Eidgenössische Technische Hochschule Zürich
Ecole polytechnique fédérale de Zurich
Politecnico federale di Zurigo
Swiss Federal Institute of Technology Zurich

Integrated Systems Laboratory

Mathieu Luisier and Anne Ziegler

# QTsolver

## Manual

**V3 - September 2015**

# Contents

# Chapter 1

# Introduction

## 1.1 A quantum transport simulator

The development of new transistors has always been supported by computer aided design (CAD) tools that should be able to predict the device characteristics before their fabrication. As the gate length of MOSFETs was longer than 100 nm and their cross section larger than 30 nm, the classical drift-diffusion (DD) approach [1, 2, 3] was accurate enough to give good predictions about transistor performances. The major concern about the DD model is that it does not capture energy quantization, quantum mechanical tunneling, and the wave nature of electrons and holes, which are all essential at the nanometer scale. Hence, it is becoming critical to accompany DD with more advanced simulation approaches that will facilitate the discovery and the emergence of novel nanoelectronic devices.

A direct and self-consistent solution of the single-electron Schrödinger equation with open boundary conditions is more accurate than DD and fulfills the quantum mechanical requirement, but demands more computational power. However, the continuous increase of the CPU performances in the recent years represents a fantastic opportunity to re-think transistor simulation at the nanometer scale and go beyond standard approaches.

In this context, we have developed QTsolver, a next generation, multi-dimensional CAD tool based on quantum mechanical concepts and dedicated to the simulation of nanoelectronic devices [4, 5, 6, 7, 8]. Due to parallelization and advanced numerical algorithms QTsolver can be used to study the electrical transport in a wide range of device structures, some of them being shown in Figure 1.1.

QTsolver has been designed to accomplish three main tasks, all within the effective mass approximation (EMA):

1. calculating the electron bandstructure of 2-D and 3-D nanostructures with any crystal and/or confinement direction,

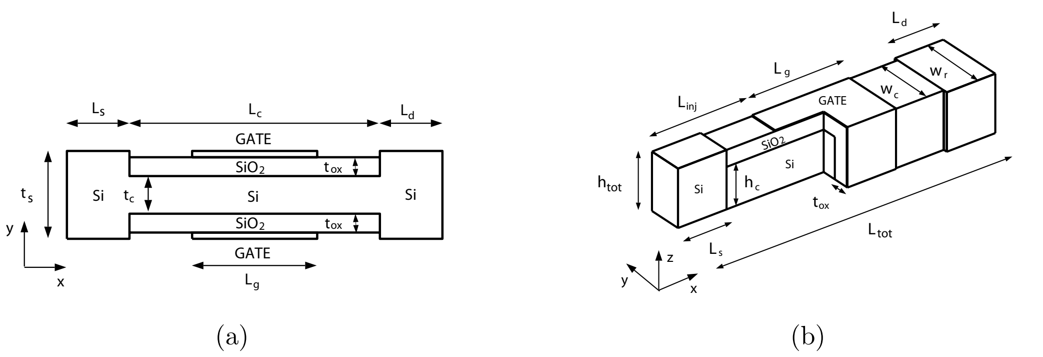


Figure 1.1: **Schematic view of device structures that the solver can simulate** (a) 2D Double-Gate Ultra-Thin-Body MOSFET, (b) 3D Tri-Gate MOSFET. The transistors can be made of any materials whose effective masses are known. The parameter $L_g$ refers to the gate length, $t_{body}$ to the body thickness, and $t_{ox}$ to the oxide layer thickness.[4]

2. computing the electron transmission probability through infinitely long [1] 2-D and 3-D nanostructures and the corresponding density-of-states (DOS),

3. calculating the self-consistent electronic current flowing through 2-D and 3-D nanostructures in the ballistic limit of transport.

### 1.1.1 Formulation of the quantum transport problem

QTsolver solves multi-dimensional Schrödinger equations with open boundary conditions (OBCs) in the Wave Function (WF) formalism. The carrier and current densities of the simulated nanostructures are obtained by self-consistently coupling the solution of the Schrödinger and Poisson equations till convergence is reached. This must be repeated for multiple bias points in order to obtain $I$-$V$ (current vs. voltage) characteristics.

The core operation in QTsolver is the solution of the effective-mass Schrödinger equation either in the real-space or mode-space approximation for each electron (hole) energy $E$ and at each bias point. For ballistic simulations the Wave Function approach is chosen because it gives exactly the same results as the well-known Non-equilibrium Green's Function (NEGF) formalism, but much faster. The Schrödinger-Poisson solver is used to calculate the carrier and current densities as well as the electrostatic potential of the device. The simulation domain is discretized in the finite difference method so that $x$ ($y$) becomes a vector with $N_\mathrm{x}$ ($N_\mathrm{y}$) entries $x_\mathrm{i}$ ($y_\mathrm{j}$).

[1]infinite means here that the simulation domain and the contacts are identical and have the same structure and properties

In quantum wells or ultra-thin-bodies (2-D) transport occurs along the $x$ direction (convention), $y$ is a direction of confinement along the body or quantum well thickness, and $z$ is assumed periodic. In this case, the momentum vector $k$ reduces to a single component $k{=}k_z$. For computational efficiency, $k_z$ is not explicitly discretized in QTSOLVER, but accounted for indirectly in the electron/hole distribution functions of the contact regions. Finally, in nanowires or FinFET's (3-D), $y$ and $z$ are directions of confinement and $k{=}0$ (no momentum dependence).

Open boundary conditions are introduced to couple the simulation domain to its surrounding environment (semi-infinite contacts or reservoirs) and allow for electrons to enter and exit a finite domain.

### 1.1.2 Real-space approximation

The 2-D EMA Schrödinger equation, in the finite difference method, can be written at each ($x_\mathrm{i}$,$y_\mathrm{j}$) point as:

$$\begin{aligned}(E - H_{\mathrm{iijj}})\,\phi_{\mathrm{ij}} - H_{\mathrm{ii+1jj}}\phi_{\mathrm{i+1j}} - H_{\mathrm{ii-1jj}}\phi_{\mathrm{i-1j}}\\ - H_{\mathrm{iijj+1}}\phi_{\mathrm{ij+1}} - H_{\mathrm{iijj-1}}\phi_{\mathrm{ij-1}} = 0,\end{aligned} \qquad (1.1)$$

where the Hamiltonian matrix element $H_{\mathrm{i_1i_2j_1j_2}}$ are defined as in [9], $\phi_{\mathrm{ij}}$ is the wave function $\phi(x_\mathrm{i}, y_\mathrm{j})$, and $E$ the electron energy. Equation 1.1 can be generalized to 3-D structures, which is not shown here for brevity. At the source, drain, and gate contacts, a single band scattering boundary ansatz[2, 5] is applied to model the open boundary conditions (OBC)

$$\begin{aligned}(\mathbf{E} - \mathbf{H}_{\mathrm{nn}})\,\boldsymbol{\phi}_\mathbf{n} - \mathbf{H}_{\mathrm{nn+1}}\boldsymbol{\phi}_\mathbf{n+1} - \mathbf{H}_{\mathrm{nn-1}}\boldsymbol{\phi}_\mathbf{n-1} &= 0, & (1.2)\\ \boldsymbol{\phi}_\mathbf{n\pm1} &= \boldsymbol{\phi}_\mathbf{n}\mathrm{e}^{\pm i\mathbf{k}_\mathrm{n}\Delta_\mathrm{n}}. & (1.3)\end{aligned}$$

The energy matrix $\mathbf{E}$ is diagonal, $\mathbf{H}_{\mathrm{nn}}$ is tri-diagonal and describes the on-site energies and connections within one grid line, $\mathbf{H}_{\mathrm{nn\pm1}}$ is diagonal and represents the connection of one grid line to the next (+ sign) or previous (− sign) one, and $\boldsymbol{\phi}_\mathbf{n}$ is a vector containing the wave function along one grid line. The index $n$ is equal to $S$, $D$, or $G$ and is used to characterize the position of one grid line. For example, if the source contact is considered, $n$ denotes the vertical grid line situated at $x{=}x_\mathrm{S}$ ($\boldsymbol{\phi}_\mathbf{n}{=}\phi(x_\mathrm{S}, y)$, $\boldsymbol{\phi}_\mathbf{n\pm1}{=}\phi(x_\mathrm{S}\pm\Delta_\mathrm{x}, y)$). For the gate contact, $n$ refers to the horizontal line with the $y$-coordinates equal to $y_\mathrm{G}$ ($\boldsymbol{\phi}_\mathbf{n} = \phi(x, y_\mathrm{G})$, $\boldsymbol{\phi}_\mathbf{n\pm1}{=}\phi(x, y_G \pm \Delta_y)$). In the scattering boundary theory the contacts are just the extension of the device grid line they are connected to. Hence, the contact wave functions taken along one grid line orthogonal to the injection direction are identical up to a phase factor $\mathrm{e}^{i\mathbf{k}_\mathrm{n}\Delta_\mathrm{n}}$. The variable $\Delta_\mathrm{n}$ is the distance between two lines. Furthermore, $\mathbf{H}_{\mathrm{nn+1}} = \mathbf{H}^\mathbf{T}_{\mathrm{n+1n}}$ holds and it can be proved, as long as the effective mass tensor is diagonal, that

$$\mathbf{H_{nn+1}} = \mathbf{H_{nn-1}} = \mathbf{T_n} \tag{1.4}$$

is valid in the contacts since both $\mathbf{H_{nn+1}}$ and $\mathbf{H_{nn-1}}$ are diagonal matrices. Inserting Eq. (1.4) into (1.3) results in the eigenvalue problem

$$\underbrace{\mathbf{T_n^{-1}} \cdot (\mathbf{E} - \mathbf{H_{nn}})}_{\mathbf{M}} \boldsymbol{\phi_n} = \underbrace{-2 \cdot \cos(\mathbf{k_n})}_{\lambda} \cdot \boldsymbol{\phi_n}. \tag{1.5}$$

All the eigenvalues $\lambda$ of $\mathbf{M}$ are required.
The boundary wave functions $\boldsymbol{\phi_S}$, $\boldsymbol{\phi_D}$, and $\boldsymbol{\phi_G}$ and vectors $\mathbf{k_S}$, $\mathbf{k_D}$, and $\mathbf{k_G}$ resulting from Eq. (1.5) are used to calculate the source, drain, and gate boundary self-energies $\mathbf{\Sigma_S}$, $\mathbf{\Sigma_D}$, and $\mathbf{\Sigma_G}$, respectively as well as the injection matrix $\mathbf{S_{inj}}$[10, 5]. Finally, Eq. (1.1) is cast into a sparse linear problem with the following form

$$\underbrace{(\mathbf{E} - \mathbf{H} - \mathbf{\Sigma_S} - \mathbf{\Sigma_D} - \mathbf{\Sigma_G})}_{\mathbf{A}} \cdot \boldsymbol{\phi} = \mathbf{S_{inj}}. \tag{1.6}$$

The matrices $\mathbf{\Sigma_S}$ and $\mathbf{\Sigma_D}$ vanish everywhere except in the left and right corner of $\mathbf{A}$, $\mathbf{\Sigma_G}$ occupies a large sparse block in the middle of $\mathbf{A}$ and destroys its block tri-diagonal structure inherited from $\mathbf{H}$[11]. The $N_{\mathrm{S}}$ states injected from the source, $N_{\mathrm{D}}$ from the drain, and $N_{\mathrm{G}}$ from the gate are included in the $(N_{\mathrm{S}} \cdot N_{\mathrm{D}} \cdot N_{\mathrm{G}}) \times (N_{\mathrm{x}} \cdot N_{\mathrm{y}})$ matrix $\mathbf{S_{inj}}$. The linear system in Eq. (1.6) is solved with a direct sparse linear solver Umfpack [12], MUMPS [13], SuperLU$_{dist}$ [14], Pardiso [15].

Equation (1.6) is solved for each electron/hole energy $E$ and each conduction/valence band valley of the material. Once the wave functions $\boldsymbol{\phi}(E)$ are known the carrier density $n(x_{\mathrm{i}}, y_{\mathrm{j}})$ and the ballistic current density $\mathbf{J}(x_{\mathrm{i}}, y_{\mathrm{j}})$ are calculated according to

$$\mathbf{n}(x_{\mathrm{i}}, y_{\mathrm{j}}) = \frac{1}{\Delta_{\mathrm{x}}\Delta_{\mathrm{y}}} \sum_{\mathrm{n,v}} \int \frac{\mathrm{d}E}{2\pi} |\phi_{\mathrm{n,v}}(x_{\mathrm{i}}, y_{\mathrm{j}}; E)|^2 \tag{1.7}$$

$$\mathbf{J}(x_{\mathrm{i}}, y_{\mathrm{j}}) = -\frac{2e}{\Delta_{\mathrm{x}}\Delta_{\mathrm{y}}\hbar} \sum_{\mathrm{n,v}} \int \frac{\mathrm{d}E}{2\pi} \tag{1.8}$$

$$\times \mathrm{Re} \begin{pmatrix} \phi^*_{\mathrm{n,v}}(x_{\mathrm{i+1}}, y_{\mathrm{j}}; E) \cdot H_{\mathrm{i+1ijj}} \cdot \phi_{\mathrm{n,v}}(x_{\mathrm{i}}, y_{\mathrm{j}}; E) \cdot \Delta_{\mathrm{x}} \\ \phi^*_{\mathrm{n,v}}(x_{\mathrm{i}}, y_{\mathrm{j+1}}; E) \cdot H_{\mathrm{iij+1j}} \cdot \phi_{\mathrm{n,v}}(x_{\mathrm{i}}, y_{\mathrm{j}}; E) \cdot \Delta_{\mathrm{y}} \end{pmatrix} \tag{1.9}$$

The index $n$ ($S$, $D$, or $G$) refers to the origin of the wave function and $v$ is the valley index. They both indicate from which part/valley the state was injected. The

Fermi levels of the contacts are already taken into account in the wave function. They determine the probability that a state injected at an energy $E$ is occupied[5]. The ultimate carrier density $n(x_\mathrm{i}, y_\mathrm{j})$ is obtained after a self-consistent calculation of the 2D (3D) electrostatic potential in the device. The drain and gate currents then follow from

$$I_\mathrm{d}(x_\mathrm{i}) = \int \mathrm{d}y \; J_\mathrm{x}(x_\mathrm{i}, y), \tag{1.10}$$

$$I_\mathrm{g}(y_\mathrm{G}) = \int_{x_\mathrm{G1}}^{x_\mathrm{G2}} \mathrm{d}x \; J_\mathrm{y}(x, y_\mathrm{G}). \tag{1.11}$$

Current continuity implies that the difference in the drain current between the two gate corners exactly corresponds to what escapes from the gate, i.e. $I_\mathrm{d}(x_\mathrm{G2})$-$I_\mathrm{d}(x_\mathrm{G1})$=$I_\mathrm{g}(y_\mathrm{G})$.

