffer is special (and thus more complicated) because it consists of three  $pK_a$  values (and it thus has four different ions).

```

!-----!
$buffer-solutions
buffer-name      = PBS                ! PBS (phosphate buffer)
number-of-ions   = 4                   !
ion-valency      = -1.0                -2.0    -3.0    1.0 !
                  ! (H2PO4)^-   (HPO4)^2-   (PO4)^3-   Na+ !
ion-name-1       = H2PO4^-            ! NaH2PO4 <==>
↪ (H2PO4)^- + Na^+
ion-name-2       = HPO4^2-            ! Na2HPO4 <==> (HPO4)^
↪ 2- + 2 Na^+
ion-name-3       = PO4^3-              !
ion-name-4       = Na^+                !
pKa              = 2.15    7.21    12.33 ! pKa,1    pKa,2    pKa,

```

(continues on next page)

(continued from previous page)

```

↪3   at 25° C ( = 298.15 K)
dpKa_dT      = 0.0044  -0.0028  -0.026           ! d pKa   / d T
z_acid       = 0.0      -1.0      -2.0           ! charge on the
↪conjugate acid species
           !   0 = H3PO4  -1 = (H2PO4)^-  -2 = (HPO4)^2-  !
$end_buffer-solutions
!-----!

```

Involved ions and molecules:

- ion-valency:  $(\text{H}_2\text{PO}_4)^-$ ,  $(\text{HPO}_4)^{2-}$ ,  $(\text{PO}_4)^{3-}$  and  $\text{Na}^+$
- ion-name-1:  $\text{NaH}_2\text{PO}_4 \rightleftharpoons (\text{H}_2\text{PO}_4)^- + \text{Na}^+$
- ion-name-2:  $\text{Na}_2\text{HPO}_4 \rightleftharpoons (\text{HPO}_4)^{2-} + 2 \text{Na}^+$
- ion-name-3:  $(\text{PO}_4)^{3-}$
- ion-name-4:  $\text{Na}^+$
- z\_acid: 0 =  $\text{H}_3\text{PO}_4$ , -1 =  $(\text{H}_2\text{PO}_4)^-$ , -2 =  $(\text{HPO}_4)^{2-}$

For the explanation of the specifiers, please also check the description in the database of the keyword `$buffer-solutions`.

### 5.3.23 \$buffer-constant-A(T)

Constant  $A(T)$  used for buffer calculations: The  $pK_a$  value depends on the ionic strength.

|                                   |                     |                 |
|-----------------------------------|---------------------|-----------------|
| <b>\$buffer-constant-A(T)</b>     |                     | <b>optional</b> |
| constant-Centigrade-to-Kelvin     | <b>double</b>       | <b>required</b> |
| T_A(T)                            | <b>double_array</b> | <b>required</b> |
| <b>\$end_buffer-constant-A(T)</b> |                     | <b>optional</b> |

Example 1

```

!-----!
$buffer-constant-A(T)                                !
constant-Centigrade-to-Kelvin = 273.15                ! Kelvin = Celsius +
↪273.15                                                !
!-----!
!=====!
!               first column: T[C]   second column: A(T) !
!=====!
T_A(T)          =   0.0      0.4918   !   0° C = 273.15 K
                  10.0      0.4989   !  10° C = 283.15 K
                  20.0      0.5070   !  20° C = 293.15 K
                  25.0      0.5114   !  25° C = 298.15 K
                  30.0      0.5161   !  30° C = 303.15 K
$end_buffer-constant-A(T)                            !
!-----!

```

Example 2

```

!-----!
$buffer-constant-A(T)                                !
constant-Centigrade-to-Kelvin = 273.15                ! Kelvin = Celsius +
↪273.15                                                !
!=====!

```

(continues on next page)



(continued from previous page)

```

!               first column: T[C]   second column: A(T)   !
!=====!
T_A(T)                = 25.0        0.5114                ! 25° C = 298.15 K
$end_buffer-constant-A(T)                                !
!-----!

```

If the keyword `$buffer-constant-A(T)` is present in the input file, the values for the keyword *\$buffer-constant-A(T)* in the database are overwritten.

It is possible to specify only one pair of “T, A(T)” values as shown in Example 2.

For more details on the physical significance of this keyword, please have a look at the database section: *\$buffer-constant-A(T)*

### 5.3.24 \$tighten

#### Tight-binding

The original *tighten* code that is used inside *nextnano<sup>3</sup>* has been written by Peter Vogl, Walter Schottky Institute, Technische Universität München. It is based on the *sp<sup>3</sup>d<sup>5</sup>s\** method. For details about this method, see e.g. Ref. [JancuPRB1998].

Here is a general template for the tighten code block for the input file, in which the variables should be specified as given in the explanations and examples further below.

```

!-----!
$tighten
calculate-tight-binding-tighten      character      optional !
tighten-method                       character      optional !
tight-binding-model                   character      optional !
destination-directory                 character      required !
input-directory                      character      optional !
filename-material-parameters          character      optional !
filename-distance-parameters          character      optional !
filename-tighten                     character      optional !
filename-k-vectors                   character      optional !
k-vectors-sample-type                 character      optional !
Brillouin-zone-path                  character      optional !
Brillouin-zone-sections              character      optional !
number-of-k-points                    integer        optional !
k-direction-from-k-point              double_array   optional !
k-direction-to-k-point               double_array   required  !
power-of-distance-dependence          double          optional !
calculate-eigenvectors                character      optional !
scale                                double          optional !
potential-energy-left                  double          optional !
potential-energy-right                 double          optional !
debug-level                           integer         optional !
!
filename-states                       character      optional !
calculate-only-lattice-geometry        character      optional !
output-Hamiltonian                    character      optional !
calculate-k-derivatives                character      optional !
tighten-option                        character      optional !
number-of-band-for-Fermi-energy         integer        optional !
number-of-electron-eigenvalues         integer        optional !
number-of-hole-eigenvalues             integer        optional !
number-of-band-for-psi                 integer        optional !

```

(continues on next page)

(continued from previous page)

|                                |              |          |                     |
|--------------------------------|--------------|----------|---------------------|
| number-of-bands-to-plot        | integer      | optional | !                   |
| swap-cation-and-anion          | character    | optional | !                   |
| rescale-to-unstrained-k-points | character    | optional | ! for graphene only |
| !-----                         |              |          |                     |
| ! Pseudopotential algorithm    |              |          |                     |
| !-----                         |              |          |                     |
| pseudopotential-scaling-factor | double       | optional | !                   |
| pseudopotential-form-factors   | double_array | optional | !                   |
| G-vectors-included             | integer      | optional | !                   |
| <b>\$end_tighten</b>           |              | optional | !                   |
| !-----                         |              |          |                     |

**\$atomic-layers** is only necessary for the heterostructure tight-binding code but not for the bulk tight-binding code.

|                            |           |          |   |
|----------------------------|-----------|----------|---|
| !-----                     |           |          |   |
| <b>\$atomic-layers</b>     |           | optional | ! |
| layer-number               | integer   | required | ! |
| !                          |           |          |   |
| ion-1                      | character | required | ! |
| ion-1-content              | double    | required | ! |
| !                          |           |          |   |
| ion-2                      | character | optional | ! |
| ion-2-content              | double    | optional | ! |
| <b>\$end_atomic-layers</b> |           | optional | ! |
| !-----                     |           |          |   |

### Example: \$atomic-layers

```

$atomic-layers

!-----
! layer 1: cation
! layer 2: anion
!-----
layer-number = 1  ion-1-name = Ga  ion-1-content = 0.5  ion-2-name = Al  ion-2-
→ content = 0.5  ! Ga/Al cation / alloy content
layer-number = 2  ion-1-name = As  ion-1-content = 1.0
→                ! As  anion / alloy content
layer-number = 3  ion-1-name = Ga  ion-1-content = 1.0
→                ! Ga  cation / alloy content
layer-number = 4  ion-1-name = As  ion-1-content = 1.0
→                ! As  anion / alloy content

```

#### Note

- The total number of layers in [001] superlattice direction must be integer multiple of 4 (e.g. 4, 8, 12, ...) because a unit cell consists of 4 **atomic layers**.
- Convert number of **layers** (atomic layers) into [nm] units: position = layer-number \* LatticeConstant / 4
- Convert number of **molecular layers** into [nm] units: position = layer-number \* LatticeConstant / 2

- The lattice constant refers to the constant distance between unit cells in a crystal lattice. A unit cell consists of 2 molecular layers in diamond/zinc blende structure.
- It must hold: `ion-1-content + ion-2-content = 1.0`.

### Example: \$tighten

```
!-----!
$tighten
calculate-tight-binding-tighten = yes
tighten-method = rashba2tighten-tighten
tight-binding-model = Scholz
destination-directory = TightBinding/
↳for tight-binding output. This is where all output goes.
input-directory = ../Syntax/Tight-binding files/
↳for tight-binding input, relative to executable path
!input-directory = H:\Tighten\TIGHTEN_nextnano3\
↳for tight-binding input, absolute path
filename-material-parameters = TB_material_parameters.in
↳is then: input-directory/filename-material-parameters
filename-distance-parameters = TB_distance_parameters.in
↳is then: input-directory/filename-distance-parameters
filename-tighten = tighten.in
↳written to destination-directory/ and will be read in again by the tighten
↳algorithm.
                                     ! If tighten-method
↳= tighten, the full filename is then: input-directory/filename-tighten
filename-k-vectors = k_vectors.dat
↳written to destination-directory/ and will be read in again by the tighten
↳algorithm.
                                     ! If tighten-method
↳= tighten, the full filename is then: input-directory/filename-k-vectors

k-vectors-sample-type = band
Brillouin-zone-path = 0.5 0.5 0.5
                    0.0 0.0 0.0
                    1.0 0.0 0.0
Brillouin-zone-sections = 0.5 0.5 0.5
                        0.0 0.0 0.0
                        0.0 0.0 0.0
                        1.0 0.0 0.0
number-of-k-points = 1000

k-direction-from-k-point = -0.5 0.0 0.5
↳of [2pi/a]
k-direction-to-k-point = 0.5 0.0 0.5
↳of [2pi/a]
power-of-distance-dependence = 2.0
calculate-eigenvectors = no
scale = 1.0
potential-energy-left = 0.0
potential-energy-right = 0.0
debug-level = 1

!-----
! Now tighten....
```

(continues on next page)

(continued from previous page)

```

!-----
filename-states = states.in ! The full filename.
↪is then: input-directory/filename-states

calculate-only-lattice-geometry = no ! 'yes' / 'no'
output-Hamiltonian = no ! 'yes' / 'lower' /
↪'no'
calculate-k-derivatives = no ! 'yes' / 'first' /
↪'second' / 'no'
tighten-option = eigenvalues-only ! 'eigenvalues-only'
↪', 'eigenvectors-and-density', 'eigenvectors-and-g-factor', 'no-eigenvalues'
number-of-band-for-psi = 1 ! used with 2 above,
↪ index of 1st of 4 bands w |psi|^2
number-of-bands-to-plot = 4 !

swap-cation-and-anion = no ! 'no' / 'yes'
↪(default is no. For testing only, cation and anion material parameters of database
↪are exchanged.)

!-----
↪-----
! Pseudopotential algorithm ==> Reference: T. P. Pearsall, Quantum Photonics (2017),
↪Chapter 4
!-----
↪-----
pseudopotential-scaling-factor = 1.0 ! (default is 1.0.
↪scaling factor for potentials (0.0 <= ... <= 1.0)

! 0.0 = no
↪potential, i.e. free electron, 1.0 = full periodic potential, anything in between
↪is a mixture

! ==> Note: This
↪feature can be used to obtain the free-electron dispersion in a zincblende lattice.
pseudopotential-form-factors = 0.0 ... 0.136 ! 10 values: U0,s,
↪ U3,s U4,s U8,s U11,s U0,a U3,a U4,a U8,a U11,a
! U0,s and U0,a can
↪be chosen both as 0.0. They just determine the reference energy.
! The subscripts s
↪and a refer to the symmetric and antisymmetric form factors.
! The lattice
↪constant is also needed!!!
G-vectors-included = 11 ! include
↪reciprocal G vectors up to this subscript, e.g. G_0 (0), G_3 (3), G_4 (4), G_8 (8),
↪G_11 (11), G_12 (12); The subscript is the length squared |G|^2.
! G_0 = (0 0 0)
↪( 1 vector, total number of G vectors: 1)
! G_3 = (1 1 1)
↪( 8 vectors, total number of G vectors: 9)
! G_4 = (2 0 0)
↪( 6 vectors, total number of G vectors: 15)
! G_8 = (2 2 0)
↪(12 vectors, total number of G vectors: 27)
! G_11 = (3 1 1)
↪(24 vectors, total number of G vectors: 51)
! G_12 = (2 2 2)
↪( 8 vectors, total number of G vectors: 59)

```

(continues on next page)

(continued from previous page)

```
$end_tighten
!-----!
```

## Parametrization

Scholz and Sarma parametrizations are given in the articles [*JancuPRB1998*] and [*Sarma2002*]. The different models can be chosen using `tight-binding-model`.

### Note

The standard parameter files are specified as follows

```
filename-material-parameters = TB_material_parameters.in ! The full filename is_
→then: input-directory/filename-material-parameters
filename-distance-parameters = TB_distance_parameters.in ! The full filename is_
→then: input-directory/filename-distance-parameters
```

and are located under

```
\nextnano3\Syntax\Tight-binding files\
\nextnano3\Syntax\Tight-binding files\TB_material_parameters.in
\nextnano3\Syntax\Tight-binding files\TB_material_parameters_JancuPRB1998.in
\nextnano3\Syntax\Tight-binding files\TB_material_parameters_SawamuraOME2018.in
```

## Syntax

It is important to select the appropriate flow-scheme needed for tight-binding in *\$simulation-flow-control*, e.g. `flow-scheme = 200`.

### calculate-tight-binding-tighten

do tight-binding calculation with *tighten*

#### options

yes or no

#### default

no (no tight-binding calculation)

### tighten-method

[*rashba2tighten* ==> *tighten*]

#### options

*rashba2tighten-tighten* generate input file for *tighten* and do tight-binding calculation with *tighten*

*rashba2tighten* only generate input file for *tighten*

*tighten* tight-binding calculation (*tighten* only)

### tight-binding-model

#### options

Scholz (default) (only for III-V materials)

Sarma (for II-VI materials)

### destination-directory

directory name for tight-binding output. This is where all output goes.

#### default

./

**example**

TightBinding/

**input-directory**

directory name for tight-binding input, relative to executable path.

**default**

./

**example**

../Syntax/Tight-binding files/

**example**

"H:\My tight-binding folder\"

**filename-tighten**

- Name of tight-binding input file. It will be generated by *rashba2tighten*, and read in again by *tighten*.
- This file is written to **destination-directory** and will be read in again by the *tighten* algorithm.
- If **tighten-method** = **tighten**, the full filename is then: **input-directory \ filename-tighten**

**default**

tighten.in

**filename-k-vectors**

- This file contains user input for lattice structure and tight-binding parameters.
- It will be generated by *rashba2tighten*, and is read in again by *tighten* program.
- If **tighten-method** = **tighten**, the full filename is then: **input-directory \ filename-k-vectors**.
- This file contains the k vectors for the tight-binding Hamiltonian that will be diagonalized, i.e. for which the eigenenergies and eigenfunctions will be calculated.
- The columns in this file are:

REAL(loop\_index\_over\_k) 0.5\*(kx+ky) 0.5\*(ky+kz) 0.5\*(kx+kz) kx ky kz

**default**

k\_vectors.dat

**k-vectors-sample-type**

Here one specifies how the k vectors have to be sampled for the **bulk** or **superlattice** tight-binding code.

**options**

**band** for band structure plot along some predefined lines (the lines are different for the **bulk** and **superlattice** code, see below)

**user-defined-path** as defined in **Brillouin-zone-path** = ...

**user-defined-sections** as defined in **Brillouin-zone-sections** = ...

For the **superlattice** tight-binding code the relevant options are:

**band**

- for band structure plot along the lines (**superlattice** code):  $\Gamma \Rightarrow$  (along  $\Delta$  to)  $X = 0.5$   $G1 \Rightarrow$  (along  $Z$  to)  $M = 0.5$  ( $G1+G2$ )  $\Rightarrow$  (along  $\Sigma$  to)  $\Gamma \Rightarrow Z = 0.5$   $G3$
- reciprocal primitive vectors:  
 $G1 = (2\pi \sqrt{2}/a_{\text{lateral}}) * (1 \ 0 \ 0) \Rightarrow$  along  $(1 \ 0 \ 0)$  in-plane direction

$G2 = (2\pi \sqrt{2}/a_{\text{lateral}}) * (0 \ 1 \ 0) \Rightarrow$  along  $(0 \ 1 \ 0)$  in-plane direction  
 $G3 = (2\pi /a_{\text{vertical}}) * (0 \ 0 \ 1) \Rightarrow$  along superlattice direction  
 $a_{\text{lateral}}$ : in-plane lattice constant  
 $a_{\text{vertical}}$ : out-of-plane lattice constant  
 Some predefined paths are:  
 $k_{\text{parallel}}100$ -Gamma- $k_{\text{superlattice}}X = -0.5 G1 \Rightarrow$  (along  $\Delta$  to)  $\Gamma \Rightarrow$   
 $Z = 0.5 G3$   
 $k_{\text{parallel}}110$ -Gamma- $k_{\text{superlattice}}M = -0.5 (G1+G2) \Rightarrow$  (along  $\Sigma$  to)  
 $\Gamma \Rightarrow Z = 0.5 G3$   
 $110G3$  values, namely  $(000)$ ,  $(kx0,0,0)$ ,  $(0,kx0,0)$ ,  $[110]$  direction and  $[1-10]$  direction  
 $100D$  “100D” - “001D”  
 two 2 values, namely  $(000)$  and the one specified  $(kx0,ky0,kz0)$   
 circle  $\Gamma = (000)$  with radius  $k_{\text{mod}}=|k0|$ , surrounding  $\Gamma$  point at distance specified.  
 $xyz$  number\_of\_k\_points values from  $(0, 0, 0) \Rightarrow (kx0,ky0,kz0)$   
 $L$ -projected-Gamma- $X$ -projected  
 $L$ -Gamma- $X$ -projected  
 $L$ -Gamma- $X$   
 $k1$ -Gamma- $k2$   
 $k1$ - $k2$   
 $R$ - $Z$ -Gamma- $X$ - $M$ - $A$  See p. 404, Fig. 8.37 of C. Hamaguchi, Basic Semiconductor Physics, 2nd edition and Fig. 8.29  
 Gamma- $X$ - $R$ - $Z$ -Gamma- $M$ - $A$ -Gamma See Fig. 8 of [SawamuraOME2018]

### Bulk tight-binding code

For the bulk tight-binding code the relevant options are:

band for band structure plot along the lines (bulk code); This is equivalent to  $L$ -Gamma- $X$ - $W$ - $L$ - $K$ -Gamma for the bulk tight-binding code.