### 1.1.3 Mode-space approximation

The coupled mode space approach, while keeping all the relevant physics, considerably simplifies the high computational burden of a real space simulation. The idea consists in separating the transport direction x from the directions of the confinement y (z). The discretized real space wave function $\Psi_{ij}(E)$ can be expanded in a coupled mode (eigenfunction) space as

$$\Psi_{ij}(E) = \sum_\mathrm{n} \Psi_{in}(E)\phi_n^i(y_j) \tag{1.12}$$

The modes $\phi_n^i$ should form a complete orthogonal basis, which implies that the number of modes should be equal to the number of discretization points along the confinement direction, $N_y$. However, in nanostructures with strong confinement, only a few low energy modes are populated depending on the geometry of the device, the effective mass in the direction of confinement, and the doping concentration. Consequently, only a reduced number of modes $N_m$ need to be considered, with the property $N_m \ll N_y$: $N_m$ is chosen in such a way that increasing the number of modes to a value superior to it does not change the carrier and current densities any more. The total mode space wave function $\Psi_{ms}$ has the size $(N_xN_m)$ instead of $(N_xN_y)$ for its real space counterpart $\Psi_{rs}$. To find the transformation from $\Psi_{rs}$ to $\Psi_{ms}$, $v^i = [\phi_1^i \dots \phi_{N_m}^i]$ matrices of size $(N_y \times N_m)$ are initially created. At position $x_i$, $v^i$ contains all the modes necessary to expand the real space Wave function localized there. In a second step, a transformation matrix $U$, with size $(N_xN_y \times N_xN_m)$, is generated: it contains the $N_x$ $v^i$'s defined above. The relationship between $\Psi_{ms}$ and $\Psi_{rs}$ is given by:

$$\Psi_{rs} = U \cdot \Psi_{ms} \tag{1.13}$$

Inserting Equation 1.13 into 1.6 defines the real space to mode-space transformation of the Schrödinger equation.

$$\mathbf{U}^T \times \left(\mathbf{E}_{RS} - \mathbf{H}_{RS} - \mathbf{\Sigma}_{RS}^{RB}\right) \times \mathbf{U} \times \mathbf{\Psi}_{MS} \quad = \quad U^T \times \mathbf{S}_{RS}^{inj}, \tag{1.14}$$

and has the size $(N_x N_m \times N_x N_m)$. Equation 1.15 can be rewritten as

$$\left(\mathbf{E}_{MS} - \mathbf{H}_{MS} - \mathbf{\Sigma}_{MS}^{RB}\right) \times \mathbf{\Psi}_{MS} \quad = \quad \mathbf{S}_{MS}^{inj}, \tag{1.15}$$

Usually, the real space Hamiltonian matrix $\mathbf{H}_{RS}$ is first constructed and then transformed into mode-space while $\Sigma_{MS}^{RB}$ and $S_{MS}^{inj}$ are directly computed in the mode-space. Note that $\mathbf{H}_{MS}$ still has a block tri-diagonal structure, but its blocks might be full (coupled modes) or diagonal (uncoupled modes).

Solving the transport equation in the coupled mode space presents a substantial advantage over the real space calculation because the size of the linear system decreases from$(N_x N_y \times N_x N_y)$ to $(N_x N_m \times N_x N_m)$. There is a gain of $N_y/N_m$ in the size of the blocks building the Hamiltonian $H_{ms}$. After the solution of the standard Equation 1.15 is obtained, carrier and current densities can be computed:

$$n(x_i, y_j) = \text{trace}\left\{U \cdot \left(-\frac{i}{L_z \Delta_x \Delta_y} \sum_{m,n} \int \frac{\mathrm{d}E}{2\pi} \mathrm{c}_m^*(x_i)) \cdot c_n(x_i)\right) \cdot U^T\right\}, \tag{1.16}$$

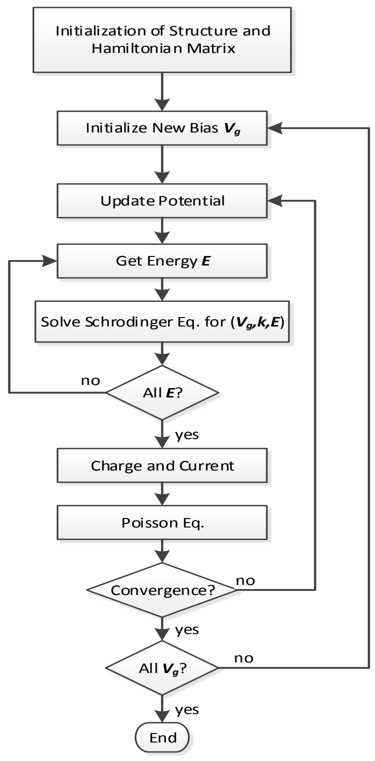
$$\begin{aligned} J_{x_i} = -\frac{e}{\hbar L_z \Delta_y} \sum_{m,n} \int \frac{\mathrm{d}E}{2\pi} \,\text{trace}\Big\{ & \left[H_{i+1imn}^{\mathrm{ms}} \cdot c_m^*(x_{i+1})) \cdot c_n(x_i))\right] \\ & - \left[c_m^*(x_i)) \cdot c_n(x_{i+1})) \cdot H_{ii+1mn}^{\mathrm{ms}}\right] \Big\}. \end{aligned} \tag{1.17}$$

All the correlation elements are involved in the calculation through the construction of the *cc* matrix from the mode-space wave function.

## 1.2 Parallelization Scheme

Depending on the size of the investigated structure the simulation time on a single processor may be too long. Hence, QTsolver uses mainly the message passing interface (MPI) [16] to distribute the workload across multiple nodes and reduce the simulation time. For example, if the transmission and the DOS of a nanowire need to be calculated for $N_E$ different energy points, each processor will treat $N_E/N$ points where $N$ is the number of available CPU's . The number of cores is selected by the user at the beginning of the simulation. This leads to a speed up factor of $N$. Figure 1.2 shows a flowchart illustrating the working principle of the simulator including this parallelization level.

Additionally, QTSolver is able to perform spatial domain decomposition with the MUMPS or the Paradiso solver. In the input deck the parameter *CPU_ppoint* can be adjusted to decompose the device structure along the x-axis. The parameter usage is further explained in Chapter 2.



Initialization of Structure and Hamiltonian Matrix
Initialize New Bias $V_g$
Update Potential
Get Energy $E$
Solve Schrodinger Eq. for ($V_g$,$k$,$E$)
no
All $E$?
yes
Charge and Current
Poisson Eq.
Convergence?
no
yes
All $V_g$?
no
yes
End

(a)

```
1:  Construction of the simulation domain
2:  Generation of the Hamiltonian matrix $H$
3:  Initialization of the FDM Poisson environment
4:  **repeat**
5:    Every bias point $V_{gs}/V_{ds}$
6:    Get source and drain Fermi levels
7:    Get initial guess for electrostatic potential $V$
8:    **repeat**
9:      Update electrostatic potential $V \to H$
10:     Compute contact bandstructure
11:     Generate energy grid $E(k)$
12:     **for** Every energy point $E$ (in parallel) **do**
13:       Compute open boundary
14:       Solve Schrödinger Eq. (in parallel)
15:       Extract DOS and TE
16:     **end for**
17:     Compute charge $\rho$ and current $J_d$
18:     Solve FEM Poisson equation $\rho \to V$
19:   **until** Self-consistent convergence of $\rho$, $V$
20: **until** All bias points are covered
```

(b)

Figure 1.2: QTSOLVER **working principle**. (a) Flow chart of the solver functionality. The energy loop on the left-hand-side is parallelized while the loops on the right-hand-side (Bias Points and Poisson solver) are not. (b) QTSOLVER algorithms.

## 1.3 Compiling the Code

QTSOLVER relies on several external packages that need to be compiled before it and then linked to it: AMD [17], Arpack [18], Aztec [19], BLAS [18], Lapack [18], Metis [20], MUMPS [21], Pardiso [22], SuperLU_*DIST* [23], UFconfig [24], and UMFPACK [25]. The message passing interface (MPI) needs to be available on the compilation platform. It is strongly recommended not to compile the BLAS and Lapack libraries, but to use those provided by vendors (ACML, MKL, GotoBLAS, or ESSL). Significant speed-up can be obtained if these high-performance numerical libraries are leveraged.

The source code of QTsolver is usually stored in a directory called "QTsolver", which should be placed in a parent directory containing all the other required libraries. A file Makefile is available to compile each library, one after the other, terminating with QTsolver. For that purpose, the user should define their compile options in a file make.inc placed in the parent directory where all the libraries and the source code of QTsolver are stored. To summarize, three steps are necessary to compile QTsolver

1. if not already available, downloading all the necessary solver libraries mentioned above (AMD, Arpack, Aztec,$\cdots$),
2. updating the file make.inc with compiler options and linking the Makefile of each library with this file,
3. entering the parent directory where the file Makefile is stored and typing "make all".

## 1.4 Starting a Job

The input of QTsolver is a command file command.cmd including a list of parameters and tasks to perform, for example a bandstructure calculation or a self-consistent simulation. A job is launched with

*mpiexec -n N* QTsolver *command.cmd*

if $N$ CPUs are available or

QTsolver *command.cmd*

on a single CPU. In order that the parallel version works a recent version of MPI must be installed.
Additionally, the environment variable *OMP_NUM_THREADS* can be set to make the Paradiso solver work on multiple threads. Be aware that the $N_x$ OMP_NUM_THREADS must be smaller or equal to the number of available cores.

## 1.5 Simulation Flow

Irrespective of the simulation type (bandstructure, transmission, or self-consistent), the first computational steps of QTsolver remain the same

1. 2-D or 3-D nanostructures are created according to the specifications given in the command file,
2. the Hamiltonian matrix corresponding to the structure as well as the boundary matrices are generated. When the nanostructure is created, a nearest-neighbor table is written. It contains the $x$, $y$, and $z$ positions of the grid

point, their type, and their nearest neighbors (either 6 or 18, depending on the presence or not of the non-diagonal terms in the inverse effective mass tensor).

After the initialization of the Hamiltonian matrix the simulation flow goes on separated directions, depending on the task to accomplish. The format of the input and output variables corresponding to each possible task is explained in details in Chapter 2 and 3 of this manual.

# Chapter 2

# Input Parameters

In this chapter we give detailed information about the available input parameters of QTsolver, if available their default value and unit. If needed they can be changed in the input deck.

## 2.1 Input deck overview

Here, we give an overview to the input platform of QTsolver. This consists of five parts:

1. **Simulation parameters -** Definition of the simulation environment,

2. **Energies, momenta, potentials -** Setting of the intervals and discretization,

3. **Material parameters -** Definition of the different materials and their properties,

4. **Device parameters -** Description of the device structure and geometry,

5. **Commands -** Setting of the simulations type.

The input deck is stored as a *.cmd* file format. Figure 2.1 shows the input deck specification for the simulation of the 2D Double Gate FET of Chapter 4. In the following sections we explained every available parameter option in the associated part of the input deck.

```
/*Simulation parameters*/
poisson_criterion        = 1e-3;
poisson_iteration        = 50;
NDim                     = 2;
Temp                     = 300;
CPU_ppoint               = 1;

/*Energies, momenta and potentials*/
n_of_modes               = 12;
Nk                       = 1001;
last_first               = 0;
eta                      = 0.0;
Elimit                   = 50e-3;
NVG                      = 15;
Vgmin                    = -0.1;
Vgmax                    = 0.6;
...

/*Material parameters*/
n_of_valleys             = 1;
deg_factor               = 1;

//Si Channel
Eg_wire                  = 0.0;
Xi_wire                  = 4.651;
Eps_wire                 = 14.4244;
me_x_wire                = 0.92;
me_y_wire                = 0.19;
...

/*Device parameters*/
no_mat                   = 9;
Lc                       = 12.0;
Ls                       = 20.0;
Ld                       = 20.0;
ts                       = 5.0;
td                       = 5.0;
tc                       = 5.0;
hox1                     = 3.0;
hox2                     = 3.0;
tcont                    = 0.0;

xmin                     = 0.0;
xmax                     = Ls+Lc+Ld;
dx                       = 0.2;

ymin                     = 0-hox1-tcont;
ymax                     = tc+hox1+tcont;
dy                       = 0.2;

mat_Eg(1)                = Eg_wire;
mat_Eps(1)               = Eps_wire;
mat_me_x(1)              = me_x_wire;
mat_me_y(1)              = me_y_wire;
...

/*Commands*/
command(1)               = Bandstructure;
command(2)               = Transmission_UWF;
```

Figure 2.1: **Input deck of the 2D Double Gate FET of Chapter 4.**

## 2.2 Simulation parameters

The following parameters define the simulation environment.

- *poisson_criterion*: Convergence criterion for the outer loop of Poisson equation. Default: 1e-3. Data type: Double.
- *poisson_iteration*: Maximum number of outer Poisson iterations per bias point. Default: 15. Data type: Integer.
- *plot_transmission*: If 1, the calculated transmission is written into an output file for all the energy points and at each bias point. The energy resolved spectral current is also written to file. Default 0 (not written to a file).
- *plot_dos*: If 1 the calculated density-of-states is written into a output file for all the energy points and at each bias point. Default 0 (not written to a file).
- *plot_valleys*: If 1 the calculated current per valley and charge density per valley are written into an output file. Default 0 (not written to a file).
- *plot_modes*: If 1 the calculated eigenvalues, the charge density and drive current per valley and per mode are written into an output file for all bias points. Default 0 (not written to a file).
- *CPU_ppoint*: number of CPUs per energy point, integer value. In large device structures, QTSOLVER offers the possibility to spatially decompose the simulation domain in *CPU_ppoint* slices along the x-axis, each of them having roughly the same width. This means that the corresponding Schrödinger equation is solved in parallel on *CPU_ppoint* different CPUs. This option is only possible if a parallel sparse linear solver such as MUMPS or SuperLU_*DIST* is selected with the command option. Default: 1. Data type: Integer.
- *doping_from_file*: If 1 then the doping concentration at each discretization point of the device structure is stored in a file. Default: 0.
- *doping_file*: File containing the doping concentration at each discretization point of the device structure. This file must have the same length as the matrix containing the x,y,z coordinates.
- *self_consistent*: If 0, the self-consistent loop between the Schrödinger and Poisson equations is turned off. Default: 1.
- *restart*: Parameter to restart QTSOLVER from existing results. In the input deck, restart=[restart opt IG start IS start ID start] where restart opt=1, 2, 3, or 4. If restart opt=1, QTSOLVER needs to restart the simulation at the beginning of the bias point with index (IG start, IS start, ID start). Some bias points were already computed and the corresponding electrostatic potentials are stored in a file passed to the input vtot_file (see below). If restart opt=2,

QTsolver needs to restart in the middle of the bias point with index (IG start, IS start, ID start), which usually corresponds to the first bias point of the simulation. The latest electrostatic potential that QTsolver calculated is stored in the file passed to the parameter vact_file. If restart opt=3, QTsolver needs to restart in the middle of the bias point with index (IG start,IS start,ID start), some bias points were already computed (their electrostatic potential is passed to vtot_file), and the latest electrostatic potential QTsolver generated is passed to vact_file. If restart opt=4, the electrostatic potential is set to 0 everywhere.

- *vtot_file*: File containing the electrostatic potential of all the bias points that were already successfully computed by QTsolver before a simulation is restarted.
- *vact_file*: File containing the latest electrostatic potential that QTsolver produced before a simulation is restarted. This option can also be used to load a pre-defined potential in band structure, transmission, or DOS calculations.
- *directory*: Directory where the results are saved. If it is not defined, the results are saved in the current directory.

## 2.3 Energies, momenta and potentials

This subsection describes how to set the energy, momentum and voltage options.

- *mode_space*: If 1, the simulation is performed in the mode-space approximation ad the number of modes must be specified, e.g. [1 16] turns on the mode space approximation and uses 16 modes to represent the confined direction. Default: 0 (real-space calculation). Data type: Integer.
- *recompute_basis*: If mode_space is turned on, the basis is computed every *recompute_basis* Poisson iteration. Default: 1 (recomputed at every Poisson iteration). Data type: Integer.
- *n_of_modes*: Number of modes for band structure calculation. This value is also used when the source and drain bandstructures are calculated to determine the electron energies at which the Schrödinger equation is solved. Default: 16. Data type: Integer.
- *Nk*: Number of k points in band structure calculation. Default: 51. Data type: Integer.
- *last_first*: If 1, the open boundary conditions of one contact only are calculated. If 0, both the source and the drain self-energies are calculated. Option only valid in bandstructure and transmission calculations (flat-band potential assumed). Default: 1.

- *eta*: Imaginary part of the energy in the device. Default: 1e-12 [eV]. Data type: Double.
- *eta_bound* : Not used.
- *Temp*: Operating temperature. Default: 300 [Kelvin].
- *UT*: Thermal Energy. Assistance variable calculated as $k_B\cdot$ *Temp* $/q$.
- *Elimit*: Energy interval that is refined after a mode is detected in the contact band structure. In this interval the distance between two energy points growths from dE_in to dE_f. Default: 50e-3 [eV]. Data type: Double.
- *Emin_tail*: Energy interval that is considered below the lowest conduction sub-band (taken from both the left and the right reservoir). Default: 4e-3[eV]. Data type: Double.
- *EOffset*: In transmission calculations: total energy interval that is considered. The energy vector goes then from Emin which is the minimum energy of the system (automatically generated from the contact band structures and Emin_tail) to Emax=Emin+EOffset. In self-consistent simulations: energy interval that is added after the highest contact Fermi level. Default: 0.5 [eV]. Data type: Double.
- *dE_in*: Distance between two energy points right after the presence of a state with zero-velocity in the contact bandstructure. Default: 1e-3 [eV]. Data type: Double.
- *dE_f*: Distance between two energy points in regions where no singularity are present. Default: 5e-3 [eV]. Data type: Double.
- *dE_sep*: Energy distance between a channel turn-on and the first upcoming discretization point. Default: 1e-4 [eV]. Data type: Double.
- *NEmax*: In transmission calculations: maximum number of calculated energy points. Default: 100. Data type: Integer.
- *NVG*: Number of gate voltage points. Data type: Integer value greater or equal to 1. Default: 1.
- *Vgmin*: Initial gate voltage. Default: 0.0 [V]. Data type: Double.
- *Vgmax*: Final gate voltage. The bias points are ramped from *Vgmin* to *Vgmax* in steps of (*Vgmax*-*Vgmin*)/(*NVG*-1). Default: 0.0 [V]. Data type: Double.
- *NVS*: Number of source voltage points. Data type: Integer value greater or equal to 1. Default: 1.
- *Vsmin*: Initial source voltage. Default: 0.0 [V]. Data type: Double.

- *Vsmax*: Final source voltage. The bias points are ramped from *Vsmin* to *Vsmax* in steps of (*Vsmax*-*Vsmin*)/(*NVS*-1). Default: 0.0 [V]. Data type: Double.
- *NVD*: Number of drain voltage points. Data type: Integer value greater or equal to 1. Default: 1.
- *Vdmin*: Initial drain voltage. Default: 0.0 [V]. Data type: Double.
- *Vdmax*: Final drain voltage. The bias points are ramped from *Vdmin* to *Vdmax* in steps of (*Vdmax*-*Vdmin*)/(*NVD*-1). Default: 0.0 [V]. Data type: Double.
- *Vsub*: Potential of the substrate when unequal to the source potential. Default: 0.0 [V]. Data type: Double.

## 2.4 Material parameters

In this section the available material parameters are described. Directions: x, y, z. From hereon named XYZ to abbreviate all directions.

- *x*: Transport direction.
- *y*: First direction of confinement
- *z*: Second direction of confinement in 3D, open direction in 2D.
- *xy, yz, xz*: Cross directions. Only valid for the effective masses.

Device regions: wire, source, drain, gate, oxide. From hereon named DR to abbreviate all device regions.

- *wire*: Channel region of the device.
- *source*: Source extension of the device.
- *drain*: Drain extension if the device.
- *gate*: Gate contact of the device.
- *ox*: Oxide region of the device. Input as ox, ox1, ox2, ox3, ox4 possible, if there are several oxide regions.

Parameters:

- *AutoContact*: If 0, the position of the source and drain contacts can be freely chosen and specified in the structure definition. If 1 the contacts are located at the left (source) and right (drain) boundaries of the simulation domain. Default: 1.

- *n_of_valleys*: Number of valleys (band minima) to be accounted for. Default: 6. Data type: Integer.
- *deg_factor*: Degeneracy of each valley, e.g. [deg1 deg2 deg3]. Default: 0.0. Data type: Double.
- *valley_splitting*: Energy splitting of each valley. Can be used to take the Gamma and X valley at the same time and shift up the X valley, e.g. [split1 split2 split3]. Default: 0.0 [eV]. Data type: Double.
- *carrier_type*: Distinction between charge carriers, either ”e” for electrons or ”h” for holes, that are computed for each valley, e.g. [carrier1 carrier2 carrier3]. Default: e. Data type: Char.
- *Eg_*DR: Band gap of the DR region. Not directly used in QTSOLVER, but can be utilized in the definition of each material region, if necessary. Unit: in [eV]. Data type: Double.
- *Eps_*DR: Relative dielectric constant of the DR region. Not applicable to drain and source region. Unit: none. Data type: Double.
- *me_*XYZ_DR: Anisotropic effective mass of the DR region along the x,y,z direction. In the case of multiple valleys the parameter must be given for each valley separately, e.g. [me_valley1 me_valley2]. Unit: none. Data type: Double.
- *Xi_wire*: Semiconductor channel affinity. Used to set the potential at the gate. Unit: in [eV]. Data type: Double.
- *phi_m*: Gate work function. Used to set the potential at the gate, which is proportional to (*phi_m* - *Xi_wire*). Unit: in [eV]. Data type: Double.
- *dEc*: Parameter that can be used in the definition of each material region to set its band offset with respect to the reference 0 energy level, e.g. *dEc**(*Eg_*OX - *Eg_*WIRE). Unit: none. Data type: Double.
- *alpha_np*: Non-parabolicity factor of the bandstructure. Can be defined for each valley, e.g. [alpha1 alpha2 alpha3]. Default: 0.0. Data type: Double.
- *Virtual_Metal_CB*: Position of the virtual conduction band of the gate metal with respect to the 0 energy level. Must be defined when gate leakage is turned on. Unit: in [eV]. Data type: Double.
- *Virtual_Source_CB*: Position of the virtual conduction band of the source metal with respect to the 0 energy level. Must be defined when schottky contact is used at the source. Unit: in [eV]. Data type: Double.
- *Virtual_Drain_CB*: Position of the virtual conduction band of the drain metal with respect to the 0 energy level. Must be defined when schottky contact is used at the drain. Unit: in [eV]. Data type: Double.

The doping concentrations in the source and drain regions must be defined in order to determine their equilibrium Fermi level positions. Unit: in [$m^{-3}$]. Data type: Double.

- *ND_S*: Donor concentration in the source.
- *NA_S*: Acceptor concentration in source.
- *ND_D*: Donor concentration in the drain.
- *NA_D*: Acceptor concentration in drain.
- *ND_G*: Donor concentration in the gate.
- *NA_G*: Acceptor concentration in gate.

## 2.5 Device parameters

### 2.5.1 Assistance variables

In the following we show the valid assistance variables which are recognized by the parser that reads the command file. Hence, they can be used in the structure definition instead of writing a fixed value many times. All units are set to [nm].

- *Lext*: Extension length.
- *LextL*: Same as Lext.
- *LextR*: Same as Lext.
- *ts*: Source thickness.
- *tc*: Channel thickness.
- *td*: Drain thickness .
- *hc*: Source height (in 3D).
- *hc*: Channel height (in 3D).
- *hd*: Drain height (in 3D).
- *tox*: Oxide thickness. Also available as tox, tox1, tox2, tox3, tox4. These variables can be used to describe any oxide layer.
- *hox*: Oxide height. Also available as hox, hox1, hox2, hox3, hox4. These variables can be used to describe any oxide layer.
- *yc*: Y coordinate of the center point in a cylinder.

- *zc*: Z coordinate of the center point in a cylinder.
- *rs*: Source radius when it is cylindrical.
- *rc*: Channel radius when it is cylindrical.
- *rd*: Drain radius when it is cylindrical.
- *rox*: Oxide radius when it is cylindrical.
- *tcont*: Contact thickness.
- *hcont*: Contact height.
- *tground* : Ground thickness.
- *tsubstrate*: Substrate thickness.

### 2.5.2 Device variables

In comparison to the assistance variables the device variables must be set in order to define a device structure in QTsolver. The variables Lc, Ls, and Ld generate the initial guess for the electrostatic potential in self-consistent simulations. It is important that these values correspond to the length of the source, the drain, and the channel. Otherwise, the initial solution will not be generated accurately.
All coordinate and dimension parameters require the data type: Double.

**NOTE:** If a point belongs to two or more different regions defined in the input deck, it has always the properties of the region that was defined first.

- *NDim*: Device dimensions, 2 for UTB, 3 for Nanowires and FinFETs. Default: 3.
- *xmin*: Minimal x coordinate of the Finite Difference Method (FDM) grid. Unit: in [nm].
- *xmax*: Maximal x coordinate of the FDM grid. Unit: in [nm].
- *dx*: Homogeneous step size in x direction. Unit: in [nm].
- *ymin*: Same as *xmin* , but for the y coordinate. Unit: in [nm].
- *ymax*: Same as *xmax* , but for the y coordinate. Unit: in [nm].
- *dy*: Same as *dx* , but for the y coordinate. Unit: in [nm].
- *zmin*: Same as *xmin* , but for the z coordinate. Unit: in [nm].
- *zmax*: Same as *xmax* , but for the z coordinate. Unit: in [nm].

- *dz*: Same as *dx* , but for the z coordinate. Unit: in [nm].
- *Ls*: Source length. This variable must be defined and it can be used to define the structure dimensions. Unit: in [nm].
- *Lc*: Channel length. Same conditions as for *Ls.* Unit: in [nm].
- *Ld*: Drain length. Same conditions as for *Ls.* Unit: in [nm].
- *no_mat*: Number of regions that form the simulation domain (semiconductor+oxide+contacts). Data type: Integer. From here on each region is characterized by an index n with $1 \leq n \leq \text{N}$, where N is the total number of regions.
- *mat_type*(n) : Outer shape of the device element. Options: square or circle.
- *mat_trans*(n): If 0, transport is not considered in this region n and only Poisson equation is solved. Default: 1. Data type: Integer.
- *mat_coord*(n,#): Coordinates defining the boundaries of region n if *mat_type*(n) = square. The index # grows from 1 to $2^{NDim}$ where *NDim* is the number of dimensions, *NDim* = 2, or 3. Example in 3D:
  mat_coord(n,1) ⇒ [xmin ymin zmin]
  mat_coord(n,2) ⇒ [xmax ymin zmin]
  mat_coord(n,3) ⇒ [xmax ymax zmin]
  mat_coord(n,4) ⇒ [xmin ymax zmin]
  mat_coord(n,5) ⇒ [xmin ymin zmax]
  mat_coord(n,6) ⇒ [xmax ymin zmax]
  mat_coord(n,7) ⇒ [xmax ymax zmax]
  mat_coord(n,8) ⇒ [xmin ymax zmax]
- *mat_radius*(n): Radius of the front and back disks of the nth region if *mat_type*(n) = circle.
- *mat_Eg*(n): Band gap of region n, the variables *Eg_*DR can be reported here for simplicity and versatility.
- *mat_Eps*(n): Relative dielectric permitivity of region n, the variables *Eps_*DR can be reported here for brevity.
- *mat_me_*XYZ(n): Effective mass along the x,y,z direction in region n, the variables *me_*XYZ_DR can be reported here for brevity.
- *mat_V*(n): Conduction (electron) or valence (hole) band edge of region n. It is convenient to set *mat_V* to 0 in the semiconductor channel and set it to the band offset with respect to the reference level in other regions, e.g. *mat_V* = *dEc**(*Eg_ox1* - *Eg_wire*) in an oxide or *mat_V* = *Virtual_Metal_CB* in the gate region.

- *mat_contact*(n): If the region n is a contact, define its type. If 1, the potential is fixed and no open boundary conditions (OBC) are applied. If 2, the potential is fixed and OBCs are possible provided that the region has a finite width. If 3, the potential is fixed and the region is considered as the substrate. Additionally the contact direction can be defined, e.g. [type direction], where the charge carrier injection direction defines the direction: x=1, -x=-1, y=2,-y-2, z=3, -z=-3. Default: 0 (no contact).

- *mat_schottky*(n): If 1, the region n is a Schottky instead of ohmic contacts. This option can only be set to 1 in the source and drain extensions, together with *mat_V* = *Virtual_YY_CB* where *YY* represents source or drain. Default: 0.

- *mat_schottky_barrier*(n): Defines the Schottky barrier height in region n (only source and drain).

- *mat_ND*(n): Homogeneous donor doping concentration in region n, the variable *ND_*DR can be reported here.

- *mat_NA*(n): Homogeneous acceptor doping concentration in region n, the variable *NA_*DR can be reported here.

- *mat_bound*(n): Not used.

## 2.6 The solver commands

In the bottom part of the input deck the user defines the computation type and the solver to accomplish the task. The available options are presented here.

- *command (#)*: Task that the simulator must accomplish. Possible options are:

  1. **Bandstructure** - Conduction/valence bandstructure of contacts,
  2. **Transmission_***solver* - Conduction/valence band transmission and DOS,
  3. **SC_***solver* - Self-consistent electron/hole simulation,

  The *solver* option depends on the solver used to compute the transport problem: PWF - Paradiso, UWF - Umfpack, SWF - SuperLU_dist and MWf - MUMPS. Examples are shown in Chapter 4. (#) grows from 1 to the total number of commands. A maximum of 20 commands for a given structure configuration is possible.

# Chapter 3

# Output files

In this chapter the output data generated by QTsolver is presented. Depending on the solver used to compute the transport or transmission problem the naming of the output data is modified. The following options are available: P - Paradiso, U - Umfpack, S - SuperLU_dist and M - MUMPS. From here on PREFIX refers to the selected solver. Additionally, V to the valley index, NG, NS, ND to the NG$^{th}$ gate bias, NS$^{th}$ source bias, and ND$^{th}$ drain bias index, respectively and C to the contact index (0: source, 1: drain, 2 - total number of contacts: gate). All the indices start at 0.

## 3.1 Bandstructure calculations

These output files are generated during the calculation of the electron/hole bandstructure. In the command file *command (#)* = Bandstructure.

- *Ekl_*V*.dat*: Bandstructure of the left contact for each valley V. Stored as [n_of_modes×Nk]. Unit: [eV].
- *Ekr_*V*.dat*: Bandstructure of the right contact for each valley V. Stored as [n_of_modes×Nk]. Unit: [eV].
- *k.dat*: Wavevector k. Stored as [1×Nk]. Unit: 1/nm.

## 3.2 Transmission and density-of-states

These output files are generated within the calculation of the electron transmission probability and the corresponding density-of-states when a flat-band potential is assumed throughout the device. In the command file *command (#)* = Transmission_*solver*.

- PREFIX_*ZE*_V_C*.dat*: The density-of-states (DOS) for valley V and contact C. The DOS is summed over the device cross section (y in 2D structures, y and z in 3D structures). Stored as [N$_x$×NE]. Unit: [1/eV].

- PREFIX_*TE*_V_C.*dat*: The transmission probability for valley V and contact C. The transmission probability is from point $x_i$ to $x_{i+1}$ and is stored at location TE(i,:) as a function of the energy. Stored as [$N_x$×NE]. Unit: [1].

- PREFIX_*E*_V.*dat*: Energy vector at which ZE and TE are calculated for valley V. Stored as [1×NE]. Unit: [eV].

## 3.3 Self-consistent simulations

These output files are generated during self-consistent transport simulation. In the command file *command (#)* = SC_*solver*.

- *Ekl*_NG_NS_ND_ V_C.*dat*: Bandstructure of the left contact for valley V, gate bias NG, source bias NS, drain bias ND, and contact C.. Stored as [n_of_modes×Nk]. Unit: [eV].

- *Ekr*_NG_NS_ND_ V_C.*dat*: Bandstructure of the right contact for valley V, gate bias NG, source bias NS, drain bias ND, and contact C.. Stored as [n_of_modes×Nk]. Unit: [eV].