Some predefined paths are:

$L$ -Gamma- $X$ - $W$ - $K$ - $L$ - $W$ - $X$ - $K$ -Gamma

- same as [JancuPRB1998] and Fig. 1 of [SawamuraOME2018]
- The points U and K have the same energies but the path to these points from  $\Gamma$  or X is not equivalent.
- $L$ - $\Gamma$  is along  $\Lambda$  axis, i.e. along  $[1,1,1]$  direction ( $L$  is at  $2\pi/a (0.5,0.5,0.5)$ ;  $\Gamma$  is at  $2\pi/a (0,0,0)$ ). The distance between these points is  $\sqrt{3}\pi/a$ .
- $\Gamma$ - $X$  is along  $\Delta$  axis, i.e. along  $[1,0,0]$  direction ( $X$  is at  $2\pi/a (1,0,0)$ )
- $X$ - $W$  is along V
- $W$ - $K$  is along
- $K$ - $L$  is along
- $L$ - $W$  is along Q
- $W$ - $X$  is along V
- $X$ - $K$  is along

- K- $\Gamma$  is along  $\Sigma$

L-Gamma-X-W-L-K-Gamma e.g. S. Sapra et al., PRB 66, 205202 (2002) (same as band)

L-Gamma-X-K-Gamma from L to  $\Gamma$  to X to K to from L to  $\Gamma$

L-Gamma-X-U-K-Gamma e.g. Tom P. Pearsall, Quantum Photonics (Band structure plots for pseudopotential calculations) (*to do: eliminate the path between U and K*)

L-Gamma-X from L to  $\Gamma$  to X

X-Gamma-Z-U'-L-Gamma-K (for strain)

110G 3 values, namely (000), (kx0,kx0,0), (-kx0, kx0,0)

110X 3 values, namely (001), (kx0,kx0,1), (kx0,-kx0,1)

110D 3 values, namely (00kz0), (kx0,kx0,kz0), (-kx0, kx0,kz0)

110L 3 values, namely (0.5,0.5,0.5), (0.5+kx0,0.5+kx0,0.5-2kx0), (0.5+kx0,0.5-kx0,0.5)

100G 3 values, namely (000), (kx0,0,0), (0,ky0,0)

two 2 values, namely (000) and (kx0,ky0,kz0)

cirG circle Gamma=(000) in kz=0 plane with radius kmod=|k0|

cirX circle X=(001) in kz=1 plane with radius kmod=|k0|

cirL circle L=(111) in plane  $\lambda*(1,1,-2) + \mu*(1,-1,0)$  with radius kmod=|k0|

xyz number\_of\_k\_points values from (0,0,0) ==> (kx0,ky0,kz0)

xff number\_of\_k\_points values from (0,ky0,kz0) ==> (kx0,ky0,kz0)

ffz number\_of\_k\_points values from (kx0,ky0,0) ==> (kx0,ky0,kz0)

### Graphene

For the bulk tight-binding code of graphene the relevant options are:

M'\_K\_Gamma\_M\_K'\_M' band structure of graphene along the path M' ==> K ==> Gamma ==> M ==> K' ==> M'

K\_Gamma\_M\_K' band structure of graphene along the path K ==> Gamma ==> M ==> K'

### Brillouin-zone-path

This is a path through the Brillouin zone passing through all these k points in this order. The number of entries must be a multiple of 3.

#### example

```
Brillouin-zone-path = 0.5 0.5 0.5 ! L
                     0.0 0.0 0.0 ! Gamma
                     1.0 0.0 0.0 ! X
                     ... ... ! ...
                     ... ... ! ...
```

### Brillouin-zone-sections

Here one can specify segments along where to sample the Brillouin zone. The number of entries must be a multiple of 6.

#### example



```

Brillouin-zone-sections = 0.5  0.5  0.5  ! L      ! Section 1: This is a
↪path through the Brillouin zone between these two k points. L ==>
↪Gamma
                                0.0  0.0  0.0  ! Gamma

                                0.0  0.0  0.0  ! Gamma ! Section 2: Then a
↪path through the Brillouin zone between these two k points is taken.
↪Gamma ==> X
                                1.0  0.0  0.0  ! X

                                0.0  0.0  0.0  ! Gamma ! Section 3: Then a
↪path through the Brillouin zone between these two k points is taken.
↪Gamma ==> L
                                0.5  0.5  0.5  ! L

                                ...   ...   ...  !
                                ...   ...   ...  !

```

### Brillouin zone of zincblende semiconductor

In order to understand the location of the k points in the Brillouin zone, the following website is very helpful:  
<http://lampx.tugraz.at/~hadley/ss1/bzones/fcc.php>

- Define Brillouin zone boundaries in units of  $2\pi/a$  where  $a$  is the lattice constant.
- The first Brillouin zone comprises those points in reciprocal space that are closer to the origin (i.e. to the Gamma point) than to any other point of the reciprocal lattice.
- The high symmetry points are called:
  - L [  $1/2$  ,  $-1/2$  ,  $1/2$  ]  $\sim$ [1-11]
  - $\Gamma$  [ 0 , 0 , 0 ] (origin, i.e. center of the first Brillouin zone)
  - X [ 1 , 0 , 0 ]  $\sim$ [100]
  - U [ 1 ,  $1/4$  ,  $-1/4$  ]
  - K [ 0 ,  $3/4$  ,  $3/4$  ]
  - $\Gamma$  [ 0 , 0 , 0 ]
  - W [ 1 ,  $\sqrt{2}/2$  , 0 ]
- The line from  $\Gamma$  to [X] along the high-symmetry direction [100] is called Delta  $\Delta$ .
- The line from  $\Gamma$  to [L] along the high-symmetry direction [111] is called Lambda  $\Lambda$ .
- The line from [K] to [X] along the high-symmetry direction [...] is called Sigma  $\Sigma$ . (*should be checked*)
- The distance in k space from  $\Gamma$  to [L] is  $\sqrt{3}\pi/a$ .
- If one goes from  $\Gamma$  to [X] along the line  $\Delta$ , the cubic symmetry splits the 8 bands into 2 bands.
- At the zone center  $\Gamma$ , the energy value is 8-fold degenerate.
- At the [X] point, there are two bands, each of which is 4-fold degenerate.
- If one goes from  $\Gamma$  to [L] along the line  $\Lambda$ , the cubic symmetry splits the 8 bands into 4 bands, the upper and lower one is 1-fold degenerate, the two intermediate ones are each 3-fold degenerate.
- The electronic wave function at the  $\Gamma$  point in the center of the Brillouin zone sees the maximum symmetry of the fcc lattice.
- There are 48 operations (rotations, reflections, inversion) that leave the structure invariant.

**number-of-k-points**

number of k vectors for which to calculate eigenvalues (only relevant for band, k\_parallel\_100-Gamma-k\_superlattice, k\_parallel\_100-Gamma-k\_superlattice, xyz so far)

**example**

1000

With the following flags one can specify a k vector in the Brillouin zone.

```
k-direction-to-k-point = 0.5 0.0 0.5 ! for superlattice tight-
→binding code: k = (kx,ky,kSL) vector in dimensionless units [2sqrt(2)pi/
→alateral], [2pi/avertical]
k-direction-to-k-point = 0.01 0.01 0.0 ! for bulk tight-
→binding code: k = (kx,ky,kz) vector in units of 2pi/(kx0/alateral,ky0/
→alateral,kz0/avertical)
```

The superlattice k.p dispersion can be calculated along an arbitrary line from the k point k-direction-from-k-point to the Gamma point and then to the k point k-direction-to-k-point. Either k-direction-from-k-point or k-direction-to-k-point or both can be zero. If both are zero, then only the Gamma point is calculated. k-direction-from-k-point can be omitted.

You can use this flag to specify a customized plot for the E(k) dispersion, e.g. along a line from [110] to the Gamma point and then to the [001] point.

```
k-direction-from-k-point = -0.5 -0.5 0.0 ! k vector in units of [2pi/a]
k-direction-to-k-point = 0.0 0.0 0.0 ! k vector in units of [2pi/a]
```

**power-of-distance-dependence****default**

2.0

$$t_i = t_0(d_0/d_i)^\eta$$

It seems that this value is not used inside the code, unscaled matrix elements are used.

**calculate-eigenvectors****value**

yes or no

**default**

no calculate eigenvalues only

**scale****default**

1.0

**example**

5.0

scale output of wave functions and  $|\psi|^2$  to improve visualization of  $|\psi|^2$  in the band edges plot

**potential-energy-left****default**

0.0 [eV]

Add potential energy to band edges (value at first layer).

**potential-energy-right****default**

0.0 [eV]

Add potential energy to band edges (value at last layer). The values in between are interpolated for each layer. This way an electrostatic potential or electric field can be included.

#### debug-level

##### default

1

get reduced standard output, this is appropriate to generate an input file if unit cell has many atoms. Note: Only first element of star is printed.

##### options

- 2 get very detailed output, excluding Hamiltonian
- 3 get very detailed output, including Hamiltonian to stdio

#### filename-states

##### default

states.in

The full filename is then: `input-directory \ filename-states`

This file is located in the folder: `\nextnano3\Syntax\Tight-binding files\`

For more information on this input file, see documentation further below.

#### calculate-only-lattice-geometry

##### default

no

##### option

yes Calculate only lattice geometry but not tight-binding Hamiltonian.

#### output-Hamiltonian

##### default

no Hamiltonian matrix not written to file

##### options

- yes whole Hamilton matrix is written column wise
- lower lower triangle is written row wise

#### calculate-k-derivatives

##### default

no do not calculate k-derivatives of  $H(k)$

##### options

- yes calculate first and second
- first calculate  $d/dk H(k)$
- second calculate  $d^2/dk_{ik_j} H(k)$

#### Note

ham\_1deriv\_output\_file

ham\_2deriv\_output\_file

... files that contain nonzero matrix elements of first (and second) k-derivative of  $H$  in same format as ham\_output\_file,

but only SPIN\_UP/SPIN\_UP part since SPIN-DN/SPIN-DN is identical and SPIN-DN/SPIN-UP=0.

```
der1.dat: ham_1deriv_output_file = 'der1.dat'
der2.dat: ham_2deriv_output_file = 'der2.dat'
```

**tighten-option****default**

eigenvalues-only calculate only eigenvalues (eigval)

**options**

- **eigenvectors-and-density** calculate eigenvectors and density of tight-binding Hamiltonian (ev+dens)
- **eigenvectors-and-g-factor** calculate eigenvectors and g-factor (ev+g)
- **no-eigenvalues** no diagonalization

**number-of-band-for-Fermi-energy****range**

$\mathbb{N}_{\neq}$

**default**

0 i.e. highest valence band number

**example**

8

zero of energy is taken at this band for Gamma point (or first k vector)

This specifier only applies to the bulk tight-binding algorithm and the pseudopotential code.

**number-of-electron-eigenvalues****range**

$\mathbb{N}$

**default**

11

**number-of-electron-eigenvalues****range**

$\mathbb{N}$

**default**

15

This specifier only applies to the superlattice tight-binding algorithm.

Include 11 electron and 15 hole energies in E(k) energy dispersion plot.

**number-of-band-for-psi****default**

1

used with 2 above, index of first of 4 bands with  $|\psi|^2$  (actually 8 bands due to spin degeneracy)

**Note**

- If `ieigenvalues_flag = 2`, wave functions are calculated at Gamma for every second of `number-of-bands-to-plot/2` consecutive bands, starting with `number-of-band-for-psi`, where -1 and 0 = top of valence band, 1 and 2 = first conduction band. Choose an odd number.

- If `ieigenvalues_flag = 3`, g-factor tensor for this plus following (nondegenerate) band is calculated

**number-of-bands-to-plot****default**

4

This is the number of bands that are plotted in the files

- `Gamma_psi_squared.dat`  $|\psi|^2$ , ... are not shifted
- `Gamma_psi_squared_shift.dat`  $|\psi|^2$ , ... are shifted by their eigenenergies with respect to the energy dispersion plot
- `Gamma_psi_squared_shift_bandedges.dat`  $|\psi|^2$ , ... are shifted by their eigenenergies with respect to the band edges plot

starting from the band indicated with `number-of-band-for-psi`.

This number does not take into account spin. E.g. if you specify 4, the program will print out 8 bands, i.e. twice as much, i.e. spin is taken into account automatically by the program.

**rescale-to-unstrained-k-points****default**

no Do not rescale band structure to unstrained k points. It can be useful to keep the high symmetry points fixed on the graph in order to compare different strains.

**option**

yes Rescale band structure to unstrained k points. This moves the high symmetry points if strained.

This flag is only relevant for the tight-binding code of graphene.

**Required input files****Material parameters**

`TB_material_parameters.in`: located in folder `nextnano3\Syntax\Tight-binding files\TB_material_parameters.in`

Material parameters that also occur in the normal nextnano database:

`c11 c12` elastic constants  $10^{-2}$  [GPa]

`a` lattice constant [Angstrom]

... tight binding parameters [eV]

**Tight-binding material parameters**

`nc na` number of electrons on cation and number of electrons on anion (e.g. 3 for Ga and 5 for As in GaAs; 4 for Si)

`Esc Epc Esec Edct2 Edce` orbital energies [eV], e.g. `Esc` is s-orbital energy of cation

`Esa Epa Esea Edat2 Edae` orbital energies [eV], e.g. `Esa` is s-orbital energy of anion

(`Edct2` and `Edat2` are used instead of `Ed` to allow for d-orbital splitting)

`Dav` absolute deformation potential [eV]; difference in absolute valence band deformation potentials (in eV) between intrinsic Scholz values and Van de Walle/Needs values. Inclusion of this term guarantees that calculated `av` agrees with Van de Walle/Needs.

`sss scpas pcsas pps ppp seses secsas scseas secpas pcseas scdas dcsas pcdas dcpas pcdap dcpap secdas dcseas dds ddp ddd`

$ss\sigma, s_cp_a\sigma, s_ap_c\sigma, pp\sigma, pp\pi, s^*s^*\sigma, s_c^*s_a\sigma, s_a^*s_c\sigma, s_c^*p_a\sigma, s_a^*p_c\sigma, s_cd_a\sigma, s_ad_c\sigma, p_cd_a\sigma, p_ad_c\sigma, p_cd_a\pi, p_ad_c\pi, s_c^*d_a\sigma, s_a^*d_c\sigma, dd\sigma, dd\pi, dd\delta$

so\_p\_c so\_p\_a onsite and intersite spin-orbit couplings

so\_d\_c so\_d\_a

so\_ppca

Gammav energy of unstrained top of valence band edge at Gamma point;

Gammac energy of unstrained conduction band edge minimum at Gamma point [eV]

Both, Gammav and Gammac, are with respect to an absolute energy scale (crude estimate only).

The last two values are not used for the tight-binding calculation. However, these are the relevant values that are contained in the output files `band_edges_nm.dat` and `band_edges.dat`. They are the values of the unstrained band edges on an absolute energy scale. They are crude estimates only. As already said, they are not input to the actual tight-binding calculation but they are similar to the results of a bulk tight-binding calculation.

The empirical tight-binding material parameters for Ge are the ones of [JancuPRB1998], Table II with the following exceptions:  $E_s, E_p, E_d, E_{s^*}$  are shifted by +1.7683 eV.

## Distance parameters

TB\_distance\_parameters.in: located in folder nextnano3\Syntax\Tight-binding files\  
TB\_distance\_parameters.in

These parameters are relevant for strained materials or for alloys.

nsss nscpas npcsas npps nppp nseses nsecsas nscseas nsecpas npcseas nscdas ndcsas npcdas ndcpas  
npcdap ndcpap nsecdas ndcseas ndds nddp nddd

bdeff strain dependent shift of onsite-d energies in [eV] (deformation potential). It is chosen to be identical for cation and anion.

bdeff = bd \* ed\_Scholz

## States

states.in: located in folder nextnano3\Syntax\Tight-binding files\states.in

This noneditable *namelist* file (*namelist* is a Fortran feature) will be read in by the *tighten* program and contains the following information:

```
&state_description

number_state_ref    = 10

state_ref_name(1)   = 's'
state_ref_name(2)   = 'px'
state_ref_name(3)   = 'py'
state_ref_name(4)   = 'pz'
state_ref_name(5)   = 'se'
state_ref_name(6)   = 'dxy'
state_ref_name(7)   = 'dyz'
state_ref_name(8)   = 'dzx'
state_ref_name(9)   = 'dx2y2'
state_ref_name(10)  = 'dz2r2'

number_coup_ref     = 21

coup_ref_name(1)    = 'sss'
```

(continues on next page)

(continued from previous page)

```

coup_ref_name(2)  = 'sps'
coup_ref_name(3)  = 'pss'
coup_ref_name(4)  = 'pps'
coup_ref_name(5)  = 'ppp'
coup_ref_name(6)  = 'seses'
coup_ref_name(7)  = 'sess'
coup_ref_name(8)  = 'sses'
coup_ref_name(9)  = 'seps'
coup_ref_name(10) = 'pses'
coup_ref_name(11) = 'sds'
coup_ref_name(12) = 'dss'
coup_ref_name(13) = 'pds'
coup_ref_name(14) = 'dps'
coup_ref_name(15) = 'pdp'
coup_ref_name(16) = 'dpp'
coup_ref_name(17) = 'seds'
coup_ref_name(18) = 'dses'
coup_ref_name(19) = 'dds'
coup_ref_name(20) = 'ddp'
coup_ref_name(21) = 'ddd'

```

/

### Generated output files

- k\_vectors.dat
- tighten.in

These are the input files for *tighten*.