- PREFIX_*ZE*_NG_NS_ND_ V_C.*dat*: The density-of-states (DOS) for valley V, gate bias NG, source bias NS, drain bias ND, and contact C. Stored as [$N_x$×NE]. Unit: [1/eV].

- PREFIX_*TE*_NG_NS_ND_V_C.*dat*: The transmission probability for valley V, gate bias NG, source bias NS, drain bias ND, and contact C. Stored as [$N_x$×NE]. Unit: [1].

- PREFIX_*IdE*_NG_NS_ND_V.*dat*: Spectral current along the transport direction and per valley V, gate bias NG, source bias NS and drain bias ND. Stored as [NE×$N_x$]. Unit: $\mu$A/($\mu$m·eV (2D), A/eV (3D).

- PREFIX_*E*_NG_NS_ND_V.*dat*: Energy vector at which ZE and TE are calculated for valley V, gate bias NG, source bias NS, and drain bias ND. Stored as [1×NE]. Unit: [eV].

- PREFIX_*Vpot.dat*: Electrostatic potential energy for all the bias points after the Schrodinger-Poisson iterations have converged. Stored as [(NG×NS×ND)×Ngrid] with *Ngrid* being the total number of discretization points. Unit: eV.

- PREFIX_*Vpot_act.dat*: Latest Electrostatic potential energy that QTSOLVER computed. Stored as [Ngrid×1]. Unit: eV.

- PREFIX_*Vg_Vs_Vd.dat*: Gate, source, and drain bias successfully computed. Stored as [(NG×NS×ND)× 3]. Unit: V.

- PREFIX_*nd.dat*: Electron density at each discretization point and for each bias point. Stored as [(NG×NS×ND)× Ngrid]. Unit: [1/m$^3$].
- PREFIX_*nd_act.dat*: Last electron density at each discretization point and for each bias point that QTSOLVER computed. Stored as [Ngrid×1]. Unit: [1/m$^3$].
- PREFIX_*Ixyz*_NG_NS_ND_.*dat*: Current density along the x,y and z directions for all the grid points *Ngrid*. Stored as [3×Ngrid]. Unit: $\mu$A/$\mu$m (2D), A (3D).
- PREFIX_*Ixyz_act.dat*: Last current density along the x,y and z directions for all the grid points *Ngrid* that QTSOLVER computed. Stored as [3×Ngrid]. Unit: $\mu$A/$\mu$m (2D), A (3D).
- PREFIX_*Ig.dat*: Gate leakage current along the transport direction at each bias point. This file contains the total current from $x_i$ to $x_{i+1}$. Stored as [(NG×NS×ND)×$N_x$]. Unit: $\mu$A/$\mu$m (2D), A (3D).
- PREFIX_*Ig_act.dat*: Last gate leakage current along the transport direction that QTSOLVER computed. Stored as [$N_x$×1]. Unit: $\mu$A/$\mu$m (2D), A (3D).
- PREFIX_*Id.dat*: Drive current along the transport direction at each bias point. Due to current conservation the values at each discretization point should be the same or compensated by the gate leakage current. This file contains the total current from $x_i$ to $x_{i+1}$. Stored as [(NG×NS×ND)×$N_x$]. Unit: $\mu$A/$\mu$m (2D), A (3D).
- PREFIX_*Id_act.dat*: Last drive current along the transport direction. Stored as [$N_x$×1]. Unit: $\mu$A/$\mu$m (2D), A (3D).
- PREFIX*grid.dat*: The first three columns contain the x, y, and z coordinates, the next 3 the corresponding indices, i.e. $\frac{x}{dx}-1$, $\frac{y}{dy}-1$, and $\frac{z}{dz}-1$. The seventh column indicates whether the Schrödinger and Poisson equations are solved at the considered grid point (1) ore only the Poisson equation (0). Stored as [7×Ngrid]. Units: Column 1-3: nm, column 4-7: Integer.
- PREFIX*gate.dat*: Index of the grid points corresponding to the gate contact, $\text{Ngrid}_{gate}$. Stored as [$\text{Ngrid}_{gate}$×3]. where Unit: nm.
- PREFIX*doping.dat*: Doping concentration at each grid point. Stored as [Ngrid×1]. Unit: 1/m$^3$.
- PREFIX*condition.dat*: Relative error of the electrostatic potential energy after each Schrödinger-Poisson iteration. Stored as [$N_{Poiss,iter}$×1] with $N_{Poiss,iter}$ being the total number of poisson iterations.
- PREFIX_*Id*_V.*dat*: Drive current along the transport direction for valley V and for all the bias points. Stored as [(NG×NS×ND)×$N_x$]. Unit: $\mu$A/$\mu$m (2D), A (3D).

- PREFIX_*Id*_V_*act.dat*: Last drive current along the transport direction for valley V. Stored as [$N_x \times 1$]. Unit: $\mu$A/$\mu$m (2D), A (3D).

- PREFIX_*nd*_V.*dat*: Electron density for valley V at each discretization point and for each bias point. Stored as [(NG×NS×ND)× Ngrid]. Unit: [1/m$^3$].

- PREFIX_*nd*_V_*act.dat*: Last electron density for valley V at each discretization point. Stored as [Ngrid×1]. Unit: [1/m$^3$].

- PREFIX_*lambda*_NG_NS_ND_V.*dat*: In mode-space the Eigenenergies ($N_m$ modes) are given along the transport direction $x$ for valley V, gate bias NG, source bias NS and drain bias ND. Stored as [$N_m \times N_x$]. Unit: [eV].

- PREFIX_*nd*_*ms*_NG_NS_ND_V.*dat*: Mode-resolved electron density along the transport direction for valley V, gate bias NG, source bias NS and drain bias ND. Stored as [$N_m \times N_x$]. Unit: [1/m$^2$] (2D), [1/m] (3D).

- PREFIX_*Id*_*ms*_NG_NS_ND_V.*dat*: Mode resolved current for valley V, gate bias NG, source bias NS and drain bias ND. Stored as [$N_x \times 1$]. Unit: $\mu$A/$\mu$m (2D), A (3D).

# Chapter 4

# Examples

In this chapter we give an overview on how to use QTsolver to perform different device simulations. Three examples are chosen to illustrate the solver functionality: a 2D Silicon double gate transistor, a 3D Germanium FinFET and a 3D Silicon Nanowire. All devices can be found in the /Example folder that is distributed with the QTsolver source code and can be launched without modification. The input parameters are summarized in Chapter 2 whereas the output files are explained in Chapter 3. Since QTsolver does not include a visualization tool the simulation results are plotted with MATLAB scripts that are also included in the /Example folder under /view_scripts.

## 4.1 2D Silicon double-gate FET

### 4.1.1 Device geometry

This example shows a Silicon planar double-gate (DG) transistor. The transport direction is along the <100> crystal orientation (x-axis), whereas the confined direction is aligned with the y-axis. The structure of the transistor is shown in Figure 4.1. It consists of a Silicon ultra-thin-body embedded between a top and bottom gate stack made of Hafnium dioxide and a gate contact around it. The structure has a total length of 52 nm while the thickness of the semiconducting channel is 5 nm. The source and drain contact are located on the left and right side of the FET.

When constructing the device structure in the input deck (Si_DG_100.cmd) the device geometry is divided into 9 regions. Figure 4.2 shows the schematic of the transistor. The numbered areas form a wire (1,2,3), a top and bottom insulator (4,5 and 6,7) and a top and bottom gate (8,9). Source and drain contacts are not defined as a separate region. They are automatically placed on the left and right side of the device.

**NOTE:** Here region 5 includes region 4, but since region 4 is defined first, a point situated at the cross location in Figure 4.2 will have properties of region 4, as derived.

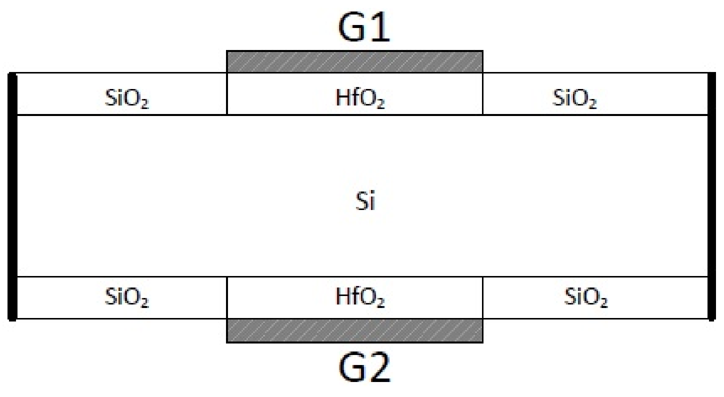


Figure 4.1: **Structure of the considered Si planar double-gate FET.** The device consists of a Silicon UTB plated on the gate top and bottom with HfO$_2$, on the source and drain extensions top and bottom with SiO$_2$ and a gate contact. The source and drain contacts are located on the left and right side of the FET.

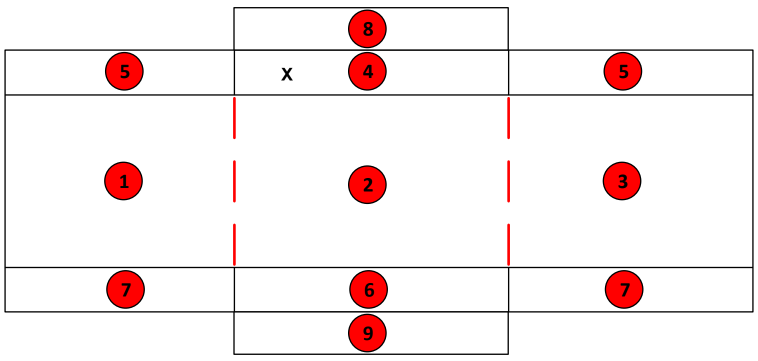


Figure 4.2: **The planar double-gate FET as constructed in** QTsolver**.** The device is divided into 9 region where areas (1,2,3) form the semiconducting part (1 and 3 are doped, not 2), (4,5) the top oxide, (6,7) the bottom and (8,9) the two gate contacts.

### 4.1.2 Bandstructure calculation

To perform a bandstructure calculation the input file must include the line *command(1)=Bandstructure;* in the command section. The simulation takes less than 5 seconds on 1 CPU to compute the simplified parabolic bandstructure of the Silicon DG using the Arpack solver.
For visualization the matlab function *getband.m* can be used. It plots the conduction bands of both the drain and source contact where 0 eV usually refers to the bulk conduction band edge. Figure 4.3 shows the resulting bandstructure for the considered ultra-thin body structure.

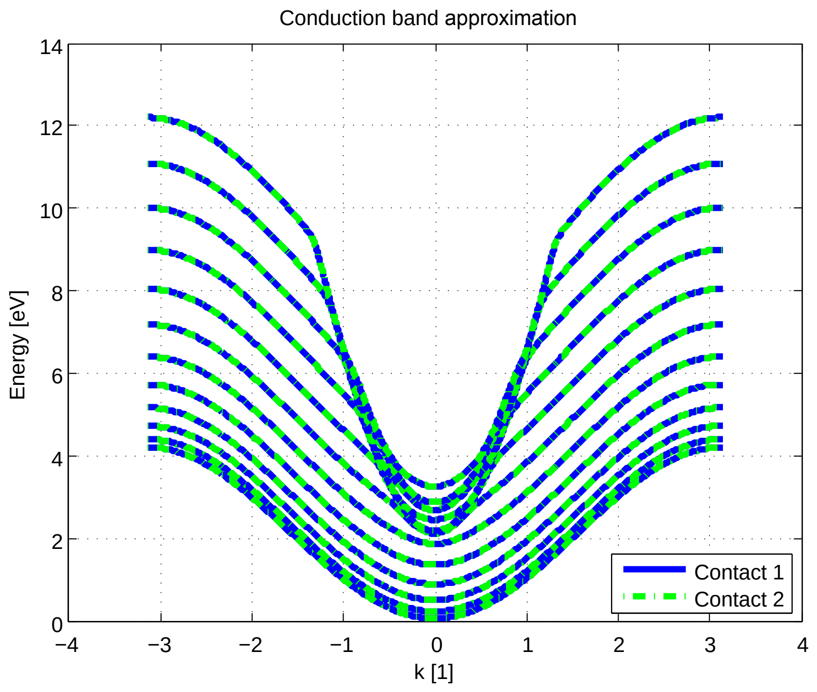


Figure 4.3: **Conduction band structure of the 2D DG FET shown in Figure 4.1.** The conduction band is plotted for both the left and right contact where 0 eV marks the bulk conduction band edge.

### 4.1.3 Flat-band condition

To evaluate the flat-band condition (transmission and density-of-states (DOS)) of the given device the input file needs to include *command(i)=Transmission_UWF;* in the command section. UWF represents the Umfpack solver and can be replaced by SWF (SuperLU_DIST Solver), PWF (Paradiso solver) or MWF (MUMPS solver).
For visualization the matlab function *checkZETE.m* can be used. It plots both, the source and drain transmission and density-of-states under assumption of a flat

potential profile. Figures 4.4 and 4.5 show the resulting plots for this Silicon <100> double gate transistor.

### 4.1.4 Self-consistent transport simulation

The I-V characteristics of the Si DG transistor can be computed through a Schrödinger-Poisson self-consistent simulation. Source, drain and gate biases can be set in the input command file and swept to determine the output/transfer characteristics. Only ballistic transport within the effective mass approximation is available. In the example the gate voltage is swept from -0.1 to 0.6 V in 15 steps while the source (Vs=0 V) and drain (Vd=0.7 V) biases remain constant (transfer characteristics). The complete transport simulation takes between 29 min and 1 hour on 1 CPU for mode-space and real-space computation, respectively.

#### 4.1.4.1 Real-space

As default, the transport simulation is done in real space. The y direction is explicitly discretized and the current density and electron density are computed on the 2D grid.

#### 4.1.4.2 Mode-space

By turning on the input parameters *mode_space* and *recompute_basis* in the .cmd file the transport calculations are done in mode-space. A significant decrease of the simulation time is obtained as a consequence, especially in 3D.

#### 4.1.4.3 Results

For both real-space and mode-space calculations, the results are identical and presented in Figure 4.6 (transfer characteristics) and Figures 4.7 and 4.8 (electron density distribution and electrostatic potential energy).

To write the transmission and density-of-states (DOS) quantities to a file the input deck needs to include *plot_transmission=1;* and *plot_dos=1;* in the Parameters section. For visualization the matlab function *gettransdos.m* can be used. It plots both the source and drain transmission and DOS. Figures 4.9 and 4.10 show the results for this Silicon <100> double gate transistor.

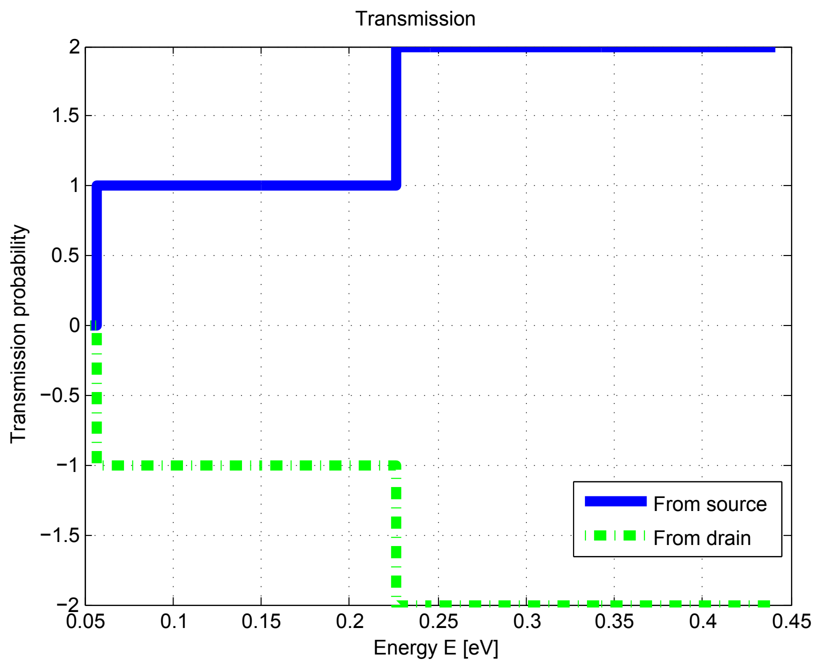


Figure 4.4: **Transmission probability through the 2D DG FET without any applied bias potentials (flat-band condition).** The transmission from left to right (solid line) is positive, while the transmission from right to left (dashed line) is negative.