- out\_structure.txt
- band\_edges\_unstrained\_nm.dat Gamma conduction band edge and topmost valence band edge (units: position [nm], energy [eV], band gap [eV])
- band\_edges\_unstrained\_layers.dat Gamma conduction band edge and topmost valence band edge (units: atomic layer, energy [eV])
- E(k)\_tighten\_new\_bandedges.dat energy dispersion E(k) where the x axis is either in units of [1/Angstrom] or integer numbers indicating the number of k points (not shifted, energies correspond to tight-binding material parameters in input file)
- E(k)\_tighten\_new.dat energy dispersion E(k) where the x axis is either in units of [1/Angstrom] or integer numbers indicating the number of k points (shifted so that topmost valence band edge equals 0 eV)
- E(k)\_tighten.dat energy dispersion E(k) (original output) (shifted so that topmost valence band edge equals 0 eV)

### Output files of tighten (bulk)

E(k)\_tighten\_bulk\_new\_noshift.dat energy dispersion E(k) (original output) (not shifted, energies correspond to tight-binding material parameters in input file)

E(k)\_tighten\_bulk\_new.dat energy dispersion E(k) (shifted so that valence band edge of first k vector equals 0 eV)

E(k)\_tighten\_bulk.dat energy dispersion E(k) (original output) (shifted so that valence band edge of Gamma k vector equals 0 eV (default) (shifted so that valence band edge of first k vector equals 0 eV (debug-level = 10)

If k-vectors-sample-type = xyz, then the |k| vectors are in units of [1/nm] of the files out\_ek\_tighten\_bulk\_new\*.dat.

```
hamtightenout_bulk.txt
driver_file_sl.in
'hamtightenout.dat': ham_output_file = 'hamout.dat'
```

## Strain

```
$simulation-flow-control
...
strain-calculation = homogeneous-strain
```

Include biaxial strain for superlattice *tighten* code.

```
strain-calculation = no-strain
```

No strain is considered for superlattice *tighten* code, i.e. strain tensor is zero and each layer has equilibrium lattice constant in growth direction.

In any case, the lattice constant in the lateral growth direction is the one of the substrate material specified in

```
$domain-ccordinates
...
pseudomorphic-on = GaAs
```

for both, homogeneous-strain and no-strain. Without strain, the vertical lattice constant is the equilibrium lattice constant of the layer material.

## 5.3.25 \$tight-binding

This is preliminary. It is not implemented yet.

```
!-----!
$tight-binding                                optional !

tight-binding-on                             character required ! 'yes' / 'no'
atomic-positions-out                         character optional ! 'yes' / 'no'

! -> TIGHT BINDING DRIVER INPUT
screen-output                               integer    optional ! 0 (no screen output), 1_
↪(minimum screen output), 2 (detailed screen output)
input-directory                             character    required ! root path for the whole TB_
↪simulation, e.g. 'TB_data\'
input-parse                                 character    required ! name of the TB parse file,_
↪e.g. 'data_driver'
TIGHTEN-state-filename                      character    required ! name of the reference_
↪states and couplings file, e.g. 'state.dat'

! SUPERCELL
supercell-start                             double_array required ! x,y,z [nm]
supercell-end                               double_array required ! x,y,z [nm]

! -> TIGHT BINDING INPUT
nearest-neighbors-order                     integer      required ! e.g. 1
floating-distance                           character     required ! 'yes' / 'no', e.g. 'yes'
nearest-distance-accuracy                   double        required ! (%), e.g. 20.0
relativistic                               character     required ! 'yes' / 'no', e.g. 'yes'
random-alloy                               character     required ! alloy -> 'yes' / 'no', e.g.
↪'no'
number-of-alloy-loops                       integer      required ! alloy
```

(continues on next page)



(continued from previous page)

|                                                                               |              |                                        |
|-------------------------------------------------------------------------------|--------------|----------------------------------------|
| periodic-boundaries                                                           | character    | required ! 'x','xy','xyz' and cyclic   |
| ↪permutations                                                                 |              |                                        |
| k-vector-filename                                                             | character    | required !                             |
| ! -> BASIS INPUT                                                              |              |                                        |
| basis-output                                                                  | integer      | required ! 0 (none), 1 (generate xmol  |
| ↪xyz file), 2 (generate basis file for strain calculation + xmol xyz file), 3 |              |                                        |
| ↪(calculate strain + generate xmol xyz file), 4 (add strain from data file +  |              |                                        |
| ↪generate xmol xyz file)                                                      |              |                                        |
| xmol-basis-filename                                                           | character    | required !                             |
| strain                                                                        | character    | required ! 'yes' / 'no'                |
| strain-directory                                                              | character    | required ! e.g. 'strain\'              |
| basis-coord-filename                                                          | character    | required ! e.g. 'basis_coord.dat'      |
| basis-strain-filename                                                         | character    | required ! e.g. 'basis_coord_strained. |
| ↪dat'                                                                         |              |                                        |
| potential                                                                     | character    | required ! 'yes' / 'no', e.g. no       |
| potential-filename                                                            | character    | required ! e.g. 'TB_potential.dat'     |
| ! -> SPARSE MATRIX INDEX INPUT                                                |              |                                        |
| sparse-matrix-format                                                          | character    | required ! 'upper'                     |
| sparse-matrix-index-filename                                                  | character    | required ! e.g. 'indices_sparse.dat'   |
| sparse-matrix-values-filename                                                 | character    | required ! e.g. 'values_sparse.dat'    |
| ! -> DIAGONALIZATION INPUT                                                    |              |                                        |
| diagonalization                                                               | character    | required ! 'yes' / 'no', e.g. yes      |
| diagonalization-method                                                        | character    | required ! 'lanczos','davidson'        |
| maximum-number-of-iterations                                                  | integer      | required ! e.g. 100000                 |
| tolerance                                                                     | double       | required ! e.g. 1.0d-6                 |
| number-of-eigenvalues                                                         | integer      | required ! e.g. 20                     |
| eigenvalues-filename                                                          | character    | required ! e.g. 'QW_gap.dat'           |
| result-format                                                                 | integer      | required ! e.g. 0                      |
| guess-energy-value                                                            | double_array | optional ! davidson (complex number in |
| ↪[eV]), e.g. (0.7,0.0) -> '0.7 0.0'                                           |              |                                        |
| eigenstates                                                                   | character    | optional ! davidson 'yes' / 'no', e.g. |
| ↪no                                                                           |              |                                        |
| eigenstates-filename                                                          | character    | optional ! davidson e.g. 'QW_gap.dat'  |
| conduction                                                                    | character    | optional ! lanczos 'yes' / 'no', e.g.  |
| ↪yes                                                                          |              |                                        |
| conduction-guess                                                              | double_array | optional ! lanczos e.g. 1.2 1.2        |
| valence                                                                       | character    | optional ! lanczos 'yes' / 'no', e.g.  |
| ↪yes                                                                          |              |                                        |
| valence-guess                                                                 | double_array | optional ! lanczos e.g. 0.5 0.5        |
| ! -> Dangling bonds saturation                                                |              |                                        |
| saturated-bond-length-factor                                                  | double       | required ! [nm]                        |
| saturated-onsite-energy-H                                                     | double       | required ! [eV]                        |
| \$end_tight-binding                                                           |              | optional !                             |
| !-----!                                                                       |              |                                        |

### 5.3.26 \$warnings

Warnings can be switched on or off. It is recommended to use the default settings which are specified in the database file database\_nn3.in. Currently, changing the default value does not have much effect.

|                       |         |                 |
|-----------------------|---------|-----------------|
| <b>\$warnings</b>     |         | <b>optional</b> |
| warnings              | logical | <b>required</b> |
| <b>\$end_warnings</b> |         | <b>optional</b> |

Specify here if warnings should be written to the .log file.

**warnings****type**

logical

**presence**

required

**options**

.FALSE.

.TRUE.

**default**

uses value specified in database\_nn3.in

Example

```
!-----!  
$warnings  
  warnings = .TRUE.      ! switch on warnings  
$end_warnings  
!-----!
```

### 5.3.27 \$quantum-bound-states

(1D only)

Finds out all eigenfunctions which are localized within a certain region where one expects bound states for example. The quantum states are specified by a certain threshold fraction of  $\psi$  within the region [x-left, x-right]. This is necessary for large quantum clusters which extend far beyond the region of interest and therefore have many irrelevant eigenstates.

|                                   |           |                 |
|-----------------------------------|-----------|-----------------|
| <b>\$quantum-bound-states</b>     |           | <b>optional</b> |
| set-number                        | integer   | <b>required</b> |
| quantum-region                    | integer   | <b>optional</b> |
| num-schroedinger-equation         | integer   | <b>optional</b> |
| charge                            | character | <b>optional</b> |
| x-left                            | double    | <b>optional</b> |
| x-right                           | double    | <b>optional</b> |
| threshold                         | double    | <b>optional</b> |
| <b>\$end_quantum-bound-states</b> |           | <b>optional</b> |

*Syntax*

**set-number****type**

integer

**presence**

required

**example**

1

Number to distinguish different sets of localized states. Has to be in ascending order.

**quantum-region**

**type**  
integer

**example**  
1

Number of quantum cluster in which to look for localized states.

#### **num-schroedinger-equation**

**type**  
integer

**example**  
1

Number of Schrödinger equation in which to look for localized states.

#### **charge**

**type**  
character

**value**  
e1 or h1

Flag whether electrons or holes are regarded.

#### **x-left**

**type**  
double

**unit**  
(nm)

**example**  
20.0

left boundary of localization region

#### **x-right**

**type**  
double

**unit**  
(nm)

**example**  
40.0

right boundary of localization region

#### **threshold**

**type**  
double

**example**  
0.6

Minimum fraction of  $\psi^2$  of certain eigenstate within [x-left, x-right] to be regarded as localized state.

### **5.3.28 \$quantumstate-recombination-rates**

Relevant data for generation and recombination rates in quantum clusters. In quantum clusters, rates for different eigenstates can be provided and a net generation and recombination rate will be calculated.

|                                                |                |                 |                 |
|------------------------------------------------|----------------|-----------------|-----------------|
| <b>\$quantumstate</b> -recombination-rates     |                |                 | <b>optional</b> |
| rate-number                                    | <b>integer</b> | <b>required</b> |                 |
| <b>quantum</b> -region                         | <b>integer</b> | <b>optional</b> |                 |
| x-left                                         | <b>double</b>  | <b>optional</b> |                 |
| x-right                                        | <b>double</b>  | <b>optional</b> |                 |
| threshold-frac                                 | <b>double</b>  | <b>optional</b> |                 |
| vb-eigenstate                                  | <b>integer</b> | <b>optional</b> |                 |
| cb-eigenstate                                  | <b>integer</b> | <b>optional</b> |                 |
| netto-gen-rate                                 | <b>double</b>  | <b>optional</b> |                 |
| <b>\$end_quantumstate</b> -recombination-rates |                |                 | <b>optional</b> |

*Syntax***rate-number****type**

integer

**presence**

required

**example**

1

It has to be numbered from 1, 2, 3, ...

**quantum-region****type**

integer

**example**

1

number of quantum-cluster

**x-left****type**

integer

**unit**

(nm)

**example**

10.0

left boundary of relevant integration range

**x-right****type**

integer

**unit**

(nm)

**example**

20.0

right boundary of relevant integration range

**threshold-frac****type**

double

**example**  
20.0

The quantum states are specified by a certain threshold fraction of  $\psi^2$  within the region [x-left, x-right]. This is necessary for large quantum regions which extend far beyond the region of interest and therefore have many irrelevant eigenstates. The generation rate is proportional to  $\text{netto\_gen\_rate} * \psi V(x)^2 * \text{Fermi\_function}(x)$ .

**vb-eigenstate**

**type**  
integer

**example**  
1

relevant valence band eigenstate number

**cb-eigenstate**

**type**  
integer

**example**  
1

relevant conduction band eigenstate number

**netto\_gen\_rate**

**type**  
double

**example**  
?

**unit**  
?

net generation rate

### 5.3.29 \$NEGF-spintronics

NEGF for spintronics: Spin transport based on the Non-equilibrium Green's functions (NEGF) method.

This part is based on the PhD thesis of [\[KubisPhD2009\]](#).

This is preliminary. It is not useful for nondevelopers.

|                               |                     |                                            |
|-------------------------------|---------------------|--------------------------------------------|
| <b>\$NEGF-spintronics</b>     |                     | <b>optional</b> !                          |
| NEGF-spintronics-on           | <b>character</b>    | <b>required</b> ! yes/no                   |
| directory-NEGF-spintronics    | <b>character</b>    | <b>optional</b> !                          |
| include-spin                  | <b>character</b>    | <b>optional</b> ! yes/no                   |
| include-scattering            | <b>character</b>    | <b>optional</b> ! yes/no                   |
| scattering-iterations         | <b>integer</b>      | <b>optional</b> !                          |
| grid-points-in-energy         | <b>integer</b>      | <b>optional</b> ! number of grid points in |
| ↪energy                       |                     |                                            |
| minimum-energy                | <b>double</b>       | <b>optional</b> ! minimum of considered    |
| ↪energies in [eV]             |                     |                                            |
| bulk-inversion-asymmetry      | <b>double</b>       | <b>optional</b> ! BIA [eVnm]               |
| structure-inversion-asymmetry | <b>double</b>       | <b>optional</b> ! SIA (Rashba) [eVnm]      |
| magnetic-field-B-left         | <b>double_array</b> | <b>optional</b> ! B_x, B_y, B_z [T]        |
| magnetic-field-B-right        | <b>double_array</b> | <b>optional</b> ! B_x, B_y, B_z [T]        |
| magnetic-field-B-up           | <b>double_array</b> | <b>optional</b> ! B_x, B_y, B_z [T]        |
| magnetic-field-B-down         | <b>double_array</b> | <b>optional</b> ! B_x, B_y, B_z [T]        |
| Fermi-level-left              | <b>double</b>       | <b>optional</b> ! [eV]                     |
| Fermi-level-right             | <b>double</b>       | <b>optional</b> ! [eV]                     |

(continues on next page)

(continued from previous page)

|                               |           |                   |
|-------------------------------|-----------|-------------------|
| Fermi-level-up                | double    | optional ! [eV]   |
| Fermi-level-down              | double    | optional ! [eV]   |
| magnetic-contacts             | character | optional ! yes/no |
| contacts-left-and-right       | character | optional ! yes/no |
| calculate-contact-left        | character | optional ! yes/no |
| calculate-contact-right       | character | optional ! yes/no |
| calculate-contact-up          | character | optional ! yes/no |
| calculate-contact-down        | character | optional ! yes/no |
| calculate-transmission        | character | optional ! yes/no |
| <b>\$end_NEGF-spintronics</b> |           | optional !        |

### 5.3.30 \$global-parameters

Global parameters are general parameters which are valid all over the device. See also *\$global-parameters* under database section.

|                                   |           |          |
|-----------------------------------|-----------|----------|
| <b>\$global-parameters</b>        |           | optional |
| lattice-temperature               | double    | required |
| temperature-sweep-active          | character | optional |
| temperature-sweep-step-size       | double    | optional |
| temperature-sweep-number-of-steps | integer   | optional |
| data-out-every-nth-step           | integer   | optional |
| <b>\$end_global-parameters</b>    |           | optional |

The lattice temperature is given in Kelvin. The lowest allowed temperature can be set in *\$program\_restrictions*.

#### **Note**

*\$global-parameters* is optional. If it is not present, the default value for the temperature is taken from *\$global-parameters* of the database file. Currently, the default temperature is set to 300 K.

#### **Note**

The band gaps (namely the conduction band edges only) are adjusted when the temperature changes. This feature can be switched off. See *\$numeric-control: varshni-parameters-on = no*

In addition, the lattice constants depend on temperature. This can be switched off as well. See *\$numeric-control: lattice-constants-temp-coeff-on = no*

### Temperature sweep

It is possible to sweep over the temperature, i.e. to vary the temperature stepwise. This is similar to

- electric field sweep (*\$electric-field*)
- magnetic field sweep (*\$magnetic-field*)
- voltage sweep (*\$voltage-sweep*)
- doping concentration sweep (*\$doping-function*)
- alloy sweep (*\$alloy-function*).

The output is labeled with *...\_ind000.dat*, *...\_ind001.dat*, *...\_ind002.dat*, ... where the index refers to the number of the temperature sweep step.

Example

```

!-----!
$global-parameters
  lattice-temperature          = 300.0 ! [K]

  temperature-sweep-active     = yes   ! yes/no
  temperature-sweep-step-size  = -10.0 ! [K], i.e. in this case -10 K
  temperature-sweep-number-of-steps = 20 ! number of temperature sweep
→ steps
  data-out-every-nth-step      = 10    ! (optional, default = 1)

$end_global-parameters
!-----!

```

In this example, the temperature is varied starting from 300 K, and then reducing the temperature 20 times in steps of -10.0, ending at 100 K. Obviously, increasing the temperature is also possible. If you do not want to write out all data specified in the output section for every step, you have to enter an integer number greater than one. If you specify 1, then output files will be generated for each temperature sweep. This is useful if you want to fill out all unused space on your hard disk.

#### Restrictions

```

flow-scheme = 13
flow-scheme = 130
flow-scheme = 14
flow-scheme = 140

```

have some special functionality so far.

130/140: same as 13/14 but including self-consistent Poisson-Schrödinger.

```

flow-scheme = 13: vary temperature as T
  lattice-temperature          = 5.0 ! 5 Kelvin
  temperature-sweep-active     = yes !
  temperature-sweep-step-size  = 5.0 ! increase by 5 K
  temperature-sweep-number-of-steps = 250 !
  data-out-every-nth-step      = 50 !

flow-scheme = 14: vary temperature as 1000/T
  lattice-temperature          = 1250 ! 1250 Kelvin
  temperature-sweep-active     = yes !
  temperature-sweep-step-size  = -5.0 ! decrease by 5 K
  temperature-sweep-number-of-steps = 250 !
  data-out-every-nth-step      = 50 !

```

See tutorial [Electron concentration in doped semiconductors \(Si, Ge, GaAs\)](#) for details.

### 5.3.31 \$output-grid

Write grid coordinates to output file. (This keyword is not very useful.)

|                          |                      |                 |
|--------------------------|----------------------|-----------------|
| <b>\$output-grid</b>     |                      | <b>optional</b> |
| grid-coordinate          | <b>integer_array</b> | <b>required</b> |
| output-file              | <b>character</b>     | <b>required</b> |
| <b>\$end_output-grid</b> |                      | <b>optional</b> |

Example

```

!-----!
$output-grid
!
```

(continues on next page)

(continued from previous page)

```

grid-coordinate = 1 0 0      ! specify coordinate axis (1 0 0) or (0
→1 0) or (0 0 1)
output-file     = grid_x.dat  ! specify output file name
!
grid-coordinate = 0 1 0      !
output-file     = grid_y.dat  !
$end_output-grid            !
!-----!

```

### 5.3.32 \$output-geometry

Write output files that contain for the material grid points:

- cluster numbers
- quantum cluster numbers
- current cluster numbers

|                              |                  |                 |
|------------------------------|------------------|-----------------|
| <b>\$output-geometry</b>     |                  | <b>optional</b> |
| cluster-numbers              | <b>character</b> | <b>required</b> |
| quantum-cluster-numbers      | <b>character</b> | <b>optional</b> |
| current-cluster-numbers      | <b>character</b> | <b>optional</b> |
| <b>\$end_output-geometry</b> |                  | <b>optional</b> |

*Example*

The output filenames can be chosen by the user.

```

$output-geometry
cluster-numbers      = geometry_cluster-numbers.dat
quantum-cluster-numbers = geometry_quantum-cluster-numbers.dat
current-cluster-numbers = geometry_current-cluster-numbers.dat
$end_output-geometry

```

### 5.3.33 \$output-raw-data

*Unformatted data output*

In order to restart the calculation not from the very beginning but from the last calculated step on has to save the data including all internal structures. This is done by the keyword `$output-raw-data`. In order to keep files small, the format is binary and is not readable by outside programs.

The unformatted data can be reused in the program to continue some calculations. For details see for example flow-scheme in *\$simulation-flow-control*.

|                              |                  |                 |
|------------------------------|------------------|-----------------|
| <b>\$output-raw-data</b>     |                  | <b>optional</b> |
| destination-directory        | <b>character</b> | <b>required</b> |
| potential                    | <b>character</b> | <b>optional</b> |
| fermi-levels                 | <b>character</b> | <b>optional</b> |
| kp-eigenstates               | <b>character</b> | <b>optional</b> |
| <b>strain</b>                | <b>character</b> | <b>optional</b> |
| <b>\$end_output-raw-data</b> |                  | <b>optional</b> |

*Syntax*

```

destination-directory
    type
        character

```



**presence**  
required  
**example**  
my-directory/  
**example**  
raw\_data/

Name of directory to which the files should be written. The folder name has to include the backslash \ (or slash /).

#### **potential**

**type**  
character  
**presence**  
optional  
**options**  
yes or no  
**name of output file**  
potentials\_store1D/2D/3D.raw  
**file format**  
binary (unformatted)

Flag whether to save the electrostatic potential.

#### **fermi-levels**

**type**  
character  
**presence**  
optional  
**options**  
yes or no  
**name of output file**  
fermi\_store1D/2D/3D.raw  
**file format**  
binary (unformatted)

Flag whether to save the quasi-Fermi levels for electrons and holes.

#### **kp-eigenstates**

**type**  
character  
**presence**  
optional  
**options**  
yes or no  
**name of output file**  
schroedinger\_store1D/2D/3D.raw  
**name of output file**  
kp\_store1D/2D/3D.raw  
**file format**  
binary (unformatted)

Flag whether to save the single-band or  $\mathbf{k} \cdot \mathbf{p}$  eigenstates and wave functions.

**strain****type**

character

**presence**

optional

**options**

yes or no

**name of output file**

strain\_store1D/2D/3D.raw

**file format**

binary (unformatted)

Flag whether to save the strain tensor.

*Example*

```
$output-raw-data
destination-directory = raw_data/
potential             = yes      ! 'yes' or 'no'
fermi-levels          = yes      ! 'yes' or 'no'
kp-eigenstates        = yes      ! 'yes' or 'no'
strain                = yes      ! 'yes' or 'no'
$end_output-raw-data
```

**5.3.34 \$input\_filename**

Specification of input file name

This is optional (and deprecated). It is much more convenient to start the simulation using command line argument `--inputfile "D:\test\Quantum Well.in"`. The only reason for the existence of this keyword is to start a simulation without command line arguments, e.g. for testing purposes. The name of the input file to be run can be written into the file `keywords.val` within the keyword `$input_filename` which must be the **first** (!) entry in this file. Apart from this modification `keywords.val` must not be changed.

The specification of the input filename is done as a first entry after `$input_filename` in the file `keywords.val` as follows:

|                             |                  |                 |
|-----------------------------|------------------|-----------------|
| <b>\$input_filename</b>     |                  | <b>optional</b> |
| input_filename              | <b>character</b> | <b>optional</b> |
| <b>\$end_input_filename</b> |                  | <b>optional</b> |

*Example 1*

```
!-----!
$input_filename      optional !
input_file1.in      character optional ! Reads in "input_file1.in".
!1D_Quantum_well.in character optional ! Reads in "1D_Quantum_well.in
↪" if the comment sign "!" is removed.
$end_input_filename  optional !
!-----!
```

A comment can be inserted using ! or #.

*Example 1*

|                                             |                  |                 |                |
|---------------------------------------------|------------------|-----------------|----------------|
| <b>\$input_filename</b>                     |                  | <b>optional</b> | <b>!</b>       |
| "D:\My nextnano input files\input_file1.in" | <b>character</b> | <b>optional</b> | <b>! Reads</b> |

(continues on next page)

(continued from previous page)

```

→in "input_file1.in".
$send_input_filename                                optional !
!-----!

```

### 5.3.35 \$Monte-Carlo

Monte Carlo transport: Here, the parameters for the Monte Carlo procedure and the mobility calculations are defined.

|                                          |                     |                           |
|------------------------------------------|---------------------|---------------------------|
| <b>\$Monte-Carlo</b>                     | <b>optional</b>     | <b>!</b>                  |
| Monte-Carlo-transport                    | <b>character</b>    | <b>required !</b>         |
| destination-directory                    | <b>character</b>    | <b>required !</b>         |
| grid-position                            | <b>double_array</b> | <b>required ! [nm]</b>    |
| energy-dispersion                        | <b>character</b>    | <b>required !</b>         |
| charge-carrier-type                      | <b>character</b>    | <b>required !</b>         |
| number-of-simulated-carriers             | <b>integer</b>      | <b>required !</b>         |
| number-of-timesteps                      | <b>integer</b>      | <b>required !</b>         |
| timestep                                 | <b>double</b>       | <b>required ! [fs]</b>    |
| number-of-electric-field-steps           | <b>integer</b>      | <b>required !</b>         |
| electric-field-start-value               | <b>double_array</b> | <b>required ! [V/m]</b>   |
| electric-field-step                      | <b>double</b>       | <b>required ! [V/m]</b>   |
| electric-field-step-factor               | <b>double</b>       | <b>required !</b>         |
| start-averaging-after-timesteps          | <b>integer</b>      | <b>required !</b>         |
| execute-averaging-after-timesteps        | <b>integer</b>      | <b>required !</b>         |
| impurity-background-doping-concentration | <b>double</b>       | <b>required ! [cm-3]</b>  |
| doping-concentration                     | <b>double</b>       | <b>required ! [cm-3]</b>  |
| LA-phonon-scattering                     | <b>character</b>    | <b>optional !</b>         |
| LO-phonon-scattering                     | <b>character</b>    | <b>optional !</b>         |
| TA-phonon-scattering                     | <b>character</b>    | <b>optional !</b>         |
| TO-phonon-scattering                     | <b>character</b>    | <b>optional !</b>         |
| acoustic-phonon-scattering               | <b>character</b>    | <b>optional !</b>         |
| polar-optical-phonon-scattering          | <b>character</b>    | <b>optional !</b>         |
| plasmon-scattering                       | <b>character</b>    | <b>optional !</b>         |
| ionized-impurity-scattering              | <b>character</b>    | <b>optional !</b>         |
| surface-roughness-scattering             | <b>character</b>    | <b>optional !</b>         |
| electron-hole-scattering                 | <b>character</b>    | <b>optional !</b>         |
| impact-ionization-scattering             | <b>character</b>    | <b>optional !</b>         |
|                                          |                     | <b>!</b>                  |
| alloy-scattering                         | <b>character</b>    | <b>optional !</b>         |
| alloy-disorder-scattering-potential      | <b>double</b>       | <b>optional ! [eV]</b>    |
|                                          |                     | <b>!</b>                  |
| mass-density                             | <b>double</b>       | <b>optional ! [kg/m3]</b> |
| sound-velocity                           | <b>double</b>       | <b>optional ! [m/s]</b>   |
| acoustic-deformation-potential           | <b>double</b>       | <b>optional ! [eV]</b>    |
|                                          |                     | <b>!</b>                  |
| <b>quantum</b> -well-width               | <b>double</b>       | <b>optional ! [nm]</b>    |
| spacer-width                             | <b>double_array</b> | <b>optional ! [nm]</b>    |
| remote-doping-sheet-density              | <b>double_array</b> | <b>optional ! [cm-2]</b>  |
| 2DEG-sheet-density                       | <b>double</b>       | <b>optional ! [cm-2]</b>  |
| 2DEG-sheet-density-number-of-subbands    | <b>double</b>       | <b>optional !</b>         |
| <b>\$end_Monte-Carlo</b>                 | <b>optional</b>     | <b>!</b>                  |

*Syntax*

```
destination-directory = Monte_Carlo/
```

Name of directory to which the Monte Carlo files should be written.  
Directory name has to include the slash.

```
Monte-Carlo-transport = yes      ! Monte Carlo transport calculations switched on
                        = no      ! Monte Carlo transport calculations switched off
                        = simple  ! Monte Carlo transport calculations switched off and
↪ a simple 2DEG algorithm is employed
```

```
grid-position = 10.0           ! grid position in units of [nm]
                                ! take material parameters from the material
↪ located at this point
grid-position = 10.0           ! x = 10 [nm] (1D)
                = 10.0 20.0     ! x = 10 [nm], y = 20 [nm] (2D)
                = 10.0 20.0 20.0 ! x = 10 [nm], y = 10 [nm], z = 20 [nm] (3D)
```

```
energy-dispersion = kp        ! use nonparabolic and anisotropic energy dispersion
↪ E(k) calculated from k.p theory, i.e.
                                ! take into account k.p dispersion into
↪ scattering rates, overlap factors and selection of final states
                = parabolic ! use parabolic and isotropic energy
↪ dispersion E(k) (single-band effective-mass approximation), i.e.
                                ! take into account parabolic dispersion into
↪ scattering rates, overlap factors and selection of final states
```

```
charge-carrier-type = holes    ! Monte Carlo transport calculations for holes
                    = electrons ! (not implemented yet)
```

```
number-of-simulated-carriers = 10000 ! number of simulated electrons or holes,
↪ respectively
                                ! (depending on charge-carrier-type = ...)
```

```
number-of-timesteps = 600 ! number of timesteps
timestep             = 10.0 ! timestep in units of [fs]
```

total time = number-of-timesteps \* timestep = 600 \* 10 fs = 6000 fs = 6 ps

```
number-of-electric-field-steps = 5      ! number of calculated points for
↪ increasing electrical field (starting value: 1 kV/cm)
electric-field-start-value      = 1.0e5 0.0 ! [V/m], F = (Fx,Fy), i.e. in this case
↪ Fx = 1.0 * 10^5 V/m = 1 kV/cm,
                                ! Fy = 0 V/m
electric-field-step             = 1.0e3  ! increase electric field strength along
↪ the channel by this magnitude (for each electric field step)
                                ! [V/m], i.e. in this case 1.0 * 10^5 V/m,
↪ Units: 1 kV/cm = 1 * 10^5 V/m = 1e5 V/m
                                ! (1.0d3 = 0.01 kV/cm = 1000 V/m = 1 kV/m
↪ = 10 V/cm)
electric-field-step-factor      = 1.1    ! increase electric field strength along
↪ the channel by this factor (for each electric field step)
                                = 1.0     ! [] linear increase
```

$$F_{x,i+1} = F_{x,i} * \text{electric-field-step-factor} + \text{electric-field-step}$$

$F_{y,i+1} = F_{y,i}$  where  $F_x$  and  $F_y$  are the electric field magnitude along the  $x$  or  $y$  direction, respectively.

```

start-averaging-after-timesteps = 400 ! number of timesteps after which the
↪averaging mechanism is started
execute-averaging-after-timesteps = 1 ! number of timesteps after which the
↪averaging mechanism is executed
! (= 1: each timestep is used for averaging)

```

```

impurity-background-doping-concentration = 0.5e15 ! in units of [cm^-3] ==> 0.5e15
↪= 0.5 * 10^15 cm^-3

```

The impurity background doping concentration which is relevant for the impurity scattering mechanism is given in units of [cm<sup>-3</sup>]. Doping with fully ionized dopants is assumed in this case (during the Monte Carlo procedure only). The whole simulation region will be covered with this background doping during the Monte Carlo procedure. It will contribute additively to the additional doping regions specified inside the input file (*\$doping-function*). It also enters the scattering rates for impurity scattering: If it is set to zero, the integral will not converge, so do not do it...

```

doping-concentration = 1e18 ! in units of [cm^-3] ==> 1e18 = 1 * 10^18 cm^-3

```

The doping concentration is given in units of [cm<sup>-3</sup>]. This value for the doping concentration is used for the impurity scattering table. It is relevant for electron-hole scattering and ionized impurity scattering.

*Flags to switch on/off certain scattering models*

```

LA-phonon-scattering = yes ! include LA phonon scattering (default)
                      = no  ! switch off LA phonon scattering

```

This affects the subroutines

- total scattering probability due to LA phonon emission in first conduction band
- total scattering probability due to LA phonon emission among hole bands
- total scattering probability due to LA phonon absorption in first conduction band
- total scattering probability due to LA phonon absorption among hole bands

```

LO-phonon-scattering = yes ! include LO phonon scattering (default)
                      = no  ! switch off LO phonon scattering

```

This affects the subroutines

- total scattering probability due to LO phonon emission in first conduction band
- total scattering probability due to LO phonon emission among hole bands
- total scattering probability due to LO phonon emission among 2D hole subbands
- total scattering probability due to LO phonon absorption in first conduction band
- total scattering probability due to LO phonon absorption among hole bands
- total scattering probability due to LO phonon absorption among 2D hole subbands

```

TA-phonon-scattering = yes ! include TA phonon scattering (default)
                      = no  ! switch off TA phonon scattering

```

This affects the subroutines

- total scattering probability due to TA phonon emission in first conduction band
- total scattering probability due to TA phonon emission among hole bands
- total scattering probability due to TA phonon absorption in first conduction band
- total scattering probability due to TA phonon absorption among hole bands

```
TO-phonon-scattering = yes ! include    TO phonon scattering (default)
                     = no  ! switch off TO phonon scattering
```

This affects the subroutines

- total scattering probability due to TO phonon emission in first conduction band
- total scattering probability due to TO phonon emission among hole bands
- total scattering probability due to TO phonon absorption in first conduction band
- total scattering probability due to TO phonon absorption among hole bands

```
acoustic-phonon-scattering = yes ! include    acoustic phonon scattering (default)
                           = no  ! switch off acoustic phonon scattering
```

This affects the subroutines

- total scattering probability due to acoustic scattering in first conduction band
- total scattering probability due to acoustic scattering among hole bands
- total scattering probability due to acoustic scattering among 2D hole subbands

The acoustic phonon scattering rates are linear functions of temperature.

```
polar-optical-phonon-scattering = yes ! include    polar optical phonon scattering_
↪(default)
                               = no  ! switch off polar optical phonon scattering
```

This affects the subroutines

- total scattering probability due to polar optical phonon emission
- total scattering probability due to polar optical phonon absorption

```
plasmon-scattering = yes ! include    plasmon scattering (default)
                   = no  ! switch off plasmon scattering
```

This affects the subroutines

- total scattering probability due to heavy hole - plasmon absorption in first conduction band
- total scattering probability due to Gamma - plasmon absorption in hole bands
- total scattering probability due to heavy hole - plasmon emission in first conduction band
- total scattering probability due to Gamma - plasmon emission in hole band

```
ionized-impurity-scattering = yes ! include    TO phonon scattering (default)
                             = no  ! switch off TO phonon scattering
```

This affects the subroutines

- total scattering probability due to ionized impurities
- total scattering probability due to ionized impurity scattering among 2D hole subbands

```
surface-roughness-scattering = yes ! include    surface roughness scattering_
↪(default)
                             = no  ! switch off surface roughness phonon scattering
```

This affects the subroutines

- total scattering probability due to surface roughness scattering among 2D hole subbands

```
electron-hole-scattering = yes ! include electron-hole scattering (default)
                        = no  ! switch off electron-hole scattering
```

This affects the subroutines

- total scattering probability due to electron-hole scattering in first conduction band
- total scattering probability due to electron-hole scattering in hole bands

```
impact-ionization-scattering = yes ! include impact ionization scattering (Kane
↪model) (default)
                             = no  ! switch off impact ionization scattering (Kane
↪model)
```

This affects the subroutines

- total scattering probability due to impact ionization scattering (Kane model)

```
alloy-scattering = yes ! include alloy scattering (default)
                 = no  ! switch off alloy scattering
```

This affects the subroutines

- total scattering probability due to alloy scattering in first conduction band
- total scattering probability due to alloy scattering among hole bands

```
alloy-disorder-scattering-potential = 1.0 ! [eV] (optional parameter)
                                     = 1.0 ! [eV] 1 eV for AlGaAs
                                     = 0.6 ! [eV] ~0.6 eV for InGaAs
```

If alloy-disorder-scattering-potential is not present, then this value will be calculated internally from the conduction (or valence) band offset of the two binary end points for each grid point, e.g. Al(x)Ga(1-x)As: CBO(AlAs) - CBO(GaAs). Some more information about the used parameters for each grid point is contained in the file AlloyScatteringInfo.dat.

#### Material parameters

The following parameters are used for deformation potential acoustic phonon scattering.

```
mass-density           = 5.79e3 ! [kg/m3] lattice density
sound-velocity         = 3.7e3  ! [m/s]  velocity of longitudinal elastic
↪waves
acoustic-deformation-potential = 7.2 ! [eV]
```

The following parameters are used for the *simple algorithm* (for delta-doped 2DEGs) to calculate the mobility.

```
quantum-well-width      = 10.0 ! [nm]
spacer-width            = 20.0 ! [nm]
remote-doping-sheet-density = 1e12 ! [cm-2]
2DEG-sheet-density      = 0.071e12 ! [cm-2]
2DEG-sheet-density-number-of-subbands = 3 ! (default: 1)
```

If 2DEG-sheet-density is not present, then the 2DEG density is calculated automatically where the number of subbands that are taken into account can be specified (default: 1). If two remote doping regions should be taken into account, one can input an array of values.

```
spacer-width           = 20.0 10.0 ! [nm] spacer width of first and 2nd
↪doping region
remote-doping-sheet-density = 1e12 1e11 ! [cm-2] remote doing density of first and
↪2nd doping region
```

### Monte Carlo

The Monte Carlo algorithm that is used inside *nextnano*<sup>3</sup> is similar to the following publication:

Subband structure and mobility of two-dimensional holes in strained Si/SiGe MOSFET's  
R. Oberhuber, G. Zandler and P. Vogl  
Physical Review B 58, 9941 (1998)

### Details

The carrier transport in quantized 2D channels is computed in terms of a momentum space ensemble Monte Carlo procedure for a spatially homogeneous channel with constant electric fields invoking consistently calculated 2D scattering rates (see subroutine `scattering` for details on scattering).