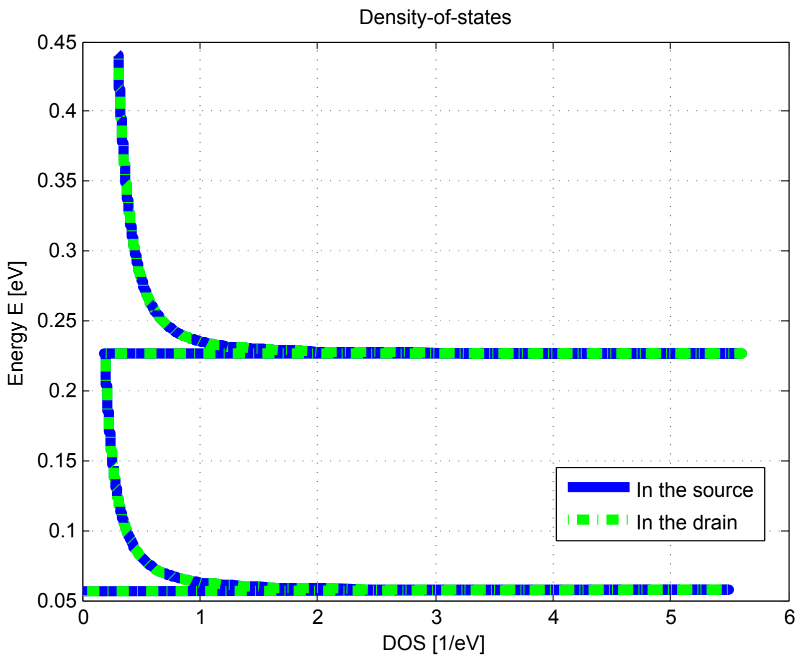


Figure 4.5: **Density-of-states of the 2D DG FET in the source and drain regions at flat-band condition.**

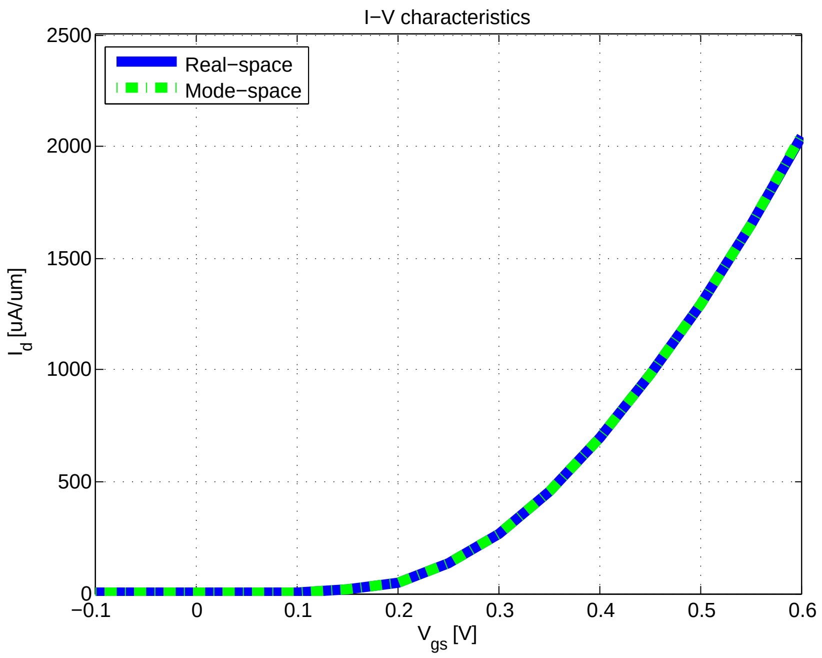
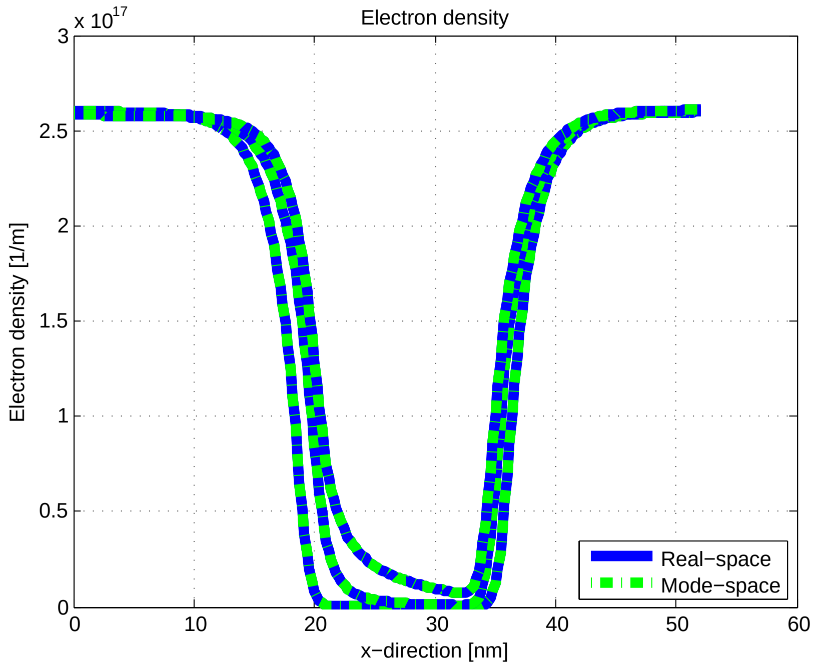


Figure 4.6: **Transfer characteristics Id-Vgs at Vds=0.7 V of the 2D double gate transistor shown in Figure 4.1.** Both, the real space (solid line) and mode-space (dashed line) results are shown.



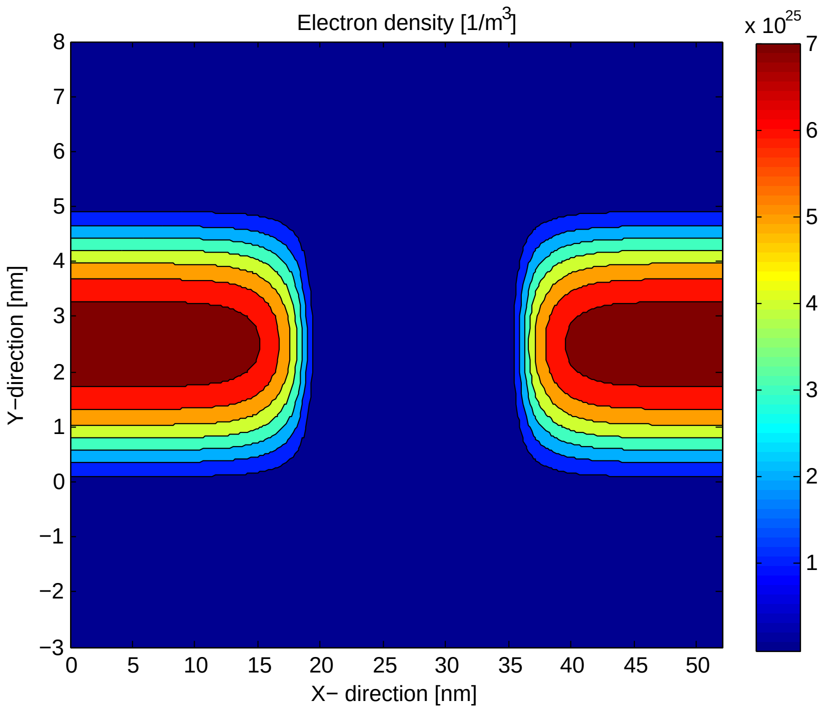
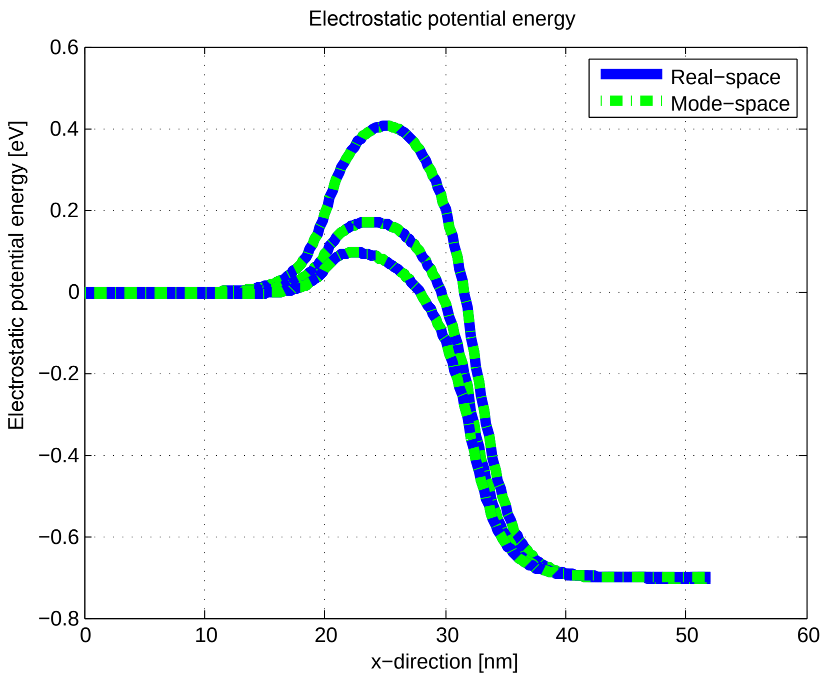


Figure 4.7: **Top: Electron density of the 2D double gate transistor shown along the transport direction at different Vgs.** Both the real space and mode-space results are shown. The charge density is integrated over the direction of confinement (y). **Bottom: 2-D contour of the electron density.** Both plots at Vgs=0.6 V and Vds=0.7 V.



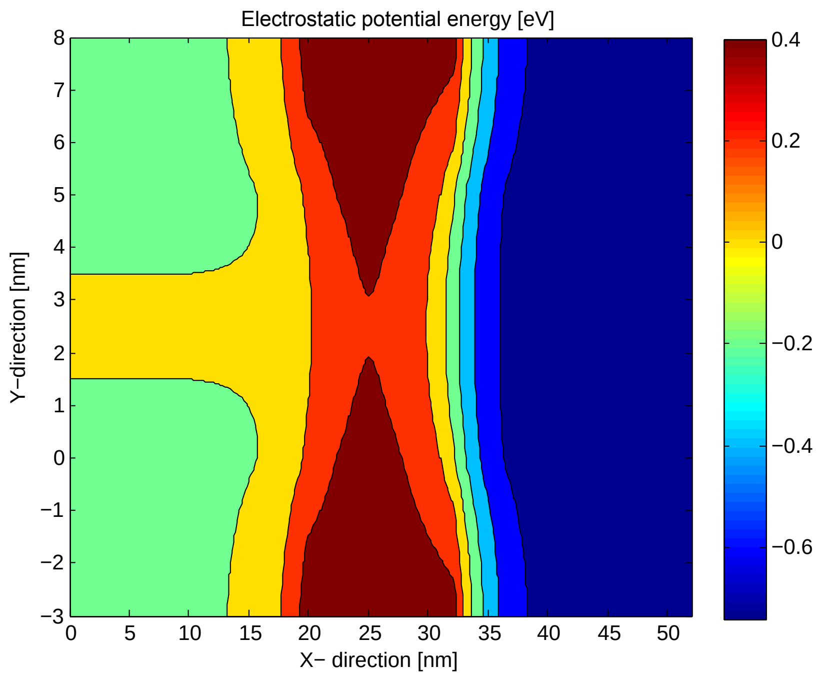


Figure 4.8: **Top: Average electrostatic potential energy along the transport direction at different Vgs in real-space and mode-space. Bottom: 2-D contour plot of the electrostatic potential energy at Vgs=0.6 V and Vds=0.7 V.**

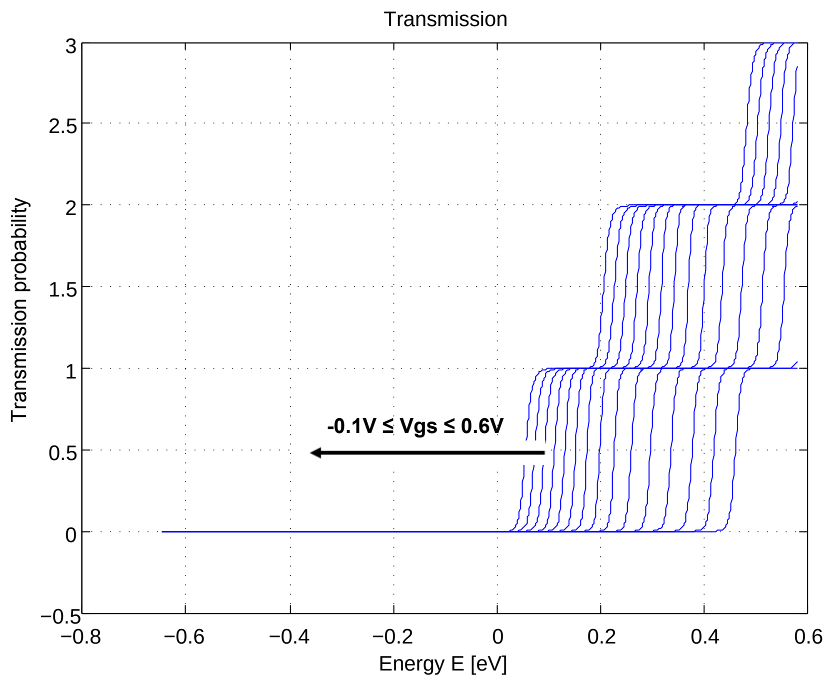
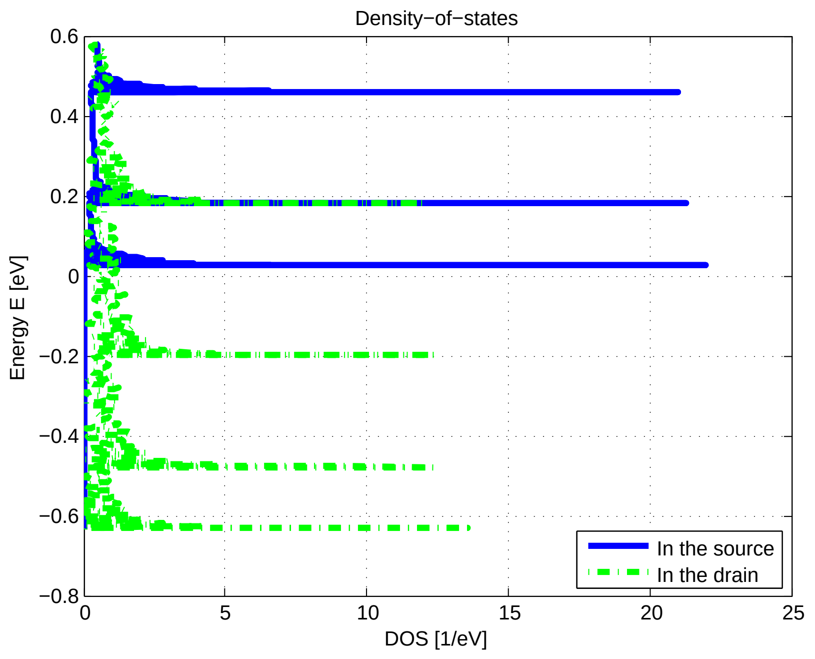