For electrons and a single conduction band, a similar procedure was carried out previously (M.V. Fischetti, S.E. Laux, PRB 48, 2244 (1993)).

The equations of motion are integrated with the numerically determined, fully nonparabolic subband dispersions  $E(k)$  in analogy to full band Monte Carlo methods that have been developed previously for bulk (M.V. Fischetti, S.E. Laux, PRB 38, 9721 (1988)).

Both the hole dispersion relations  $E(k)$  and the inverse relation  $k(E)$  have been determined by a 2D discretization of  $k$  space.

The Monte Carlo method requires both the individual as well as the total scattering rates. The latter can only be obtained, for each initial state in subband ... with energy  $E(k)$ , by a numerical integration of Eq. (...) over all final states. This amounts to integrating over the wavevectors  $q, q$  with the numerically determined 2D band structure.

*Restrictions:* So far, this Monte Carlo procedure only makes sense for 1D simulations with quantum confinement.

Note: Do not forget to specify the parameters for the density of states (DOS) calculation. The DOS is used for the calculation of the scattering tables.

### \$output-kp-data

```
...  
DOS-density-of-states = ...  
DOS-Emin-Emax       = ...  
DOS-points           = ...
```

### Scattering mechanisms

- polar optical phonon emission
- ionized impurities
- polar optical phonon absorption
- LO phonon emission in first conduction band
- LO phonon emission among hole bands
- LO phonon emission among 2D hole subbands
- LO phonon absorption in first conduction band
- LO phonon absorption among hole bands
- LO phonon absorption among 2D hole subbands
- TO phonon emission in first conduction band
- TO phonon emission among hole bands
- TO phonon absorption in first conduction band
- TO phonon absorption among hole bands
- LA phonon emission in first conduction band
- LA phonon emission among hole bands



- LA phonon absorption in first conduction band
- LA phonon absorption among hole bands
- TA phonon emission in first conduction band
- TA phonon emission among hole bands
- TA phonon absorption in first conduction band
- TA phonon absorption among hole bands
- acoustic scattering in first conduction band
- acoustic scattering among hole bands
- acoustic scattering among 2D hole subbands
- alloy scattering in first conduction band
- alloy scattering among hole bands
- electron-hole scattering in first conduction band
- electron-hole scattering in hole bands
- heavy hole plasmon absorption in first conduction band
- Gamma plasmon absorption in hole bands
- heavy hole plasmon emission in first conduction band
- Gamma plasmon emission in hole band
- surface roughness scattering among 2D hole subbands
- ionized impurity scattering among 2D hole subbands

## Output

*Output files for  $k.p$  data*

*Density of states (DOS)*

The two-dimensional density of states (DOS) is written to the files `Schroedinger_kp/DOS_hl_6x6kp.dat` and `DOS_hl_6x6kp_sum.dat` (and similar for electrons). The DOS has been calculated from the energy dispersion  $E(\mathbf{k}) = E(k_x, k_y)$  and is used inside the Monte Carlo procedure (e.g. scattering tables).

For details, see [\*\\$output-kp-data\*](#).

*Output files for transport data*

After the total simulation time is reached, the averaged transport quantities are written to the output files with the subroutine `write_outputMC`. This is done for each value of the applied electric field. The results of the calculated transport quantities are stored in the files:

- `mobility10.dat`
- `mobility11.dat`
- `subband_energy.dat`
- `subband_velocity_x.dat`
- `subband_density.dat`

The growth direction is along [001], so the carrier channel is perpendicular to this direction. The electric field is applied in the [100] direction, so the output file `mobility10.dat` contains the mobility into this direction. To investigate the anisotropy in the two-dimensional plane, also the mobility calculated from the velocity in [110] direction is stored (`mobility11.dat`).

**mobility10.dat**

This file contains the calculated low field mobility, both calculated from the velocity (4th column of the file) and the diffusion constant average (5th column) obtained by the Monte Carlo procedure. The value of the electric field in [kV/cm] is given in the 1st column, and the corresponding velocity in x direction, average energy, mobility from velocity and mobility from diffusion constant are listed in the following columns.

electric field [kV/cm] vx [cm/s] Eav [eV]  $\mu_{v,x}$  [cm<sup>2</sup>/Vs]  $\mu_{Diff,x}$  [cm<sup>2</sup>/Vs]

$\mu_{v,x} = v_x / F_x$  where  $F_x$  is the *electric field* denoted in the 1st column.

**mobility11.dat**

This file contains the mobility along the [11] direction in the two-dimensional k space, and is similar to the file `mobility10.dat` discussed above.

electric field [kV/cm] vxy [cm/s] Eav [eV]  $\mu_{v,xy}$  [cm<sup>2</sup>/Vs]

**subband\_energy.dat**

This file contains for each electric field value (1st column) the average kinetic energy within each subband (2nd column: subband 1, 3rd column: subband 2, ...) in units of [eV].

electric field [kV/cm] Eav,1 Eav,2 Eav,3 Eav,4 Eav,5 Eav,6 ...

**subband\_velocity\_x.dat**

This file contains the average velocity in each subband in units of [cm/s]. Its structure is similar to the previous file.

electric field [kV/cm] vav,1 vav,2 vav,3 vav,4 vav,5 vav,6 ...

**subband\_density.dat**

This file gives the occupation of each subband.  $N_{av,i}$  is the average number of particles in subband  $i$ . The sum over all  $N_{av,i}$  must be equal to the number of particles that have been simulated, i.e. `number-of-simulated-carriers`.

electric field [kV/cm] Nav,1 Nav,2 Nav,3 Nav,4 Nav,5 Nav,6 ...

The following keywords are not documented yet in the new documentation, i.e. they are linked to the old documentation.

### 5.3.36 \$domain-coordinates

The documentation for this keyword is available here: [https://www.nextnano.com/nextnano3/input\\_parser/keywords/domain-coordinates.htm](https://www.nextnano.com/nextnano3/input_parser/keywords/domain-coordinates.htm)

### 5.3.37 \$material

The documentation for this keyword is available here: [https://www.nextnano.com/nextnano3/input\\_parser/keywords/material.htm](https://www.nextnano.com/nextnano3/input_parser/keywords/material.htm)

### 5.3.38 \$strain-minimization-model

The documentation for this keyword is available here: [https://www.nextnano.com/nextnano3/input\\_parser/keywords/strain-minimization-model.htm](https://www.nextnano.com/nextnano3/input_parser/keywords/strain-minimization-model.htm)

### 5.3.39 \$poisson-boundary-conditions

The documentation for this keyword is available here: [https://www.nextnano.com/nextnano3/input\\_parser/keywords/poisson-boundary-conditions.htm](https://www.nextnano.com/nextnano3/input_parser/keywords/poisson-boundary-conditions.htm)

### 5.3.40 \$quantum-dot-layer-density

The documentation for this keyword is available here: [https://www.nextnano.com/nextnano3/input\\_parser/keywords/quantum-dot-layer-density.htm](https://www.nextnano.com/nextnano3/input_parser/keywords/quantum-dot-layer-density.htm)

### 5.3.41 \$output-strain

The documentation for this keyword is available here: [https://www.nextnano.com/nextnano3/input\\_parser/keywords/output-strain.htm](https://www.nextnano.com/nextnano3/input_parser/keywords/output-strain.htm)

### 5.3.42 \$output-bandstructure

The documentation for this keyword is available here: [https://www.nextnano.com/nextnano3/input\\_parser/keywords/output-bandstructure.htm](https://www.nextnano.com/nextnano3/input_parser/keywords/output-bandstructure.htm)

### 5.3.43 \$output-1-band-schroedinger

The documentation for this keyword is available here: [https://www.nextnano.com/nextnano3/input\\_parser/keywords/output-1-band-schroedinger.htm](https://www.nextnano.com/nextnano3/input_parser/keywords/output-1-band-schroedinger.htm)

The output of the eigenvalues and eigenfunctions for the single-band Schrödinger equations (effective-mass) is controlled by this keyword. All eigenfunctions and eigenvalues between `cb-min-ev` and `cb-max-ev` are written out for each band.

|                                         |               |                 |
|-----------------------------------------|---------------|-----------------|
| <b>\$output-1-band-schroedinger</b>     |               | <b>optional</b> |
| destination-directory                   | character     | required        |
| sg-structure                            | character     | optional        |
| effective-mass-tensor                   | character     | optional        |
| complex-wave-functions                  | character     | optional        |
| eigenvalues-only                        | character     | optional        |
| scale                                   | double        | optional        |
| shift-wave function-by-eigenvalue       | character     | optional        |
| conduction-band-numbers                 | integer_array | optional        |
| cb-min-ev                               | integer       | optional        |
| cb-max-ev                               | integer       | optional        |
| valence-band-numbers                    | integer_array | optional        |
| vb-min-ev                               | integer       | optional        |
| vb-max-ev                               | integer       | optional        |
| interband-matrix-elements               | character     | optional        |
| intraband-matrix-elements               | character     | optional        |
| intraband-lifetime                      | character     | optional        |
| intraband-matrix-elements-operator      | character     | optional        |
| stark-effect-out                        | character     | optional        |
| voltage-offset                          | double        | optional        |
| lever-arm-length                        | double        | optional        |
| resonance-bound-states                  | integer_array | optional        |
| resonance-incidence                     | integer_array | optional        |
| <b>\$end_output-1-band-schroedinger</b> |               | <b>optional</b> |

*Example*

```

$output-1-band-schroedinger

destination-directory      = Schroedinger_1band/
sg-structure               = yes
effective-mass-tensor     = yes
complex-wave-functions    = yes
scale                     = 1.0

```

(continues on next page)

(continued from previous page)

```

conduction-band-numbers          = 1 2 3
cb-min-ev                        = 1
cb-max-ev                        = 10

valence-band-numbers            = 1 2 3
vb-min-ev                       = 1
vb-max-ev                       = 10

interband-matrix-elements       = yes
intraband-matrix-elements      = yes

$end_output-1-band-schroedinger

```

*Syntax***destination-directory****example**

```
my-directory/, Schroedinger_1band/
```

Name of directory to which the files should be written. Directory name has to include the slash.

**sg-structure****example**

```
yes or no
```

**default**

```
no
```

Flag whether to write out the Schrödinger structure file (sg\_info.txt). This file describes the internal number and degeneracy of the Schrödinger equations that have to be solved.

**Note**

If the energy bands are split due to strain, e.g. X valley and L valley, then the Schrödinger equation has to be solved for different band edges. If the masses are anisotropic, then for each mass valley a separate Schrödinger equation has to be solved.

**Note**

The files here have different labels:

cb3 = conduction band no. 3 (1 = Gamma band, 2 = L band, 3 = X band)

qc1 = quantum cluster no. 1

sg3 = no. of Schrödinger equation to be solved

deg1 = degeneracy of Schrödinger equation to be solved

Output of effective mass tensor

**effective-mass-tensor****example**

```
yes or no
```

**default**

```
no
```

Flag whether to write out the effective mass tensor  $(1/m)_{ij}$  for each Schrödinger equation to be solved. Output are 6 components of the symmetric 3x3 matrix for each grid point. The file is called `*mass_tensor*.dat`.

Example for 1D output: `cb3_mass_tensor_qc1_sg3_deg1.dat`

```
position[nm]  (1/m)_xx  (1/m)_yy  (1/m)_zz  (1/m)_xy  (1/m)_xz
↪ (1/m)_yz
...
0.350750E+002  0.769231E+000  0.434783E+001  0.434783E+001  0.000000E+000  0.
↪ 0.000000E+000  0.000000E+000
...
1/1.3 = 0.769  1/0.23 = 4.434  1/0.23 = 4.34
```

In this example the mass tensor is diagonal and  $(1/m)_{xx}$  contains  $1/(\text{longitudinal mass } m_l)$  and  $(1/m)_{yy} = (1/m)_{zz}$  contains  $1/(\text{transverse mass } m_t)$  of the X valley of GaAs.

The effective masses are specified in the database.

```
$binary-zb-default
binary-type = GaAs-zb-default
conduction-band-masses = 0.067 0.067 0.067 ! Gamma (isotropic)
                        1.9 0.0754 0.0754 ! L (ml mt mt)
                        1.3 0.23 0.23 ! X (ml mt mt)
```

More information on [effective masses](#).

Note that the labels x, y and z of the mass tensor output are defined with respect to the *simulation* coordinate system (and not *crystal* coordinate system).

### complex-wave-functions

#### example

yes or no

#### default

no

Flag whether to print out the wave functions  $\psi$  (amplitudes) including real and imaginary parts in addition to the output of the probability densities  $\psi^2$ . The amplitudes of a single-band Hamiltonian are typically real and the imaginary part is zero. This does not hold for nonzero magnetic fields or nonzero superlattice vectors. Depending on the algorithm inside the code (e.g. storage of the Hamiltonian in a real or complex array, or usage of a real or complex eigenvalue solver), the output contains only the real part, or the real part and the imaginary part (which is zero in most cases). In the case of a complex eigenvalue solver, the imaginary part might be nonzero even for zero magnetic field or no superlattice vector. If this is the case, the wave functions could be converted into a real basis. (It would have been better to call this specifier `amplitudes` rather than `complex-wave-functions`.)

### eigenvalues-only

#### example

yes or no

#### default

no

Sometimes one is only interested in plotting out the eigenvalues but not the eigenfunctions. Nevertheless, internally in the program the eigenfunctions are used, e.g. for calculating the density.

## 5.3.44 \$output-kp-data

The documentation for this keyword is available here: [https://www.nextnano.com/nextnano3/input\\_parser/keywords/output-kp-data.htm](https://www.nextnano.com/nextnano3/input_parser/keywords/output-kp-data.htm)

### 5.3.45 \$output-densities

The documentation for this keyword is available here: [https://www.nextnano.com/nextnano3/input\\_parser/keywords/output-densities.htm](https://www.nextnano.com/nextnano3/input_parser/keywords/output-densities.htm)

### 5.3.46 \$output-current-data

The documentation for this keyword is available here: [https://www.nextnano.com/nextnano3/input\\_parser/keywords/output-current-data.htm](https://www.nextnano.com/nextnano3/input_parser/keywords/output-current-data.htm)

### 5.3.47 \$output-file-format

The documentation for this keyword is available here: [https://www.nextnano.com/nextnano3/input\\_parser/keywords/output-file-format.htm](https://www.nextnano.com/nextnano3/input_parser/keywords/output-file-format.htm)

### 5.3.48 \$output-section

The documentation for this keyword is available here: [https://www.nextnano.com/nextnano3/input\\_parser/keywords/output-section.htm](https://www.nextnano.com/nextnano3/input_parser/keywords/output-section.htm)

### 5.3.49 \$output-material

Writes material parameters on grid coordinates to output files.

|                                     |           |                      |
|-------------------------------------|-----------|----------------------|
| <b>\$output-material</b>            |           |                      |
| destination-directory               | character | optional<br>required |
| number-of-conduction-bands          | character | optional             |
| conduction-band-energies            | character | optional             |
| conduction-band-masses              | character | optional             |
| conduction-band-degeneracies        | character | optional             |
| conduction-band-nonparabolicities   | character | optional             |
| number-of-valence-bands             | character | optional             |
| valence-band-energies               | character | optional             |
| valence-band-masses                 | character | optional             |
| valence-band-degeneracies           | character | optional             |
| valence-band-nonparabolicities      | character | optional             |
| static-dielectric-constants         | character | optional             |
| optical-dielectric-constants        | character | optional             |
| lattice-constants                   | character | optional             |
| doping-concentration                | character | optional             |
| pressure                            | character | optional             |
| elastic-constants                   | character | optional             |
| absolute-deformation-potentials-cbs | character | optional             |
| absolute-deformation-potential-vb   | character | optional             |
| uni-ax-cb-deformation-potentials    | character | optional             |
| uni-ax-vb-deformation-potentials    | character | optional             |
| spontaneous-polarization            | character | optional             |
| piezoelectric-polarization          | character | optional             |
| Luttinger-parameters                | character | optional             |
| 6x6kp-parameters                    | character | optional             |
| 8x8kp-parameters                    | character | optional             |
| L0-phonon-energy                    | character | optional             |

(continues on next page)

(continued from previous page)

|                       |                       |
|-----------------------|-----------------------|
| grid-position         | double_array optional |
| \$end_output-material | optional              |

**destination-directory****type**

character

**example**

material\_parameters/

Name of directory to which the files should be written. Directory name has to include the slash (for DOS and / for Linux)

**number-of-conduction-bands****type**

character

**example**

num-cbands

Name of file for storing the number of conduction bands at each grid point.

**conduction-band-energies****type**

character

**example**

cb-energies

Name of file for storing the conduction band energies  $E_c^\Gamma$ ,  $E_c^L$ ,  $E_c^X$  at each grid point.

**conduction-band-masses****type**

character

**example**

cb-masses

Name of file for storing the conduction band masses  $m^\Gamma$ ,  $m^L$ ,  $m^X$  at each grid point.

**conduction-band-degeneracies****type**

character

**example**

cb-degeneracy

Name of file for storing the degeneracies of the conduction bands at  $\Gamma$ ,  $L$ ,  $X$  for each grid point.

**conduction-band-nonparabolicities****type**

character

**example**

cb-nonpar

Name of file for storing the non-parabolicities of the conduction bands  $\Gamma$ ,  $L$ ,  $X$  for each grid point.

**number-of-valence-bands****type**

character

**example**

num-vbands

Name of file for storing the number of valence bands at each grid point.

**valence-band-energies****type**

character

**example**

vb-energy

Name of file for storing the average of the three valence band energies  $E_{v,av}$  at each grid point.

**valence-band-masses****type**

character

**example**

vb-masses

Name of file for storing the valence band masses  $m_{hh}$ ,  $m_{lh}$ ,  $m_{so}$  at each grid point.

**valence-band-degeneracies****type**

character

**example**

vb-degeneracy

Name of file for storing the valence band degeneracies of the hh, lh, so states at each grid point.

**valence-band-nonparabolicities****type**

character

**example**

vb-nonpar

Name of file for storing the valence band non-parabolicities of the hh, lh, so states at each grid point.