Figure 4.9: **Transmission probability through the transistor shown in Figure 4.1 at different Vgs.**



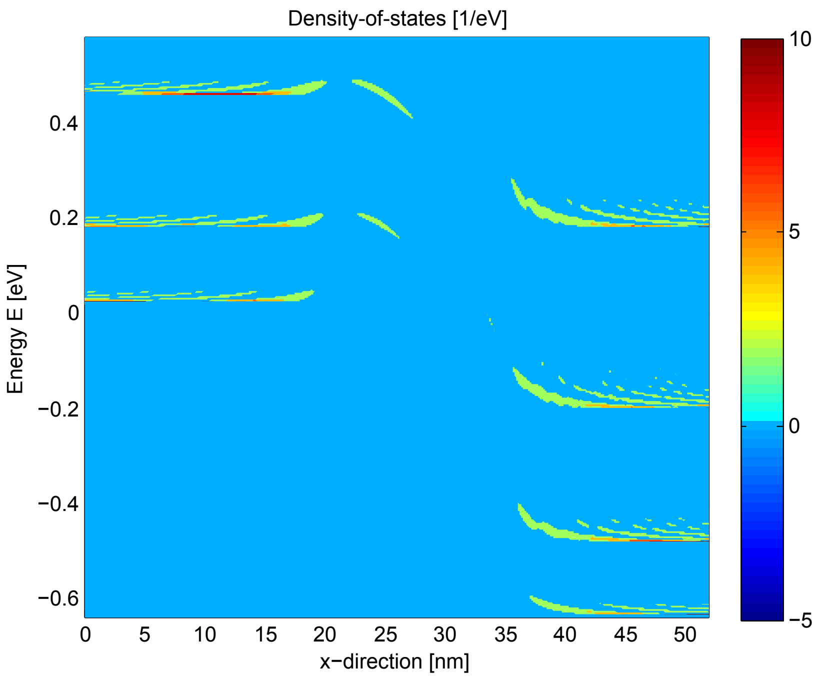


Figure 4.10: **Top: Energy resolved density-of-states in the source and drain region of the Si 2D DG FET. Bottom: Spectral DOS at Vgs=0.6 V and Vds=0.7V.**

## 4.2 3D Germanium FinFET

### 4.2.1 Device geometry

The second example deals with a Germanium FinFET . The transport direction is along the <100> crystal orientation and defines the x-axis, whereas the confined directions are along the transverse y-axis and the vertical z-axis. Since the transport direction does not align with the principle axes of one of the 8 L-valley minima of Ge, a full effective mass tensor including the *mx, my, mz* as well as the cross terms *mxy, myz, mzx* must be defined in the input deck. The structure of the transistor is shown in Figure 4.11. It consists of a $SiO_2$ substrate with a Germanium fin extension on top forming a 52 nm long channel with a 5 x 5 $nm^2$ cross-section. On the left, top, and right the fin is covered with a 3 nm thin layer of Hafnium dioxide. The gate is contacted threefold (top,left, right) to the fin. The source and drain contact are located on the left and right side of the FET.

When constructing the device structure in the Ge_FinFET_100.cmd command file the device geometry in divided into 16 parts. Figure 4.12 shows the schematic of the transistor. The numbered areas form a wire (1,2,3), the gate insulator (4,5,6), a thin source and drain oxide layer (7,8,9), the substrate (10,11), a second thicker oxide layer the channel (12), a second thicker oxide layer around the source and drain (13) and gate connection on top, left, and right (14,15,16).

### 4.2.2 Bandstructure calculation

To perform a bandstructure calculation the input file must include the line *command(1)=Bandstructure;* in the command section. The simulation takes less than 10 seconds on 1 CPU to compute the simplified parabolic bandstructure of the Germanium FinFET using the Arpack solver.
For visualization the matlab function *getband.m* can be used. It plots the conduction bands of both the drain and source contact where 0 eV usually refers to the bulk conduction band edge. Figure 4.13 shows the resulting bandstructure for the considered 3D structure.

### 4.2.3 Flat-band condition

To evaluate the flat-band condition and obtain the transmission and density-of-states (DOS) of the given device the input file needs to include *command(i)=Transmission_UWF;* in the command section. UWF represents the Umfpack solver and can be replaced by SWF (SuperLU_DIST Solver), PWF (Paradiso solver) or MWF (MUMPS solver). For visualization the matlab function *checkZETE3D.m* can be used. It plots both the source and drain transmission and density-of-states under the assumption of a flat potential profile. Figures 4.14 and 4.15 show the resulting plots for this Germanium <100> FinFET.

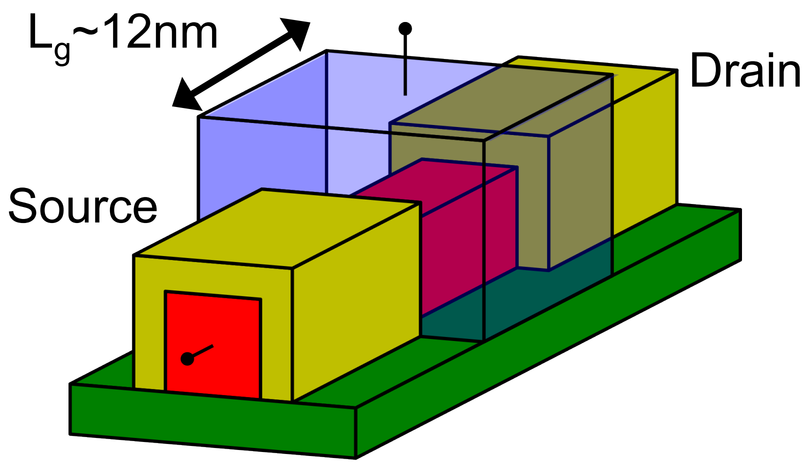
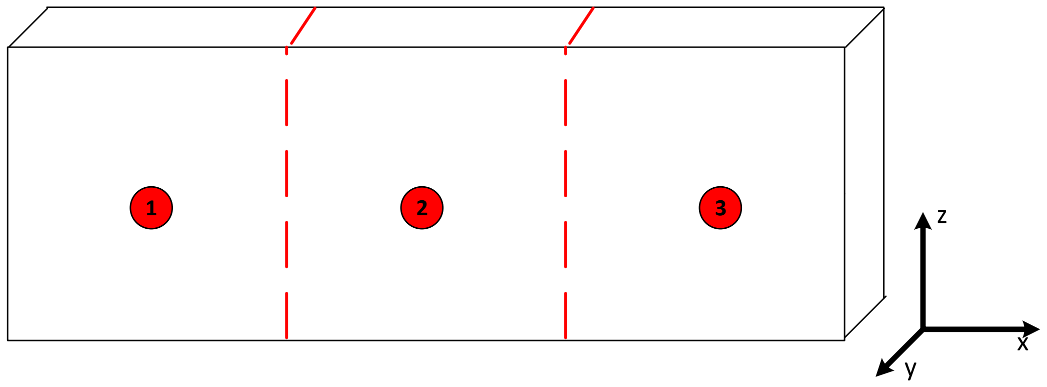


Figure 4.11: **Structure of the considered Germanium <100> FinFET.** It consists of a silicon oxide bulk with a Germanium fin extension on top forming a 52 nm channel in length with a cross-section of 5 x 5 nm$^2$. On the left, top and right side the fin is covered with a 3 nm thin layer of Hafnium dioxide. Additionally, a spacer surrounds the complete fin with Hafnium oxide. The gate is contacted threefold (top, left, right) to the fin. The source and drain contact are located at the left and right side of the FET.

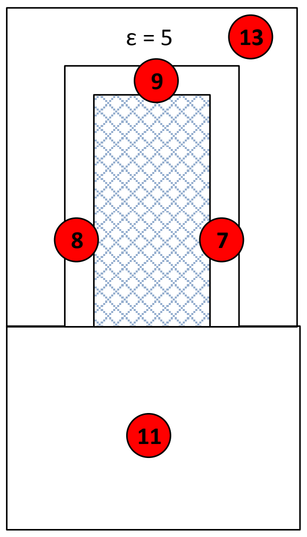
### 4.2.4 Self-consistent transport simulation

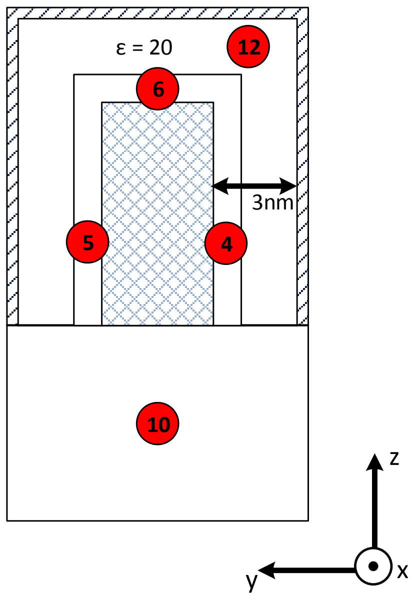
The I-V characteristics of the Ge FinFET transistor can be computed through a Schrödinger-Poisson self-consistent simulation. A source, drain and gate bias can be selected in the input command file and swept to determine the output/transfer characteristics. Only ballistic transport within the effective mass approximation is available. In the example the gate voltage is swept from -0.1 to 0.1 V in 5 steps while the source (Vs=0 V) and drain (Vds=0.7 V) biases remain constant (transfer characteristics). The complete transport simulation takes between 20 min and 3 days per bias point on 1 CPU for mode-space and real-space computation, respectively. For both simulation approaches, the resulting entities are equal and are presented in Figure 4.16 showing the transfer characteristics and Figures 4.17 and 4.18 showing the electron density distribution and the potential energy, respectively.

To obtain the transmission and density-of-states (DOS) of the device the input file needs to include *plot_transmission=1;* and *plot_dos=1;* in the parameters section. For visualization the matlab function *gettransdos3d.m* can be used. It plots both the source and drain transmission and DOS. Figures 4.19 and 4.20 show the resulting plot for the Germanium <100> FinFET.



(a)





(b) Source / Drain

(c) Channel

Figure 4.12: **The Germanium <100> FinFET as constructed in** QTsolver. The device is divided into 16 regions where (1,2,3) form the semiconducting part (1 and 3 doped, 2 not), (4,5,6) the gate insulator, (7,8,9) a thin source and drain oxide layer, (10,11) the substrate, (12) a second thicker oxide layer the channel, (13) a second thicker oxide layer around the source and drain, and (14,15,16) gate connection on top, left, and right (upward diagonal patter on top).

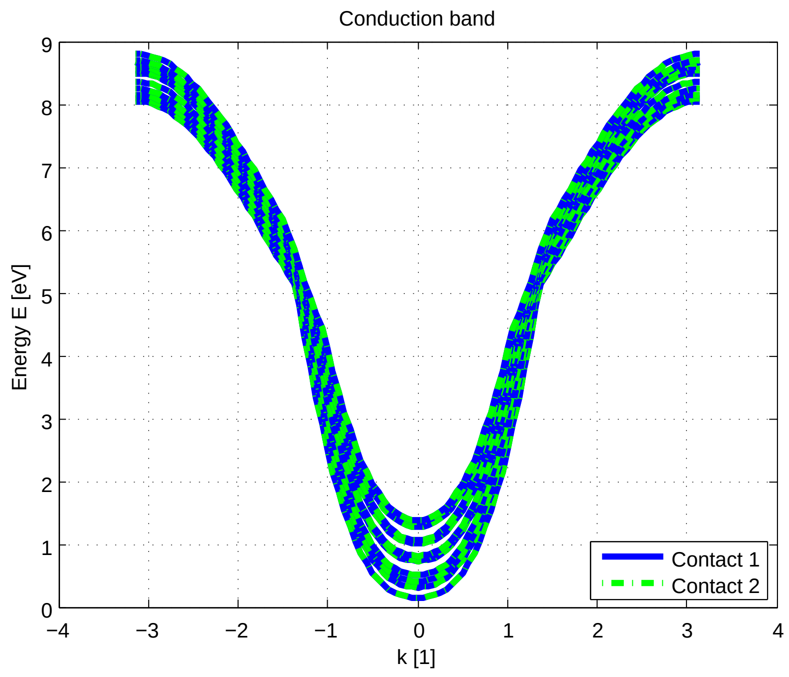


Figure 4.13: **Conduction band structure of the Germanium <100> FinFet.** The conduction band is plotted for both the left and right contacts where 0 eV marks the conduction band edge.

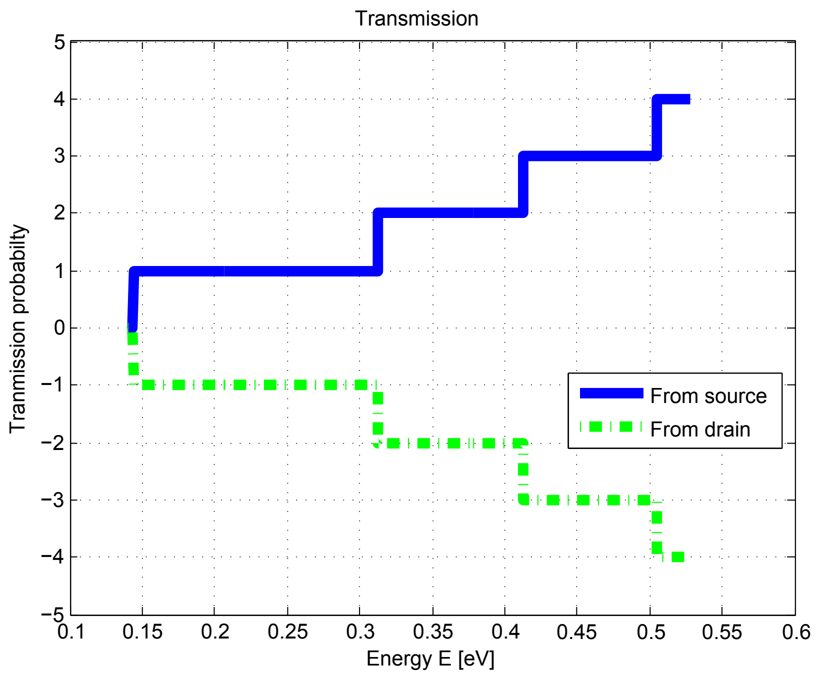


Figure 4.14: **Transmission probability through the 3D FinFET without any applied bias potentials (flat-band condition).** The transmission from left to right (solid line) is positive, while the transmission from right to left (dashed line) is negative.