**static-dielectric-constants****type**

character

**example**

st-dielc

Name of file for storing the static dielectric constant  $\epsilon(0)$  at each grid point.

**optical-dielectric-constants****type**

character

**example**

op-dielc

Name of file for storing the optical dielectric constant  $\epsilon(\infty)$  at each grid point.

**doping-concentration**



**type**

character

**example**

doping\_concentration

Name of file for storing the doping concentration in units of  $1 \cdot 10^{18} \text{ 1/cm}^3$  at each grid point.

**lattice-constants****type**

character

**example**

lattice-constants

Name of file for storing the lattice-constant  $a$  (zincblende) or  $a, c$  (wurtzite) at each grid point.

**pressure****type**

character

**example**

pressure

Name of file for storing the pressure in units of [GPa] at each grid point.

**elastic-constant****type**

character

**example**

elastic-constants

Name of file for storing the elastic constants  $c_{11}, c_{12}, c_{44}$  (zincblende) or  $c_{11}, c_{12}, c_{13}, c_{33}, c_{44}$  (wurtzite) at each grid point.

**absolute-deformation-potentials-cbs****type**

character

**example**

abs\_def\_cb

Name of file for storing the absolute deformation potentials for the conduction band at  $\Gamma, L, X$  at each grid point.

**absolute-deformation-potential-vb****type**

character

**example**

abs\_def\_vb

Name of file for storing the absolute deformation potential for the valence band at each grid point.

**uni-ax-cb-deformation-potentials****type**

character

**example**

uni-ax\_cb\_def

Name of file for storing the uniaxial deformation potential conduction band at  $\Gamma, L, X$  at each grid point.

**uni-ax-vb-deformation-potentials**

**type**  
character

**example**  
uniax\_vb\_def

Name of file for storing the uniaxial deformation potentials  $b$ ,  $d$  for the valence band at each grid point.

#### **spontaneous-polarization**

**type**  
character

**example**  
pyro\_polarization

Name of file for storing the pyroelectric polarization constant at each grid point.

#### **piezoelectric-polarization**

**type**  
character

**example**  
piezo\_polarization

Name of file for storing the piezoelectric polarization constants  $e_{14}$  (zincblende) or  $e_{33}$ ,  $e_{31}$ ,  $e_{15}$  (wurtzite) in units of [C/cm<sup>2</sup>] at each grid point.

#### **Luttinger-parameters**

**type**  
character

**example**  
Luttinger

Name of file for storing the Luttinger parameters  $\gamma_1$ ,  $\gamma_2$ ,  $\gamma_3$  at each grid point.

#### **6x6kp-parameters**

**type**  
character

**example**  
6x6kp

Name of file for storing the 6-band **k.p** parameters  $L$ ,  $M$ ,  $N$ ,  $\Delta_{so}$  at each grid point.

#### **8x8kp-parameters**

**type**  
character

**example**  
8x8kp

Name of file for storing the 8-band **k.p** parameters  $L'$ ,  $M'$ ,  $N'$ ,  $B$ ,  $P$ ,  $S$ ,  $\Delta_{so}$  at each grid point.

#### **LO-phonon-energy**

**type**  
character

**example**  
LO\_phonon\_energy

Name of file for storing the LO-phonon energy  $E_{LO}$  in units of eV at each grid point.

#### **grid-position**

**type**

double\_array

**example**

10d0 (1D)

10d0 20d0 (2D)

10d0 20d0 30d0 (3D)

Prints out the material parameters for the binary (or ternary) material at this position. This feature can be used to calculate the material parameters of a ternary, and print it to a file. This data format can then be modified and read in from the database or input file (“copy and paste”). Example: MaterialParametersGridPointAl(x)Ga(1-x)N.dat

```
$binary-wz-default
!
binary-type                = Al(x)Ga(1-x)N, x=0.400, 1-x=0.600-wz-
default
conduction-bands           = 3
!
conduction-band-masses     = 0.208000D+00 0.208000D+00 0.
252000D+00 ! [m0]          0.946800D+00 0.286800D+00 0.
493200D+00 ! [m0]          0.143400D+01 0.300000D+00 0.
376000D+00 ! [m0]
conduction-band-degeneracies = 2 8 6
!
...
```

**Example**

```
$output-material
destination-directory      = material_parameters/
number-of-conduction-bands = cb-nums.dat
conduction-band-energies   = cb-energies.dat
conduction-band-masses     = cb-masses.dat
conduction-band-nonparabolicities = cb-nonpara.dat
conduction-band-degeneracies = cb-deg.dat
number-of-valence-bands    = vb-nums.dat
valence-band-energies      = vb-energy.dat
valence-band-masses        = vb-masses.dat
valence-band-nonparabolicities = vb-nonpara.dat
valence-band-degeneracies  = vb-deg.dat
static-dielectric-constants = static-dielectric-constants.dat
optical-dielectric-constants = optical-dielectric-constants.dat
lattice-constants          = lattice-constants.dat
pressure                   = pressure.dat
elastic-constants          = elastic-constants.dat
absolute-deformation-potentials-cbs = cb-abs-defpots.dat
absolute-deformation-potential-vb = vb-abs-defpot.dat
uniax-cb-deformation-potentials = cb-uniax-defpots.dat
uniax-vb-deformation-potentials = vb-uniax-defpots.dat
Luttinger-parameters      = Luttinger-parameters.dat
6x6kp-parameters          = 6x6kp-parameters.dat
8x8kp-parameters          = 8x8kp-parametersv
LO-phonon-energy           = LO_phonon_energy.dat
piezoelectric-polarization = piezo-constants.dat
```

(continues on next page)

(continued from previous page)

|                          |                            |
|--------------------------|----------------------------|
| spontaneous-polarization | = pyro-constants.dat       |
| doping-concentration     | = doping-concentration.dat |
| grid-position            | = 5d0                      |
| \$end_output-material    |                            |

### 5.3.50 \$doping-function

The documentation for this keyword is available here: [https://www.nextnano.com/nextnano3/input\\_parser/keywords/doping-function.htm](https://www.nextnano.com/nextnano3/input_parser/keywords/doping-function.htm)

### 5.3.51 \$interface-states

The documentation for this keyword is available here: [https://www.nextnano.com/nextnano3/input\\_parser/keywords/interface-states.htm](https://www.nextnano.com/nextnano3/input_parser/keywords/interface-states.htm)

### 5.3.52 \$global-settings

The documentation for this keyword is available here: [https://www.nextnano.com/nextnano3/input\\_parser/keywords/global-settings.htm](https://www.nextnano.com/nextnano3/input_parser/keywords/global-settings.htm)

### 5.3.53 \$simulation-flow-control

The documentation for this keyword is available here: [https://www.nextnano.com/nextnano3/input\\_parser/keywords/simulation-flow-control.htm](https://www.nextnano.com/nextnano3/input_parser/keywords/simulation-flow-control.htm)

### 5.3.54 \$grid-specification

The documentation for this keyword is available here: [https://www.nextnano.com/nextnano3/input\\_parser/keywords/grid-specification.htm](https://www.nextnano.com/nextnano3/input_parser/keywords/grid-specification.htm)

### 5.3.55 \$import-data-on-material-grid

The documentation for this keyword is available here: [https://www.nextnano.com/nextnano3/input\\_parser/keywords/import-data-on-material-grid.htm](https://www.nextnano.com/nextnano3/input_parser/keywords/import-data-on-material-grid.htm)

### 5.3.56 \$binary-zb-default

The documentation for this keyword is available here: [https://www.nextnano.com/nextnano3/input\\_parser/keywords/binary-zb-default.htm](https://www.nextnano.com/nextnano3/input_parser/keywords/binary-zb-default.htm)

### 5.3.57 \$binary-wz-default

The documentation for this keyword is available here: [https://www.nextnano.com/nextnano3/input\\_parser/keywords/binary-wz-default.htm](https://www.nextnano.com/nextnano3/input_parser/keywords/binary-wz-default.htm)

### 5.3.58 \$ternary-zb-default

The documentation for this keyword is available here: [https://www.nextnano.com/nextnano3/input\\_parser/keywords/ternary-zb-default.htm](https://www.nextnano.com/nextnano3/input_parser/keywords/ternary-zb-default.htm)

### 5.3.59 \$ternary-wz-default

The documentation for this keyword is available here: [https://www.nextnano.com/nextnano3/input\\_parser/keywords/ternary-wz-default.htm](https://www.nextnano.com/nextnano3/input_parser/keywords/ternary-wz-default.htm)

### 5.3.60 \$alloy-function

The documentation for this keyword is available here: [https://www.nextnano.com/nextnano3/input\\_parser/keywords/alloy-function.htm](https://www.nextnano.com/nextnano3/input_parser/keywords/alloy-function.htm)

### 5.3.61 \$numeric-control

The documentation for this keyword is available here: [https://www.nextnano.com/nextnano3/input\\_parser/keywords/numeric-control.htm](https://www.nextnano.com/nextnano3/input_parser/keywords/numeric-control.htm)

### 5.3.62 \$CBR-current

The documentation for this keyword is available here: [https://www.nextnano.com/nextnano3/input\\_parser/keywords/CBR-current.htm](https://www.nextnano.com/nextnano3/input_parser/keywords/CBR-current.htm)

### 5.3.63 \$quantum-regions

The documentation for this keyword is available here: [https://www.nextnano.com/nextnano3/input\\_parser/keywords/quantum-regions.htm](https://www.nextnano.com/nextnano3/input_parser/keywords/quantum-regions.htm)

### 5.3.64 \$quantum-model-electrons

The documentation for this keyword is available here: [https://www.nextnano.com/nextnano3/input\\_parser/keywords/quantum-model-electrons.htm](https://www.nextnano.com/nextnano3/input_parser/keywords/quantum-model-electrons.htm)

### 5.3.65 \$quantum-model-holes

The documentation for this keyword is available here: [https://www.nextnano.com/nextnano3/input\\_parser/keywords/quantum-model-holes.htm](https://www.nextnano.com/nextnano3/input_parser/keywords/quantum-model-holes.htm)

### 5.3.66 \$voltage-sweep

The documentation for this keyword is available here: [https://www.nextnano.com/nextnano3/input\\_parser/keywords/voltage-sweep.htm](https://www.nextnano.com/nextnano3/input_parser/keywords/voltage-sweep.htm)

### 5.3.67 \$electric-field

The documentation for this keyword is available here: [https://www.nextnano.com/nextnano3/input\\_parser/keywords/electric-field.htm](https://www.nextnano.com/nextnano3/input_parser/keywords/electric-field.htm)

### 5.3.68 \$simple-drift-models

The documentation for this keyword is available here: [https://www.nextnano.com/nextnano3/input\\_parser/keywords/simple-drift-models.htm](https://www.nextnano.com/nextnano3/input_parser/keywords/simple-drift-models.htm)

### 5.3.69 \$NEGF

The documentation for this keyword is available here: [https://www.nextnano.com/nextnano3/input\\_parser/keywords/NEGF.htm](https://www.nextnano.com/nextnano3/input_parser/keywords/NEGF.htm)

### 5.3.70 \$electrolyte

The documentation for this keyword is available here: [https://www.nextnano.com/nextnano3/input\\_parser/keywords/electrolyte.htm](https://www.nextnano.com/nextnano3/input_parser/keywords/electrolyte.htm)

### 5.3.71 \$electrolyte-ion-content

The documentation for this keyword is available here: [https://www.nextnano.com/nextnano3/input\\_parser/keywords/electrolyte-ion-content.htm](https://www.nextnano.com/nextnano3/input_parser/keywords/electrolyte-ion-content.htm)

Keywords: This section contains the definition of the allowed syntax in your input file.

The old documentation of the keywords for *nextnano<sup>3</sup>* is available here: [https://www.nextnano.com/nextnano3/input\\_parser/keywords/keywords.htm](https://www.nextnano.com/nextnano3/input_parser/keywords/keywords.htm)

## COMMAND LINE

Command line usage:

```
$ nextnano3_Intel_64bit.exe --inputfile 1D_HEMT.in --database database_nn3.in --  
↪ license License_nnp.lic --outputdirectory 1D_HEMT/
```

Currently, *nextnano*<sup>3</sup> provides the following options:

**-i input\_file, --inputfile input\_file** Name of input file to be read in.

Example:

- `--inputfile 1D_GaAs_HEMT.in`
- `--inputfile "1D GaAs HEMT.in"`

Blanks in file names are supported if using quotation marks " ".

Default value: Entry in `keywords.val`.

**-d database\_file, --database database\_file** Name of material parameter database file to be read in.

Example:

- `--database database_modified_my_Si_parameters.in`
- `--database "my database_nn3.in"`

Blanks in file names are supported if using quotation marks " ".

Default value: Entry in file `database_nn3_keywords.val`, i.e. `database_nn3.in`.

**-l license\_file, --license license\_file** Name of license file to be read in.

Example:

- `--license license.dat`
- `--license License_nnp.lic`
- `--license H:\Documents\nextnano\license\license.dat`
- `--license "H:\My Documents\nextnano\license\license.dat"`

Default value: `License_nnp.lic`

**-o output\_directory, --outputdirectory output\_directory** Directory name where to output results.

Example:

- `--outputdirectory my_output_foldername`
- `--outputdirectory my_output_foldername/`

- `--outputdirectory "my output foldername/"`
- `--outputdirectory "my output\my folder\"`
- `--outputdirectory "<name_of_input_file>"`
- `--outputdirectory "<name_of_input_file>\output"`
- `--outputdirectory "E:\My Documents\nextnano_output\  
<name_of_input_file>"`

In the last three examples the string `<name_of_input_file>` is provided and *not* the name of the input file. This special string `<name_of_input_file>` will be replaced automatically with the actual name of the input file.

Blanks in folder names are supported if using quotation marks " ".

**-t i, --threads i**

Number of parallel threads (*i* = integer) to benefit from parallelization on multi-core CPUs.

Example:

- `--threads 1` uses 1 thread (default)
- `--threads 4` uses 4 threads (only works with an executable that had been compiled with OpenMP option)
- `--threads 0` automatic decision, i.e. use value specified in input file, or use default value if not specified in input file

This setting only affects parallelization of e.g.  $k_{||}$  vectors or energy grid points (NEGF). Parallelization of calls to MKL are still parallelized automatically.

**--debuglevel i**

Generate additional debug information (*i* = integer).

Example:

- `--debuglevel 3`

The integer number has the following meaning:

- `0`: no debug information at all (default)
- `1`: modest debug information
- `2`: more debug information
- `3`: even more debug information
- *n*: any other integer number is also possible, some have specific meanings (for developers only)
- `> 1000`: Here, the files `keywords_nn3.xml` and `database_nn3_keywords_nn3.xml` are created, which are used by *nextnanomat* for its auto completion feature.
- `< 0`: automatic decision, i.e. use value specified in input file, or use default value if not specified in input file

**--log**

Generate a \*.log file of screen output (standard output).

The file will be written to: `<outputdirectory>/<inputfilename>_log.log`

**--cancel i**

Automatically cancel simulation after *i* minutes (*kill*, i.e. program stops immediately).

If value is `<= 0`, the simulation is not canceled.

Example:

- `--cancel 10`



|                                  |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
|----------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>--softkill i</b>              | <p>Automatically cancel simulation after <i>i</i> minutes (<i>soft kill</i>, i.e. program exits iteration cycle and writes output).</p> <p>If value is <math>\leq 0</math>, the simulation is not canceled.</p> <p>Example:</p> <ul style="list-style-type: none"> <li>• <code>--softkill 10</code></li> </ul> <p>Alternative option: If the user saves a file called <code>SOFT_KILL</code> (without file extension) into the output folder of the currently running simulation, a <i>soft kill</i> will be performed.</p>                                                                                                                                                                                                                                                                                                                  |
| <b>--system operating_system</b> | <p>Flag indicating on which operating system the executable is executed.</p> <p>Example:</p> <ul style="list-style-type: none"> <li>• <code>--system default</code> (default)</li> <li>• <code>--system windows</code></li> <li>• <code>--system linux</code></li> </ul> <p>Available options are <code>default</code>, <code>windows</code>, <code>linux</code>, <code>unix</code> and <code>mac</code>.</p> <p>When <code>default</code> is specified, the corresponding setting of your executable is applied.</p> <p>Typically, it is not required to specify this flag as the executables had been compiled for the respective operating systems (and thus this setting has already been applied automatically).</p> <p>Currently, the internal settings for <code>linux</code>, <code>unix</code>, <code>mac</code> are identical.</p> |
| <b>--condor</b>                  | Flag that is necessary if job is submitted to HTCondor.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
| <b>--postprocess</b>             | Enable postprocessing from input file using <code>!DATA !ENDDATA</code> keywords.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
| <b>-p, --parse</b>               | Parse input file and quit.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
| <b>--parse-keywords</b>          | Parse <code>keywords.val</code> file. By default, the content of the file <code>keywords.val</code> is read in from within the source code.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
| <b>--parse-database</b>          | Parse <code>database_nn3_keywords.val</code> file. By default, the content of the file <code>database_nn3_keywords.val</code> is read in from within the source code.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| <b>-v, --version</b>             | Show version number only.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
| <b>-h, --help</b>                | Display available command line options.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |

#### Note

For versions before 2021, the double-hyphen flags `--` are not supported. Please use instead the appropriate single-hyphen flags `-`:

- `-inputfile`
- `-database`
- `-license`
- `-outputdirectory`
- `-help` (Displays allowed options.)