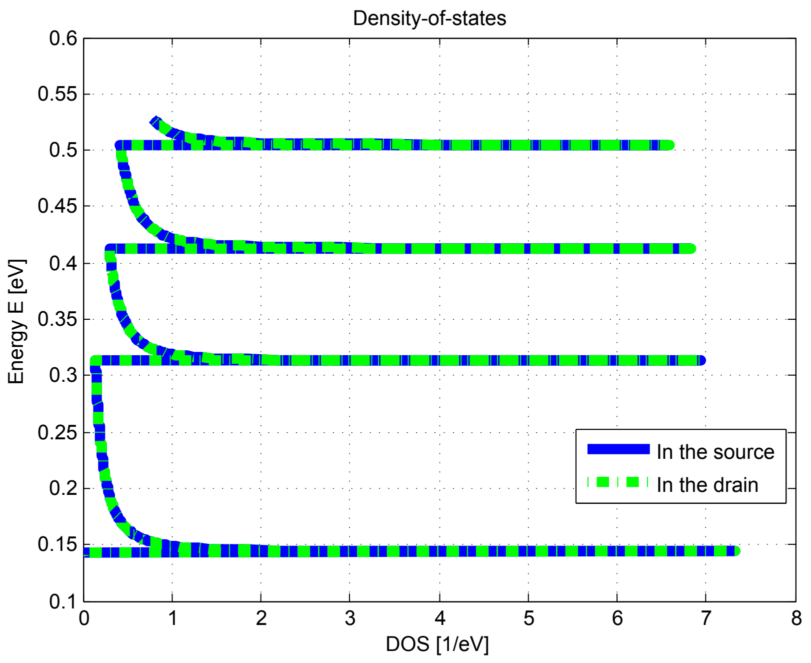


Figure 4.15: **Density-of-states of the 3D FinFET in the source and drain regions at flat-band condition.**

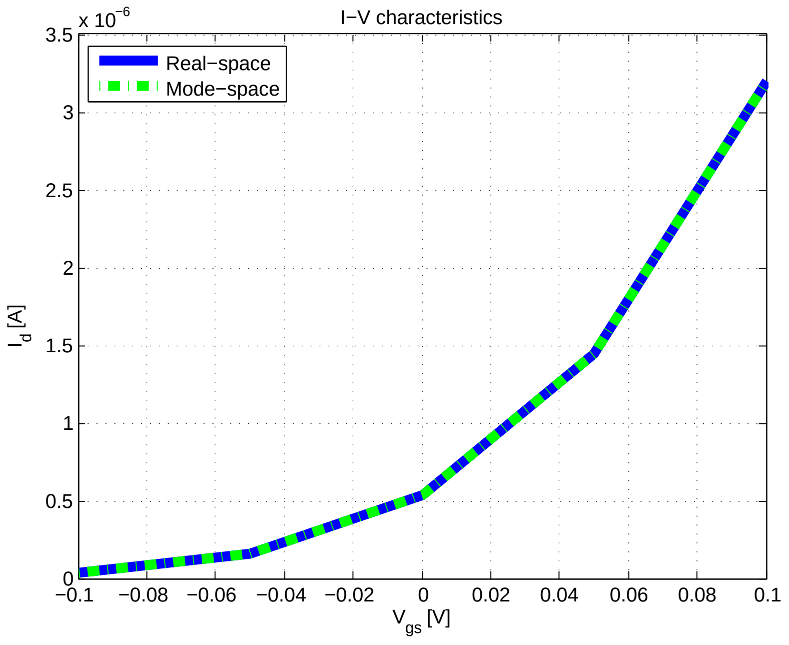
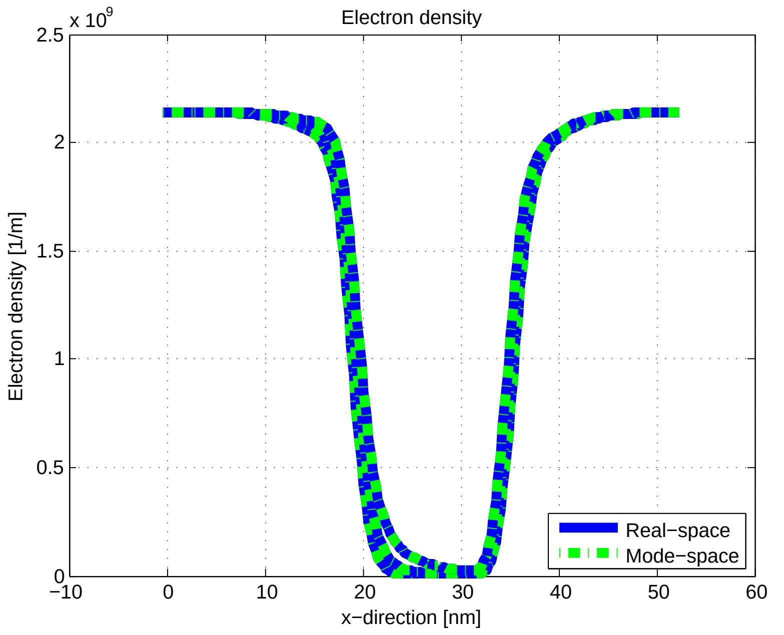


Figure 4.16: **Transfer characteristics Id-Vgs at Vds=0.7 V of the 3D Fin-FET shown in Figure 4.11.** Both, the real space (solid line) and mode-space (dashed line) results are shown.



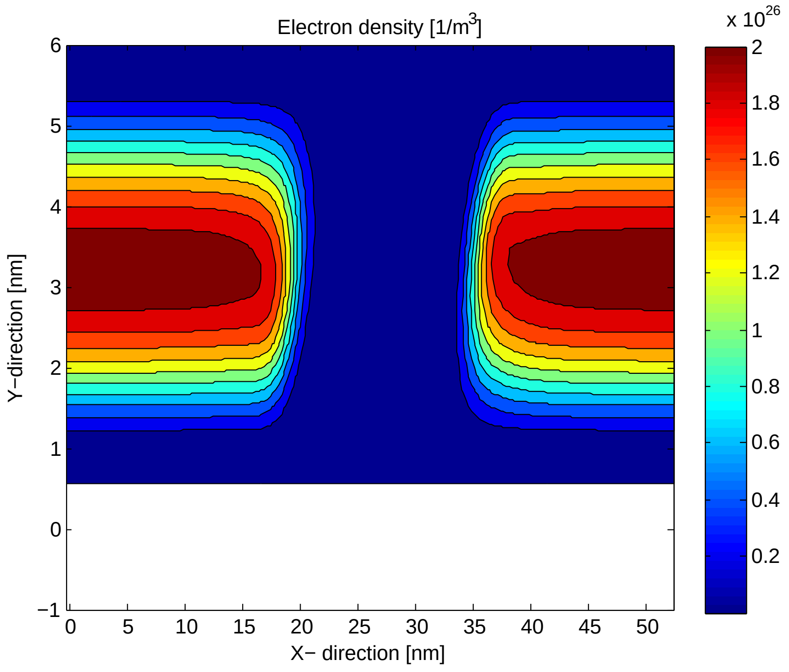
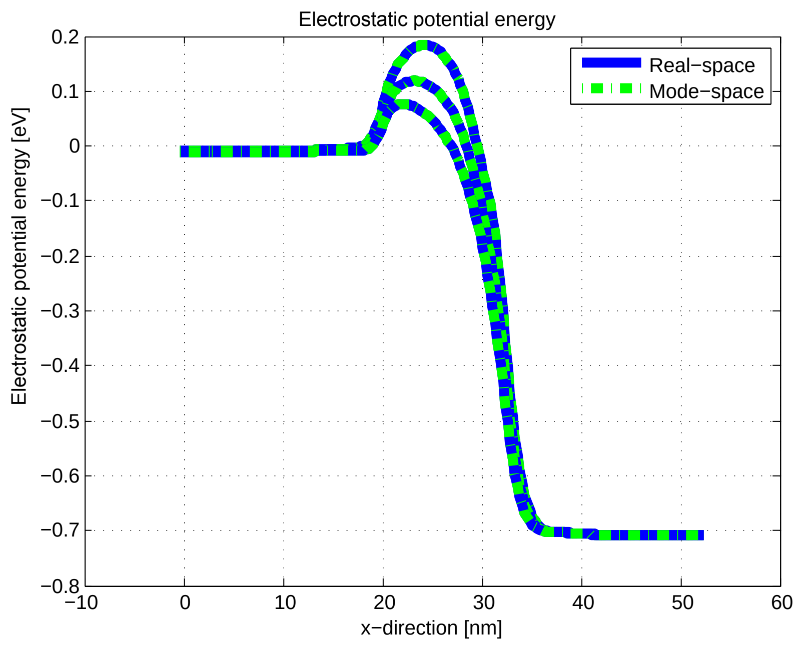


Figure 4.17: **Top: Electron density of the 3D FinFET shown along the transport direction at different Vgs and Vds=0.7 V.** Both the real space and mode-space results are shown. The charge density is integrated over the directions of confinement (y and z). **Bottom: 2-D contour of the electron density at the Vgs=0.1 V and Vds=0.7 V shown at z=2.5 nm.**



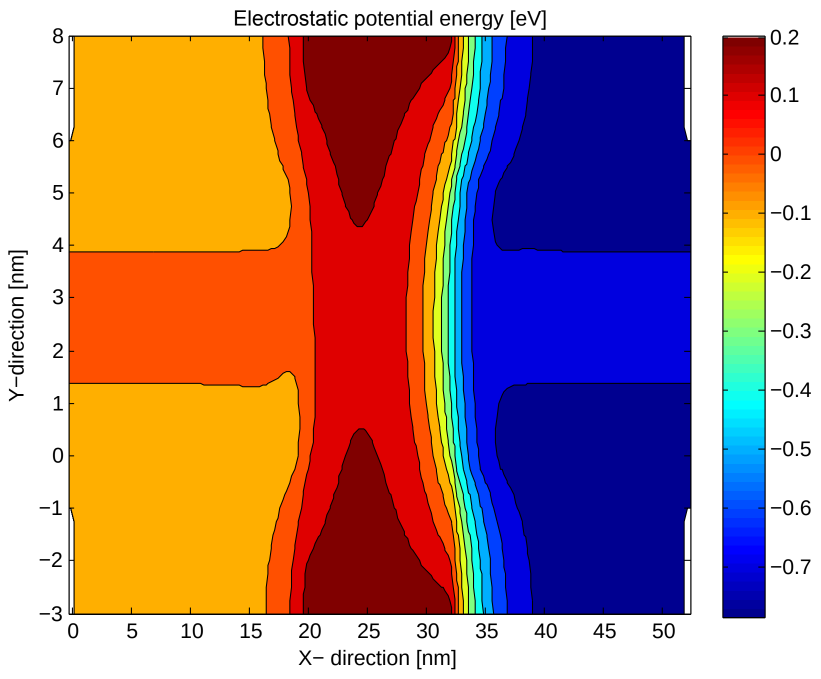


Figure 4.18: **Top: Average electrostatic potential energy along the transport direction at different Vgs in real-space and mode-space. Bottom: 2-D contour plot of the electrostatic potential energy at Vgs=0.1 V and Vds=0.7 V shown at z=2.5 nm.**

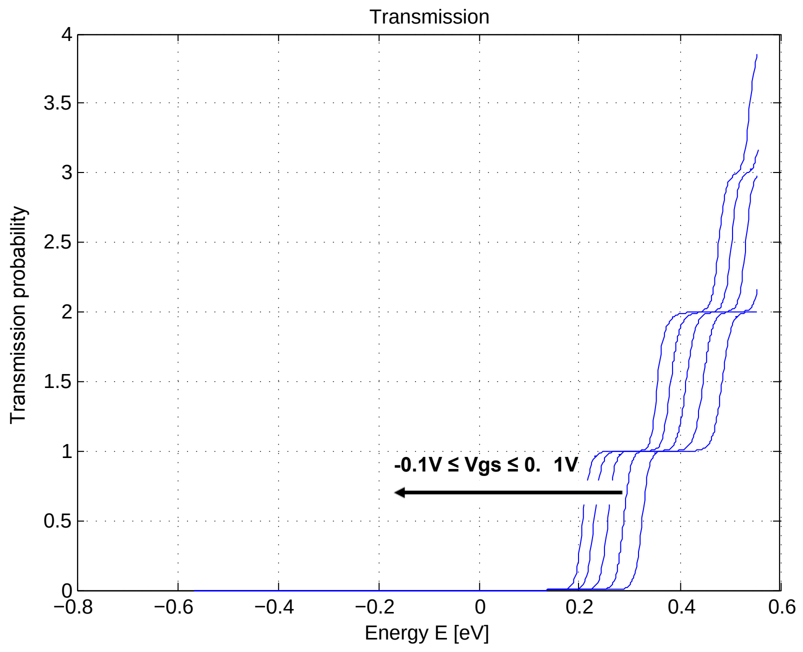
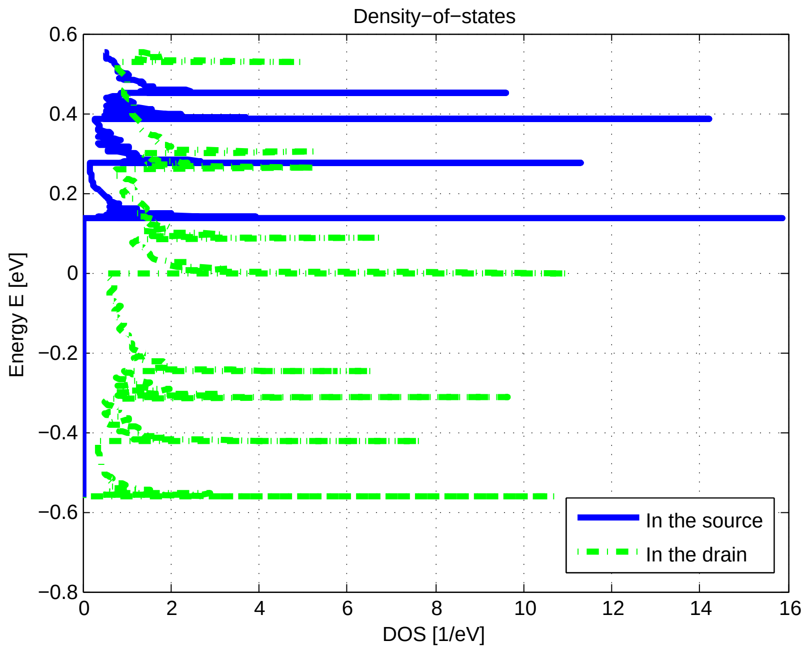


Figure 4.19: **Transmission probability through the transistor shown in Figure 4.11 at different Vgs.**



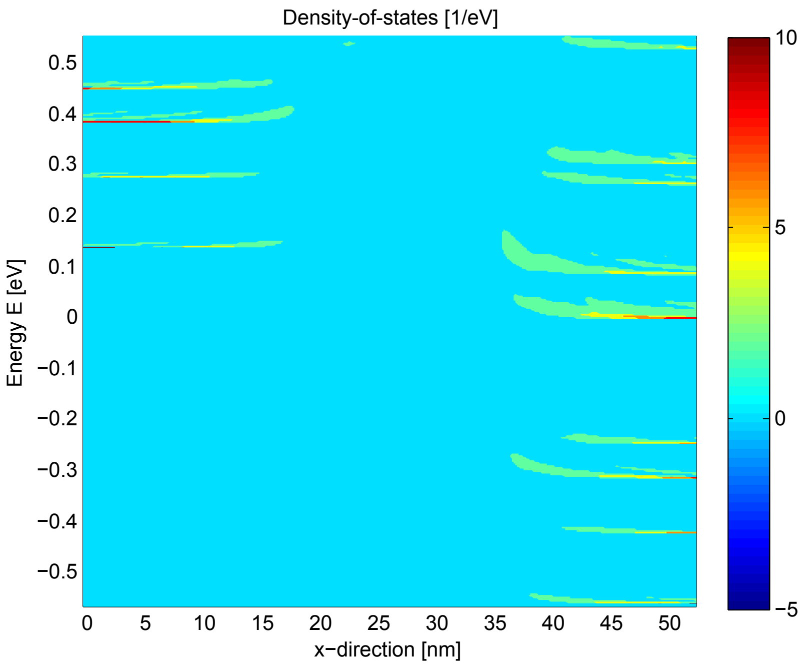


Figure 4.20: **Top: Energy resolved density-of-states in the source and drain region of the Ge 3D FinFET. Bottom: Spectral DOS at Vgs=0.1 V and Vds=0.7V.**

## 4.3 P-type 3D Silicon Nanowire

### 4.3.1 Device geometry

This example presents the simulation of a 3D circular Silicon Nanowire. The transport direction is along the <110> crystal orientation (x-axis), whereas the directions of confinement are aligned with the y- and z-axis. The structure of the transistor is shown in Figure 4.21. It consists of a circular Silicon nanowire surrounded by a gate stack made of $SiO_2$ and a gate contact on top. The structure has a total length of 31 nm while the radius of the semiconducting channel is 2.5 nm. The source and drain contacts are located on the left and right side of the FET. The source and drain region are p-doped, the gate region is intrinsic.