## RELEASE NOTES

### 7.1 2.7.2.s (2025-12-09)

- New specifier ‘output-lifetime’ (\$CBR-current): Calculation of lifetime based on the calculated DOS in the device (experimental feature)
- default file extension for input files is now “.nn3” (previous default file extension was “.in”)

### 7.2 2.6.12.s (2025-05-25)

- bugfix for exciton/biexciton calculation in 3D

### 7.3 2.6.11.s (2025-04-01)

- minor bugfixes

### 7.4 2.6.9.b (2024-12-18)

- bugfix for labeling acceptors in output files.
- Altermatt model of incomplete ionization.
- removed information on boundary conditions from the output filename.
- Added script postprocessing method.
- Increased allowed string length for `potential-from-function`

---

### 7.5 2.6.5.b (2024-08-27)

- filename energy spectrum fixed
- bugfix for excitons
- other minor bugfixes

---

### 7.6 2.6.4.b (2024-08-05)

- new output of doping ionization ratio
  - minor bugfix for  $\mathbf{k} \cdot \mathbf{p}$  solver
  - Altermatt incomplete ionization model implemented
-

## 7.7 2.6.1.b (2024-04-02)

- minor changes
- 

## 7.8 2.6.0.b (2024-02-01)

- multiple improvements of messages in log file
  - scaling bugfix for lifetimes
  - bugfix for drift velocity
  - parametrization bugfix for 30-band  $\mathbf{k} \cdot \mathbf{p}$  model
- 

## 7.9 2.5.4.b (2023-07-27)

### syntax

- `MIN()` and `MAX()` functions introduced
- `!WHEN` statement introduced for conditional lines
- `!TEXT` and `!ENDTEXT` statements introduced for multi-line comments
- `!DATA` statement got introduced for post-processor

### output

- output of material parameters in the *nextnano++* database style

### models

- excitonic absorption
  - tunneling current for CBR
- 

## 7.10 2.3.8.b (2022-09-29)

- Bowing parameters of band edges has been updated in the default database.
  - 30-band  $\mathbf{k} \cdot \mathbf{p}$  model for bulk crystals without strain has been implemented.
- 

## 7.11 2.3.7.b (2022-08-03)

- band-anti-crossing (BAC) model for nitrides as an experimental feature
-

## 7.12 EARLIER

- Bulk band structure with 14-band and 30-band k.p
- Improvements for CBR
- New 2D CBR input files (QPC)
- Improvement of 2DEG mobility input files (e.g. GaN)
- Synonyms in material database (e.g.  $\text{Al}(x)\text{In}(x)\text{As}$  and  $\text{In}(x)\text{Al}(1-x)\text{As}$ )
- Input files for tutorial: “Intersubband absorption in infinite QW”
- Input files for intersubband absorption in Ge/SiGe QWs
- Added more tutorial input files to samples folder
- Improvements for intersubband absorption (single-band, simple k.p model, k.p)
- Grid lines can be independent of region object boundaries (has always been the case for nn++)
- Added XML support to input files
- New region objects: circle/sphere
- New region objects: triangle, polygonal\_prism, regular\_prism, hexagonal\_prism, polygonal\_pyramid, regular\_pyramid, hexagonal\_pyramid
- 2nd order piezo constants and 2nd order piezo tutorial
- Bulk band structure with pseudopotential method
- Bulk band structure with tight-binding method (improvements)
- Piezo/pyro tutorial for arbitrary (hkil) in wurtzite



- *Where to find simulation LOG file*
- *What is the difference between `nextnano3` and `nextnano++`?*
- *Can I convert `nextnano3` input files into `nextnano++` input files?*
- *How can I speed up my calculations with respect to CPU time?*
- *Can I take advantage of parallelization of the `nextnano` software on multi-core CPUs?*
- *What boundary conditions are available?*
- *What are quasi-Fermi levels*
- *I do not understand the  $\mathbf{k} \cdot \mathbf{p}$  parameters*
- *Can I add new materials to the database?*
- *Should I use averaged outputs and boxes?*

---

## 8.1 Where to find simulation LOG file

The simulation log file is a file with the same name as the input file and the extension `*.log`. It is located in the output folder of the simulation. It is necessary for the support team to debug issues with the simulation, thus it should always be attached to a support request. Please also include additional debug information in case the log file shall be used for support, refer to *nextnanomat*

## 8.2 What is the difference between `nextnano3` and `nextnano++`?

The short answer is: Use `nextnano++`.

This is the tool where we put on most effort in improving it by adding new features. The `nextnano3` tool is mainly distributed for historical reasons.

The `nextnano3` tool is written in Fortran. It has been developed at the Walter Schottky Institute from 1999 to 2010. The `nextnano++` tool is written in C++. It has been developed at the Walter Schottky Institute from 2004 to 2008. The tools have been written by different people that were working at the Walter Schottky Institute of the Technische Universität München in the Theoretical Semiconductor Physics group of Prof. Peter Vogl.

**Essentially, both tools cover the same physics and methods, namely**

- the strain equation
- the Poisson equation
- the Schrödinger equation
- the drift-diffusion current equation.

Additionally, *nextnano*<sup>3</sup> includes

- solar cells
- electrolytes
- graphene
- tight-binding (for bulk and one-dimensional superlattices)
- the self-consistent CBR method (1D, 2D and 3D)
- the NEGF method (1D only) which is particularly suited for quantum cascade lasers.

In contrast to *nextnano++*, *nextnano*<sup>3</sup> is not able to

- treat the magnetic field within the  $\mathbf{k} \cdot \mathbf{p}$  approach
- calculate the g tensor
- include quaternary materials.

The *nextnano++* tool is much faster for drift-diffusion calculations and for 2D/3D simulations. For instance, if you work on LEDs, MOSFETs or Quantum Dots, *nextnano++* is much better suited.

There are some applications where it is irrelevant which tool to use. In this case we recommend to use both. This has the advantage that the results of one tool can be compared to the results of the other one in order to gain more confidence in them. For some applications, one tool should be preferred. Please contact <support[at]nextnano.com> to find out which tool to choose for your particular application.

*nextnano*<sup>3</sup> syntax

- “\$” character for the keywords: \$regions ... \$end\_regions
- “%” character for the variables: %QuantumWellWidth = 5.0
- “!” character for comments: ! This is a comment. (# is supported by both *nextnano* and *nextnano*<sup>3</sup> as a comment sign.)

*nextnano++* syntax

- “{ }” brackets for the keywords: structure{ ... }
- “\$” character for the variables: \$QuantumWellWidth = 5.0`
- “#” character for comments: # This is a comment. (# is also supported by *nextnano*<sup>3</sup>.)

## 8.3 Can I convert nextnano<sup>3</sup> input files into nextnano++ input files?

Within *nextnanomat*, there is an experimental feature to convert a *nextnano*<sup>3</sup> input file to a *nextnano++* input file. How to use this automatic conversion:

In the menu select Tools ==> Convert nextnano<sup>3</sup> input file to nextnano++

When you use the automatic conversion of *nextnano*<sup>3</sup> input file into *nextnano++*, you will find that it probably does not work completely.

If you save and run the *nextnano++* input file that has been converted and that has the suffix `_nnp.in`, very likely some errors appear indicating which line(s) to change. Then some manual adjustments are needed, but the rough structure should help a lot for the conversion.

## 8.4 How can I speed up my calculations with respect to CPU time?

The most obvious way is to reduce the **number of grid points** you are using. For instance, for the following p-n junction simulation, a grid spacing of **1 nm** was used (gray lines in Figure 8.4.1 ). If one is using a coarse grid of only **10 nm**, the calculated values (squares in Figure 8.4.1) agree very well with the calculated values of the thin lines.



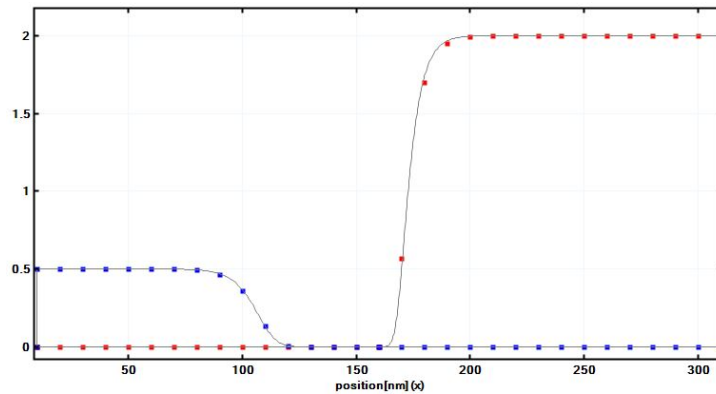


Figure 8.4.1: Hole (blue) and electron (red) densities of the p-n junction in units of  $10^{18} \text{cm}^{-3}$ . The gray lines are from simulations using a 1 nm grid spacing. The squares are from a simulation that uses only a 10 nm grid resolution. Note that the center coordinate of this plot is  $x=160$  nm. The depletion width for the holes is around  $\text{wp:math:approx } 50 \text{ nm}$ , for the electrons it is  $\text{wn:math:approx } 10 \text{ nm}$  which is of the order of the grid spacing. Even in this case, the calculated electron density is reasonably accurate.

The difference in CPU time comes from the fact that for the 10 nm resolution the dimension of the matrix that is used for discretizing the Poisson equation is 30, while in the case for the 1 nm grid spacing it has the dimension 300. The proper choice of an optimal grid spacing is very relevant for 2D and 3D simulations, as can be seen in the following.

#### 1D simulation (length of sample: $x = 300$ nm)

- 1 nm grid spacing: dimension of Poisson matrix:  $N = 300$
- 10 nm grid spacing: dimension of Poisson matrix:  $N = 30$

#### 2D simulation (length of sample: $x = 300$ nm, $y = 300$ nm)

- 1 nm grid spacing: dimension of Poisson matrix:  $N = 300 \cdot 300 = 90,000$
- 10 nm grid spacing: dimension of Poisson matrix:  $N = 30 \cdot 30 = 900$

#### 3D simulation (length of sample: $x = 300$ nm, $y = 300$ nm, $z = 300$ nm)

- 1 nm grid spacing: dimension of Poisson matrix:  $N = 300 \cdot 300 \cdot 300 = 27,000,000$
- 10 nm grid spacing: dimension of Poisson matrix:  $N = 30 \cdot 30 \cdot 30 = 27,000$

If a quantum mechanical simulation is performed, the numerical effort of eigenvalue solvers increases with the number of grid points  $N$  with order  $O(N^2)$ .

## 8.5 Can I take advantage of parallelization of the nextnano software on multi-core CPUs?

### The short answer is:

Some numerical routines are parallelized which is done automatically. These are the numerical routines, e.g. for calculating the eigenvalues with a LAPACK solver (which itself uses BLAS).

### The long answer is:

The nextnano software includes the Intel® Math Kernel Library (MKL). MKL includes the BLAS and LAPACK library routines for numerical operations. The MKL dynamically changes the number of threads.

- nextnano++ - uses MKL (parallel version). The executables that are compiled with the Intel and Microsoft compilers use MKL (parallel version). The executable that is compiled with the GNU compiler (gcc/gfortran) uses the nonparallelized version of the BLAS and LAPACK source codes available from [netlib webpage](http://netlib.org).

- *nextnano*<sup>3</sup> - uses MKL (parallel version) The executables that are compiled with the Intel and NAG (64-bit) compilers use MKL (parallel version). The executables that are compiled with the GNU compiler (gfortran) and NAG (32-bit) use the nonparallelized version of the BLAS and LAPACK source codes available from [netlib webpage](#). There is a *nextnano*<sup>3</sup> executable available that uses OpenMP parallelization for
  - CBR (parallelization with respect to energy grid)
  - NEGF (parallelization with respect to energy grid and further loops) `number-of-MKL-threads = 8`
  - Calculation of eigenstates for each  $k_{||}$  (1D and 2D simulations)
  - Matrix-vector products of numerical routines Note: Not all operations are thread-safe, e.g. one cannot combine  $k_{||}$  parallelization with the ARPACK eigenvalue solver. Only for this executable, the flag `number-of-parallel-threads = 4` has an effect. The NEGF keyword also supports `number-of-MKL-threads = 4` (0 means *dynamic* with is recommended) and `MKL-set-dynamic = yes / no`.
- *nextnano.NEGF* - uses MKL (parallel version)
- *nextnano.MSB* - uses MKL (parallel version)

The NEGF algorithms (*nextnano.NEGF*, *nextnano.MSB*, CBR) include matrix-matrix operations which are well parallelized within the BLAS routines.

If e.g. 4 nextnano simulations are running in parallel on a quad-core CPU, i.e. 4 nextnano executables are running simultaneously and each of them is using calls to the parallelized MKL library simultaneously, the total performance might be slower compared to running these simulations one after the other. In this case using a nextnano executable compiled with the serial version of the Intel MKL could be faster.

In fact, it strongly depends on your nextnano application (e.g. 1D vs. 3D simulation, LAPACK vs. ARPACK eigenvalue solver, ...) if you benefit from parallelization or not. In general, the best parallelization can be obtained if you run several nextnano simulations in parallel. For instance, you could do parameter sweeps (e.g. sweep over quantum well width) using *nextnanomat*'s *Template* feature, i.e. if you run 4 simulations simultaneously on a quad-core CPU, e.g. for 4 different quantum well widths.

## 8.6 What boundary conditions are available?

There are three different boundary conditions that we use:

- *periodic*:  $\psi(x=0) = \psi(x=L)$
- *Dirichlet*:  $\psi(x=0) = \psi(x=L) = 0$ , and
- *Neumann*:  $\frac{d\psi}{dx} = \text{const}$  at the left ( $x=0$ ) and right ( $x=L$ ) boundary. Typically,  $\text{const} = 0$ .

By specifying both Dirichlet and Neumann boundary conditions, the system would be over-determined.

## 8.7 What are quasi-Fermi levels

So-called *quasi-Fermi levels* which are different for electrons  $E_{F,n}$  and holes  $E_{F,p}$  are used to describe non-equilibrium carrier concentrations. In equilibrium the quasi-Fermi levels are constant and have the same value for both electrons and holes,  $E_{F,n} = E_{F,p} = 0$  eV. The electron current is proportional to the electron mobility  $\mu_n(x)$ , carrier density  $n(x)$  and the gradient of the quasi-Fermi level of the carriers,  $\nabla E_{F,n}(x)$ , and analogously for the holes.

## 8.8 I do not understand the $k \cdot p$ parameters

In the literature, there are two different notations used:

- Dresselhaus–Kip–Kittel (DKK):  $L, M, N^+, N^-$  (zinc blende);  $L_1, L_2, M_1, M_2, M_3, N_1^+, N_1^-, N_2^+, N_2^-$  (wurtzite)

- Luttinger parameters:  $\gamma_1, \gamma_2, \gamma_3, \kappa$  (zinc blende); Rashba–Sheka–Pikus (RSP) parameters  $A_1, A_2, A_3, A_4, A_5, A_6, A_7$  (wurtzite)

They are equivalent and can be converted into each other.

Some authors only use 3 parameters  $L, M, N$  (or  $\gamma_1, \gamma_2, \gamma_3$ ) which is fine for bulk semiconductors without magnetic field but not for heterostructures because the latter require 4 parameters, i.e.  $N^+, N^-$  (instead of  $N$  only) or  $\kappa$ . If these parameters are not known, they can be approximated.

There are different  $\mathbf{k} \cdot \mathbf{p}$  parameters for

- 6-band  $\mathbf{k} \cdot \mathbf{p}$  and
- 8-band  $\mathbf{k} \cdot \mathbf{p}$ .

The 8-band  $\mathbf{k} \cdot \mathbf{p}$  parameters can be calculated from the 6-band parameters taking into account the temperature dependent band gap  $E_{\text{gap}}$  and the Kane parameter  $E_P$  (zinc blende). For wurtzite the parameters are  $E_{\text{gap}}$  and the Kane parameters  $E_{P1}, E_{P2}$ .

The 8-band Hamiltonian also needs the conduction band mass parameter  $S$  (zinc blende) or  $S_1, S_2$  (wurtzite). They can be calculated from the conduction band effective mass  $m_c$ , the band gap  $E_{\text{gap}}$ , the spin-orbit split-off energy  $\Delta_{\text{so}}$  and the Kane parameter  $E_P$  (zinc blende). For wurtzite the parameters are  $m_{c,\parallel}, m_{c,\perp}, E_{\text{gap}}, \Delta_{\text{so}}$ , the crystal-field split-off energy  $\Delta_{\text{cr}}$  and the Kane parameters  $E_{P1}, E_{P2}$ .

Finally there is the inversion asymmetry parameter  $B$  for zinc blende. For wurtzite there are  $B_1, B_2, B_3$ .

For more details on these equations, please refer to Section 3.1 *The multi-band  $\mathbf{k} \cdot \mathbf{p}$  Schrödinger equation* in the PhD thesis of S. Birner.

### Spurious solutions

Some people rescale the 8-band  $\mathbf{k} \cdot \mathbf{p}$  in order to avoid *spurious solutions*. The 8-band  $\mathbf{k} \cdot \mathbf{p}$  parameters can be calculated from the 6-band parameters taking into account the band gap  $E_{\text{gap}}$ , the spin-orbit split-off energy  $\Delta_{\text{so}}$  and the Kane parameter  $E_P$  (zinc blende). For wurtzite the parameters are  $E_{\text{gap}}$ , the spin-orbit split-off energy  $\Delta_{\text{so}}$ , the crystal-field split-off energy  $\Delta_{\text{cr}}$  and the Kane parameters  $E_{P1}, E_{P2}$ .

For more details, please refer to Section 3.2 *Spurious solutions* in the PhD thesis of S. Birner.

Specific implementation in *nextnano++* can be found in its documentation, *nextnano++*.

### Specific implementation *nextnano*<sup>3</sup>

- See section *Choice of  $\mathbf{k} \cdot \mathbf{p}$  parameters* in `$numeric-control`.
- See section  *$\mathbf{k} \cdot \mathbf{p}$  parameters* in `Which material parameters are used?`.
- See section *Luttinger-parameters* in `$binary-zb-default`.

## 8.9 Can I add new materials to the database?

Yes.

### Option 1 (quick)

Directly in the input file, you can overwrite certain material parameters of any material already defined in the database. For instance if you need “HfO2”, you can use the material “SiO2” and just change the static dielectric constant and conduction and valence band edges or any other relevant parameters that you need. So basically, you are using the material “SiO2” with modified static dielectric constant and band edges.

Please note that every material has assigned crystal symmetry of either zinc blende (including diamond type) or wurtzite.

### Option 2 (general)

The material parameters are contained in ASCII text files. You can find them in the installation folder, e.g., `C:\Program Files\nextnano\<date>\nextnano++\database\database_nnp.in`.

These files can be edited with any text editor, such as *Notepad++*.

It is best if you search for a material such as “GaSb” and then simply use “Copy & Paste” to reproduce all relevant entries. Then you can rename “GaSb” to anything else, like “GaSb\_test”. Finally, you adjust

the necessary material parameters that you need. In most cases you do not have to replace all material parameters. It is only necessary to replace the ones that you need in the simulation.

It is a good idea to save the new database to a new location, such as C:\Users\<user name>\Documents\nextnano\My Database\database\_nnp\_GaSb\_modified.in You can then read in the new *nextnano*<sup>3</sup> database specifying the location within the Tools Options of *nextnanomat*.

Tools => Options... => Workspace => nextnano++ database file:

## 8.10 Should I use averaged outputs and boxes?

The averaged = yes is similar to boxes = no. Note that boxes is related to output of material grid points while averaged is related to output of simulation grid points.

2D and 3D simulations can produce a lot of output data (order of GB). It is strongly recommended to use averaged = yes for 2D and 3D simulations to avoid excessive consumption of your hard disk.

---

*Last update: 01/07/2025*

BOOKS

- [Beynon1996]  
[Buffer solutions: The basics](#)  
R. J. Beynon, and J. S. Easterby  
Oxford University Press (1996)
- [ChuangOpto1995]  
[Physics of Optoelectronic Devices](#)  
S. L. Chuang  
John Wiley & Sons, Inc., New York (1995)
- [DattaETMS1995]  
[Electronic Transport in Mesoscopic Systems](#)  
S. Datta  
Cambridge Studies in Semiconductor Physics and Microelectronic Engineering, Cambridge University Press, Cambridge (1995)  
DOI 10.1017/CBO9780511805776
- [Davies1998]  
J. H. Davies  
*The Physics of Low-Dimensional Semiconductors - An Introduction*  
Cambridge University Press (1998)
- [FaistQCL2013]  
[Quantum Cascade Lasers](#)  
J. Faist  
Oxford University Press (2013)
- [GrahnlTSP1999]  
[Introduction To Semiconductor Physics](#)  
H. T. Grahn  
World Scientific Publishing (1999)
- [HarrisonQWWD2005]  
[Quantum Wells, Wires and Dots](#)  
P. Harrison  
Theoretical and Computational Physics of Semiconductor Nanostructures, 2<sup>nd</sup> ed., John Wiley & Sons, Ltd (2005)  
DOI 10.1017/CBO9780511805776
- [Landau Lifshitz]  
*Quantum Mechanics Non-Relativistic Theory*  
L. D. Landau, E. M. Lifshitz  
(Pergamon Press, Vol. 3)
- [Shi-Dong-Liang]

R.-S. Liang,  
Quantum Tunneling and Field Electron Emission Theories  
(World Scientific, Singapore, 2014)  
p. 61

- [PiprekE2003]  
Semiconductor Optoelectronic Devices: Introduction to Physics and Simulation  
J. Piprek  
Elsevier (2003)

## JOURNAL PAPERS

- [AltermattI2006]  
A simulation model for the density of states and for incomplete ionization in crystalline silicon. I. Establishing the model in Si:P  
P. P. Altermatt, A. Schenk, G. Heiser  
J. Appl. Phys. **100**, 1113714 (2006)
- [AltermattII2006]  
A simulation model for the density of states and for incomplete ionization in crystalline silicon. II. Investigation of Si:As and Si:B and usage in device simulation  
P. P. Altermatt, A. Schenk, B. Schmitthüsen, G. Heiser  
J. Appl. Phys. **100**, 1113715 (2006)
- [Andrews2008]  
Doping dependence of LO-phonon depletion scheme THz quantum-cascade lasers  
A. M. Andrews, A. Benz, C. Deutsch, G. Fasching, K. Unterrainer, P. Klang, W. Schrenk, G. Strasser  
Materials Science and Engineering B **147**, 152 (2008)
- [Andreev2000]  
Theory of the electronic structure of GaN/AlN hexagonal quantum dots  
A. D. Andreev, E. P. O'Reilly  
Phys. Rev. B **62**, 15851 (2000)
- [Arora1982]  
Electron and hole mobilities in silicon as a function of concentration and temperature  
N.D. Arora, J.R. Hauser, D.J. Roulston  
IEEE Trans. Electron Devices **29**, 292 (1982)
- [BahderPRB1990]  
Eight-band kp model of strained zinc blende crystals  
Thomas B. Bahder  
Phys. Rev. B **41**, 11992 (1990)
- [BastardPRB1982]  
Exciton binding energy in quantum wells  
G. Bastard, E. E. Mendez, L. L. Chang, and L. Esaki  
Physical Review B **26**, 1974 (1982)
- [Beynon1988]  
A Macintosh Hypercard stack for calculation of thermodynamically-corrected buffer recipes  
R. J. Beynon  
Comput. Appl. Biosci. **4** (4), 487 (1988)
- [BirnerPhotonikInt2008]  
Simulation of quantum cascade lasers - optimizing laser performance  
S. Birner, T. Kubis, and P. Vogl

- Photonik international **2**, 60 (2008)
- [BirnerPhotonik2008]  
Simulation zur Optimierung von Quantenkaskadenlasern  
S. Birner, T. Kubis, and P. Vogl  
Photonik **1**, 44 (2008)
  - [BirnerCBR2009]  
Ballistic Quantum Transport using the Contact Block Reduction (CBR) Method - An introduction  
S. Birner, C. Schindler, P. Greck, M. Sabathil, and P. Vogl  
Journal of Computational Electronics **8**, 267–286 (2009)
  - [BelyakovBurdov2008]  
Anomalous splitting of the hole states in silicon quantum dots with shallow acceptors  
V. A. Belyakov, V. A. Burdov  
J. Phys. Condens. Matter **20**, 025213 (2008)
  - [Burdov2002]  
Electron and hole spectra of silicon quantum dots  
V. A. Burdov  
JETP **94**, 411 (2002)
  - [Canali1975]  
Electron and Hole Drift Velocity Measurements in Silicon and Their Empirical Relation to Electric Field and Temperature  
C. Canali, G. Majni, R. Minder, and G. Otaviani  
IEEE Trans. Electron Dev. **22** (11), 1045 (1975)
  - [CapassoIEEE1986]  
Resonant Tunneling Through Double Barriers, Perpendicular Quantum Transport Phenomena in Superlattices, and Their Device Applications  
F. Capasso, K. Mohammed, and A. Y. Cho  
IEEE Journal of Quantum Electronics **QE-22**, 1853 (1986)
  - [CardonaPR1966]  
Energy-band structure of Germanium and Silicon: The k.p method  
M. Cardona, F. H. Pollak  
Physical Review **142** (2), 530 (1966)
  - [CarloSST2003]  
Microscopic theory of nanostructured semiconductor devices: beyond the envelope=function approximation  
A. D. Carlo  
Semiconductor Science and Technology **18** (1), R1-31
  - [CaugheyThomas1967]  
Carrier mobilities in silicon empirically related to doping and field  
D. Caughey, R. Thomas  
Proc. IEEE **55**, 2192 (1967)
  - [Chilleri2021]  
An improved empirical model for a semiconductor's velocity-field characteristic applied to gallium arsenide  
J. Chilleri, Y. Wang, M. S. Shur, and S. K. O'Leary  
Solid State Communications **330**, 114240, (2021)
  - [DarwishIEEE1997]  
An Improved Electron and Hole Mobility Model for General Purpose Device Simulation  
M. N. Darwish, J. L. Lentz, M. R. Pinto, P. M. Zeitzoff, T. J. Krutsick, and H. H. Vuong  
IEEE Transactions on Electron Devices **44**, 1529 (1997)



- [Dehlinger2000]  
[Intersubband Electroluminescence from Silicon-Based Quantum Cascade Structures](#)  
G. Dehlinger, L. Diehl, U. Gennser, H. Sigg, J. Faist, K. Ensslin, D. Grützmacher, E. Müller  
*Science* **290**, 2277 (2000)
- [Eastman1980]  
[DESIGN CRITERIA FOR GaAs MESFETs RELATED TO STATIONARY HIGH FIELD DOMAINS](#)  
L. F. Eastman, S. Tiwiari, and M. S. Shur  
*Solid-State Electronics* **23** (4), 383 (1980)
- [Farahmand2001]  
[Monte Carlo Simulation of Electron Transport in the III-Nitride Wurtzite Phase Materials System: Binaries and Ternaries](#)  
M. Farahmand, C. Garetto, E. Bellotti, K. F. Brennan, M. Goano, E. Ghillino, G. Ghione, J. D. Albrecht, and P. P. Ruden  
*IEEE Trans. Electron Dev.* **48** (3), 535 (2001)
- [FerreiraBastard1989]  
[Evaluation of some scattering times for electrons in unbiased and biased single- and multiple-quantum-well structures](#)  
R. Ferreira, G. Bastard  
*Physical Review B* **40** (2), 1074 (1989)
- [Ferreira2006]  
[Optical Properties of Ellipsoidal CdSe Quantum Dots](#)  
W.S. Ferreira, J.S. de Sousa, J.A.K. Freire, G.A. Farias, V.N. Freire  
*Brazilian Journal of Physics* **36**, 438 (2006)
- [FowlerNordheim1928]  
[Electron emission in intense electric fields](#)  
R. H. Fowler and L. Nordheim  
*Royal Society* **119** (781), 173 (1928)
- [Friedrich2005]  
[Quantum-cascade lasers without injector regions operating above room temperature](#)  
A. Friedrich, G. Böhm, M.C. Amann, G. Scarpa  
*Applied Physics Letters* **86**, 161114 (2005)
- [GovernalePRB1998]  
[Gauge-invariant grid discretization of the Schrödinger equation](#)  
M. Governale, C. Ungarelli  
*Phys. Rev. B* **58** (12), 7816 (1998)
- [GreinAIP1995]  
[Long wavelength InAs/InGaSb infrared detectors: Optimization of carrier lifetimes](#)  
C. H. Grein, P. M. Young  
*J. Appl. Phys.* **78**, 7143 (1995)
- [HalvorsenPR2000]  
[Optical transitions in broken gap heterostructures](#)  
E. Halvorsen, Y. Galperin, K. A. Chao  
*Physical Review B* **61**, 16743 (2000)
- [Hänsch1986]  
[The hotelectron problem in small semiconductor devices](#)  
W. Hänsch and M. Miura-Mattausch  
*J. Appl. Phys.* **60**, 650 (1986)
- [Hu2005]

Resonant-phonon-assisted THz quantum-cascade lasers with metal-metal waveguides

Q. Hu, B.S. Williams, S. Kumar, H. Callebaut, S. Kohen, J.L. Reno

Semiconductor Science and Technology **20**, S228 (2005)

- [JancuPRB1998]  
Empirical spds\* tight-binding calculation for cubic semiconductors: General method and material parameters  
J.-M. Jancu, R. Scholz, F. Beltram, F. Bassani  
Physical Review B **57**, 6493 (1998)
- [Kilpstein2010]  
Operator ordering and interface-band mixing in the Kane-like Hamiltonian of lattice-matched semiconductor superlattices with abrupt interfaces  
P. C. Kilpstein  
Phys. Rev. B **81**, 235314 (2010)
- [Klaassen1992]  
A unified mobility model for device simulation - I. Model equations and concentration dependence  
D. B. M. Klaassen  
Solid-State Electronics **35**, 953 (1992)
- [KlimeckSM2000]  
sp3s\* Tight-binding parameters for transport simulations in compound semiconductors  
G. Klimeck, R. C. Bowen, T. B. Boykin, T. A. Cwik  
Superlattices and Microstructures **27** (5), 519-524 (2000)
- [KouwenhovenRPP2001]  
Few-electron quantum dots  
L.P. Kouwenhoven, D.G. Austing, and S. Tarucha  
Rep. Prog. Phys. **64**, 701 (2001)
- [Lazarenkova2001]  
Miniband formation in a quantum dot crystal  
O. L. Lazarenkova and A. A. Balandin  
J. Appl. Phys. **89**, 5509 (2001)
- [LenzlingerSnow1969]  
Fowler-Nordheim Tunneling into Thermally Grown SiO<sub>2</sub>  
M. Lenzlinger and E. H. Snow  
J. Appl. Phys. **40**, 278 (1969)
- [LeverJLT2010]  
Design of Ge–SiGe Quantum-Confined Stark Effect Electroabsorption Heterostructures for CMOS Compatible Photonics  
L. Lever, Z. Ikonik, A. Valavanis, J. D. Cooper, and R. W. Kelsall  
Journal of Lightwave Technology **28**, 3273 (2010)
- [\_LivnehPRB2012]  
k,p model for the energy dispersions and absorption spectra of InAs/GaSb type-II superlattices  
Y. Livneh, P. C. Klipstein, O. Klin, N. Snapi, S. Grossman, A. Glozman, and E. Weiss  
Physical Review B **86**, 235311 (2012)
- [LombardiIEEE1988]  
A physically based mobility model for numerical simulation of nonplanar devices  
C. Lombardi, S. Manzini, A. Saporito, and M. Vanzi  
IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems **7**, 1164 (1988)
- [MamaluyCBR2003]  
Efficient method for the calculation of ballistic quantum transport

- D. Mamaluy, M. Sabathil, and P. Vogl  
J. Appl. Phys. **93**, 4628 (2003)
- [MasettiIEEE1983]  
Modeling of Carrier Mobility Against Carrier Concentration in Arsenic-, Phosphorus- and Boron-Doped Silicon  
G. Masetti, M. Severi, S. Solmi  
IEEE Trans. Electron Devices, Vol. ED-30 (7), 764 (1983)
  - [Page2001]  
300 K operation of a GaAs-based quantum-cascade laser at  $\lambda=9\ \mu\text{m}$   
H. Page, C. Becker, A. Robertson, G. Glastre, V. Ortiz, C. Sirtori  
Appl. Phys. Lett. **78** (22), 3529 (2001)
  - [ParkChuang2000]  
Comparison of zinc-blende and wurtzite GaN semiconductors with spontaneous polarization and piezoelectric field effects  
S.-H. Park, S.L. Chuang  
J. Appl. Phys. **87**, 353 (2000)
  - [Park2000]  
Crystal Orientation Effects on Electronic Properties of Wurtzite GaN/AlGaIn Quantum Wells with Spontaneous and Piezoelectric Polarization  
S.-H. Park  
J. Appl. Phys. **39**, 3478 (2000)
  - [ReichPR2002]  
Tight-binding description of graphene  
S. Reich, J. Maultzsch, C. Thomsen, P. Ordejon  
Physical Review B **66**, 035412 (2002)
  - [RideauPRB2006]  
Strained Si, Ge, and Si(1-x)Ge(x) alloys modelled with a first-principles-optimized full-zone k.p method  
D. Rideau, M. Feraille, L. Ciampolini, M. Minondo, C. Tavernier, H. Jaouen, A. Ghetti  
Physical Review B **74** (19), 195208 (2006)
  - [Rochat2005]  
Low-threshold terahertz quantum-cascade lasers  
M. Rochat, L. Ajili, H. Willenberg, J. Faist, H. Beere, G. Davies, E. Linfield, D. Ritchie  
Applied Physics Letters **81** (8), 1381 (2002)
  - [SaitoS2001]  
Optical Properties and Raman Spectroscopy of Carbon Nanotubes  
M.S. Dresselhaus, G. Dresselhaus, P. Avouris (Eds.)  
Topics in Applied Physics **80**, Springer (2001)
  - [Sarma2002]  
Realistic tight-binding model for the electronic structure of II-VI semiconductors  
S. Sapra, N. Shanthi, D. D. Sarma  
Physical Review B **66**, 205202 (2002)
  - [SawamuraOME2018]  
Nearest-neighbor sp<sup>3</sup>d<sup>5</sup>s\* tight-binding parameters based on the hybrid quasi-particle self-consistent GW method verified by modeling of type-II superlattices  
A. Sawamura, J. Otsuka, T. Kato, T. Kotani, S. Souma  
Optical Materials Express **8**, 1569 (2018)
  - [Schwierz2010]  
Reversed Anionic Hofmeister Series: The Interplay of Surface Charge and Surface Polarity

N. Schwierz, D. Horinek, R. R. Netz  
Langmuir **26**, 7370 (2010)

- [SirtoriPRB1994]  
Nonparabolicity and a sum rule associated with bound-to-bound and bound-to-continuum intersubband transitions in quantum wells  
C. Sirtori, F. Capasso, J. Faist, and S. Scandolo  
Physical Review B **50**, 8663 (1994)
- [Sirtori1998]  
GaAs/AlGaAs quantum cascade lasers  
C. Sirtori, P. Kruck, S. Barbieri, P. Collot, J. Nagle, M. Beck, J. Faist, U. Oesterle  
Applied Physics Letters **73** (24), 3486 (1998)
- [TsuEsaki1973]  
Tunneling in a finite superlattice  
R. Tsu and L. Esaki  
Appl. Phys. Lett. **22**, 562 (1973)
- [VurgaftmanJAP2001]  
Band parameters for III-V compound semiconductors and their alloys  
I. Vurgaftman and J. R. Meyer  
J. Appl. Phys. **89**, 5815 (2001)
- [VurgaftmanJAP2003]  
Band parameters for nitrogen-containing semiconductors  
I. Vurgaftman and J. R. Meyer  
J. Appl. Phys. **94**, 3675 (2003)
- [VoglJPCS1983]  
A Semi-Empirical Tight-Binding Theory of the Electronic Structure of Semiconductors  
P. Vogl, H. P. Hjalmarson, and J. D. Dow  
Journal of Physics and Chemistry of Solids **44** (5), 365 (1983)
- [ZakharovaPR2001]  
Hybridization of electron, light-hole, and heavy-hole states in InAs/GaSb quantum wells  
A. Zakharova, S. T. Yen, K. A. Chao  
Physical Review B **64**, 235332 (2001)

- [AndlauerPhD2009]  
[Optoelectronic and spin-related properties of semiconductor nanostructures in magnetic fields](#)  
T. Andlauer  
Selected Topics of Semiconductor Physics and Technology (G. Abstreiter, M.-C. Amann, M. Stutzmann, and P. Vogl, eds.), Vol. **105**, Verein zur Förderung des Walter Schottky Instituts der Technischen Universität München e.V., München, 157 pp. (2009)
- [BirnerPhD2011]  
[Modeling of semiconductor nanostructures and semiconductor-electrolyte interfaces](#)  
S. Birner  
Selected Topics of Semiconductor Physics and Technology (G. Abstreiter, M.-C. Amann, M. Stutzmann, and P. Vogl, eds.), Vol. **135**, Verein zur Förderung des Walter Schottky Instituts der Technischen Universität München e.V., München, 239 pp. (2011)  
ISBN 978-3-941650-35-0
- [KubisPhD2009]  
[Quantum transport in semiconductor nanostructures](#)  
T. C. Kubis  
Selected Topics of Semiconductor Physics and Technology (G. Abstreiter, M.-C. Amann, M. Stutzmann, and P. Vogl, eds.), Vol. **114**, Verein zur Förderung des Walter Schottky Instituts der Technischen Universität München e.V., München, 253 pp. (2009)
- [SchusterPhD2005]  
[Hochauflösende optische Spektroskopie an niedrigdimensionalen Halbleiterstrukturen](#)  
R. Schuster  
PhD Thesis, University of Regensburg, Germany, 2005