When constructing the device structure in the Si_NW_Circle_110.cmd command file the device geometry is divided into 7 regions. The numbered areas form a wire (1,2,3), an insulator (5), a low-k spacer on the source and drain (4,6), and a gate (7).

All the command files that are needed to simulate the bandstructure, transmission and I-V characteristics of this device can be found in the directory EXAMPLE/SI_NW_CIRCLE_110/.

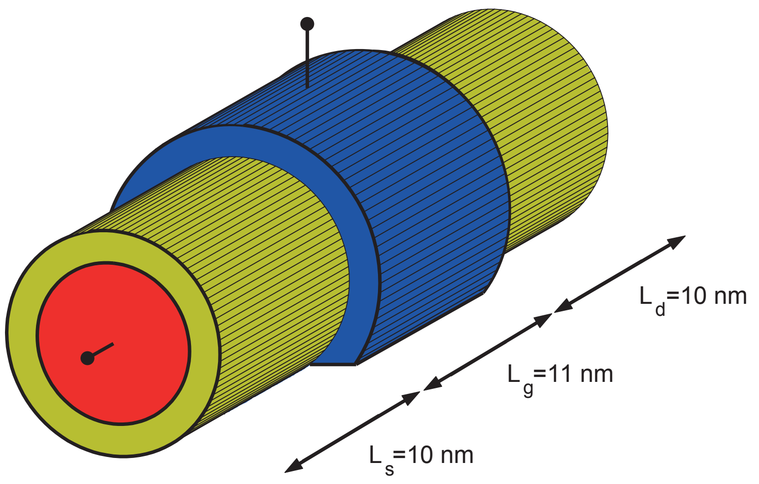


Figure 4.21: **Structure of the considered 3D Silicon Nanowire.** The device consists of a Silicon nanowire surrounded by with $SiO_2$, on the source and drain extensions with a low-k spacer, and a gate contact. The source and drain contacts are located at the left and right side of the device.

### 4.3.2 Bandstructure calculation

To perform a bandstructure calculation the input file must include the line *command(1)=Bandstructure;* in the command section. The simulation takes less than 7 seconds on 1 CPU to compute the simplified parabolic bandstructure of the Silicon NW using the Arpack solver.
For visualization the matlab function *getband.m* can be used. It plots the valence bands of both the drain and source contact where 0 eV usually refers to the bulk valence band edge. Figure 4.22 shows the resulting bandstructure for the considered nanowire.

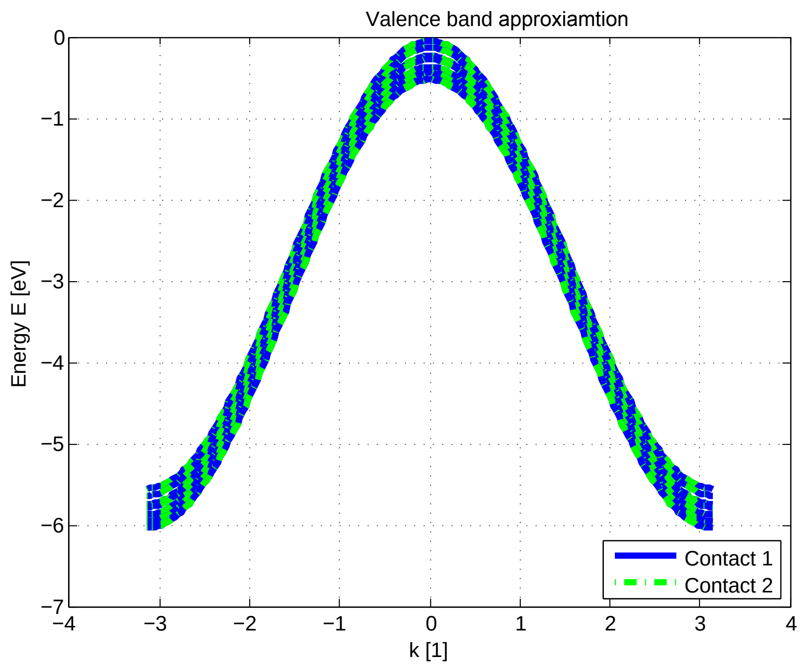


Figure 4.22: **Valence band structure of the 3D Silicon Nanowire shown in Figure 4.21.** The valence band is plotted for both the left and right contact where 0 eV marks the bulk valence band edge.

### 4.3.3 Flat-band condition

To evaluate the flat-band condition and obtain the transmission and density-of-states (DOS) of the given device the input file needs to include *command(i)=Transmission_UWF;* in the command section. UWF represents the Umfpack solver and can be replaced by SWF (SuperLU_DIST Solver), PWF (Pardiso solver) or MWF (MUMPS solver). For visualization the matlab function *checkZETE3D.m* can be used. It plots both the source and drain transmission and density-of-states under the assumption of a flat potential profile. Figures 4.23 and 4.24 show the resulting plots for this Silicon nanowire.

### 4.3.4 Self-consistent transport simulation

The I-V characteristics of the 3D Silicon Nanowire can be computed through a self-consistent Schrödinger-Poisson simulation. A source, drain and gate bias can be selected in the input command file and swept to determine the output/transfer characteristics. Only ballistic transport within the effective mass approximation is available. In this example the gate voltage is swept from -0.3 to 0.1 V in 5 steps while the source (Vs=0 V) and drain (Vds=-0.7 V) remain constant (transfer characteristics). The complete transport simulation takes between 20 min and 2 days per bias point on 1 CPU for mode-space and real-space computation, respectively.

For both simulation approaches, the resulting entities are equal and are presented in Figure 4.25 showing the transfer characteristics and Figures 4.26 and 4.27 showing the electron density distribution and the potential energy, respectively.

To obtain the transmission and density-of-states (DOS) of the device the input file needs to include *plot_transmission=1;* and *plot_dos=1;* in the Parameters section. For visualization the matlab function *gettransdos3d.m* can be used. It plots both the source and drain transmission and DOS. Figures 4.28 and 4.29 show the resulting plot for the 3D Silicon Nanowire.

#### 4.3.4.1 Mode-resolved charge density and current

To obtain the mode-resolved current and charge density of the device the input file needs to include *plot_modes=1;* in the parameters section. Figures 4.30 and 4.31 shows both, the current and charge density per mode for the 3D Silicon Nanowire. **NOTE:** When the mode-resolved quantities are summed up over all modes and valleys, one obtains the overall charge density and current.

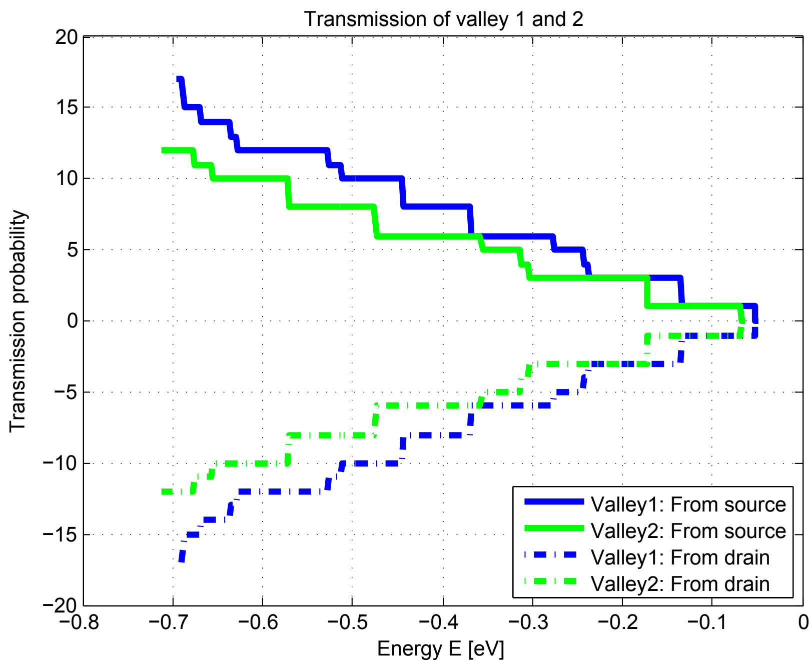


Figure 4.23: **Transmission probability through the 3D Silicon Nanowire without any applied bias potentials (flat-band condition).** The transmission from left to right (solid line) is negative, while the transmission from right to left (dashed line) is positive.

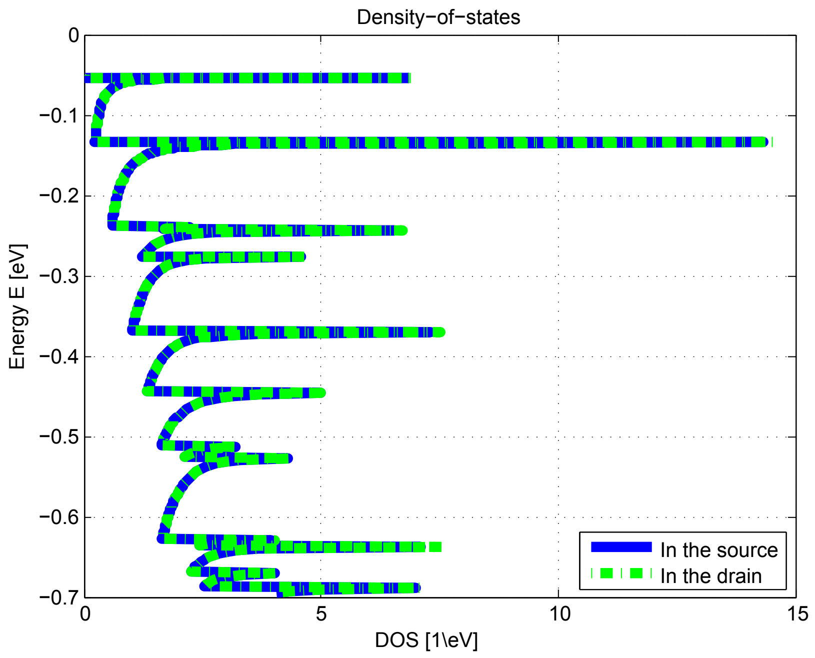


Figure 4.24: **Density-of-states of the 3D Silicon Nanowire in the source and drain regions at flat-band condition.**

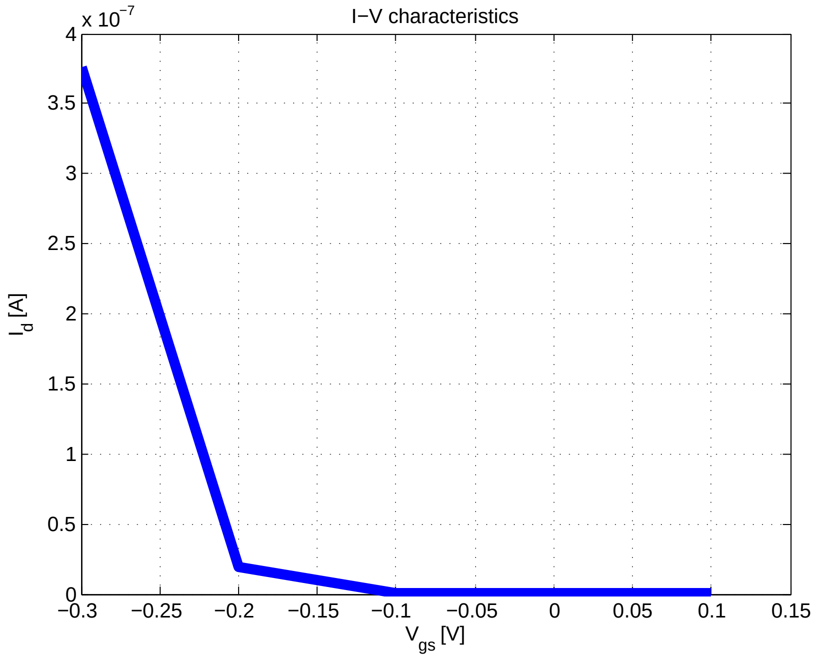
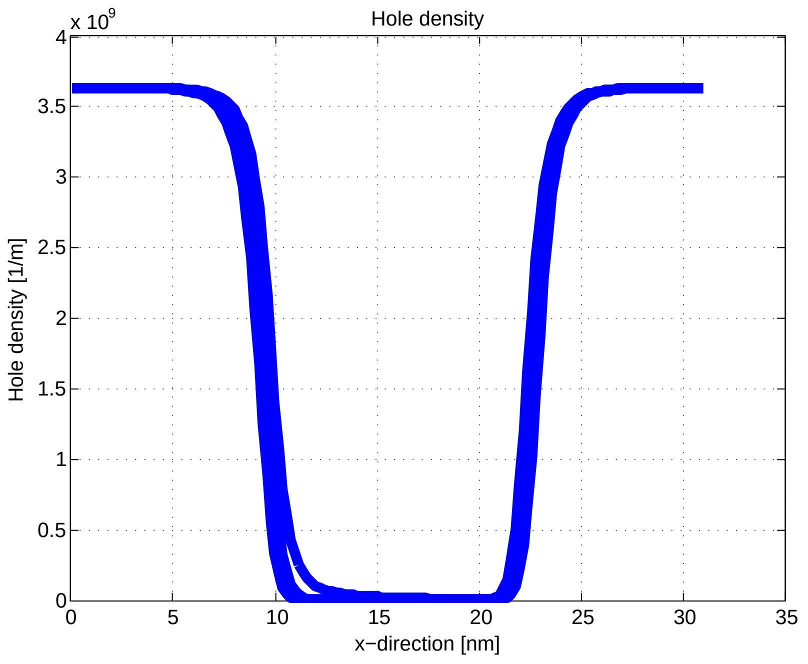


Figure 4.25: **Transfer characteristics Id-Vgs at Vds=-0.7 V of the 3D Silicon Nanowire in mode-space approximation as shown in Figure 4.21.**



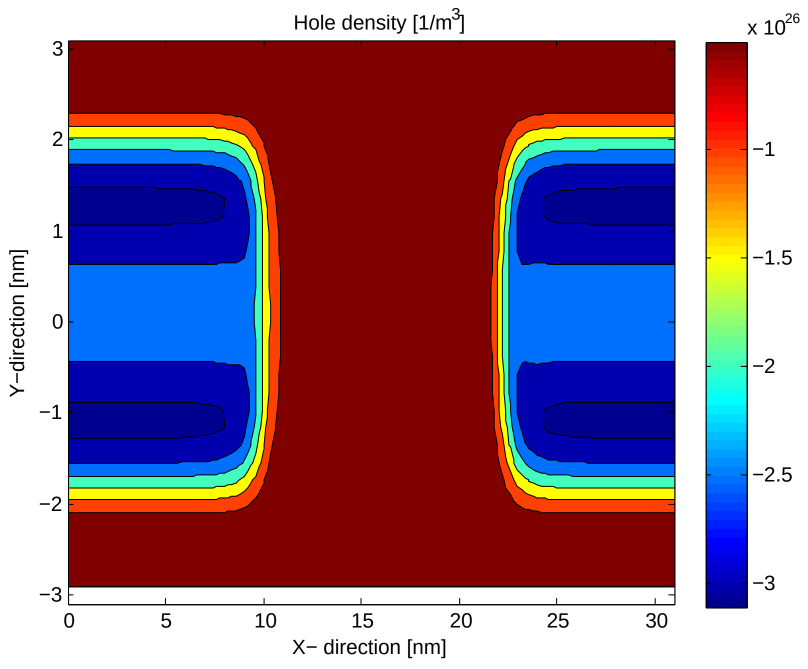


Figure 4.26: **Top: Hole density of the 3D Silicon Nanowire shown along the transport direction at different Vgs.** The mode-space results are shown. The charge density is integrated over the directions of confinement (y and z). **Bottom: 2-D contour of the hole density at Vgs=0.1 V, Vds=0.7 V, and z=2.5 nm.**

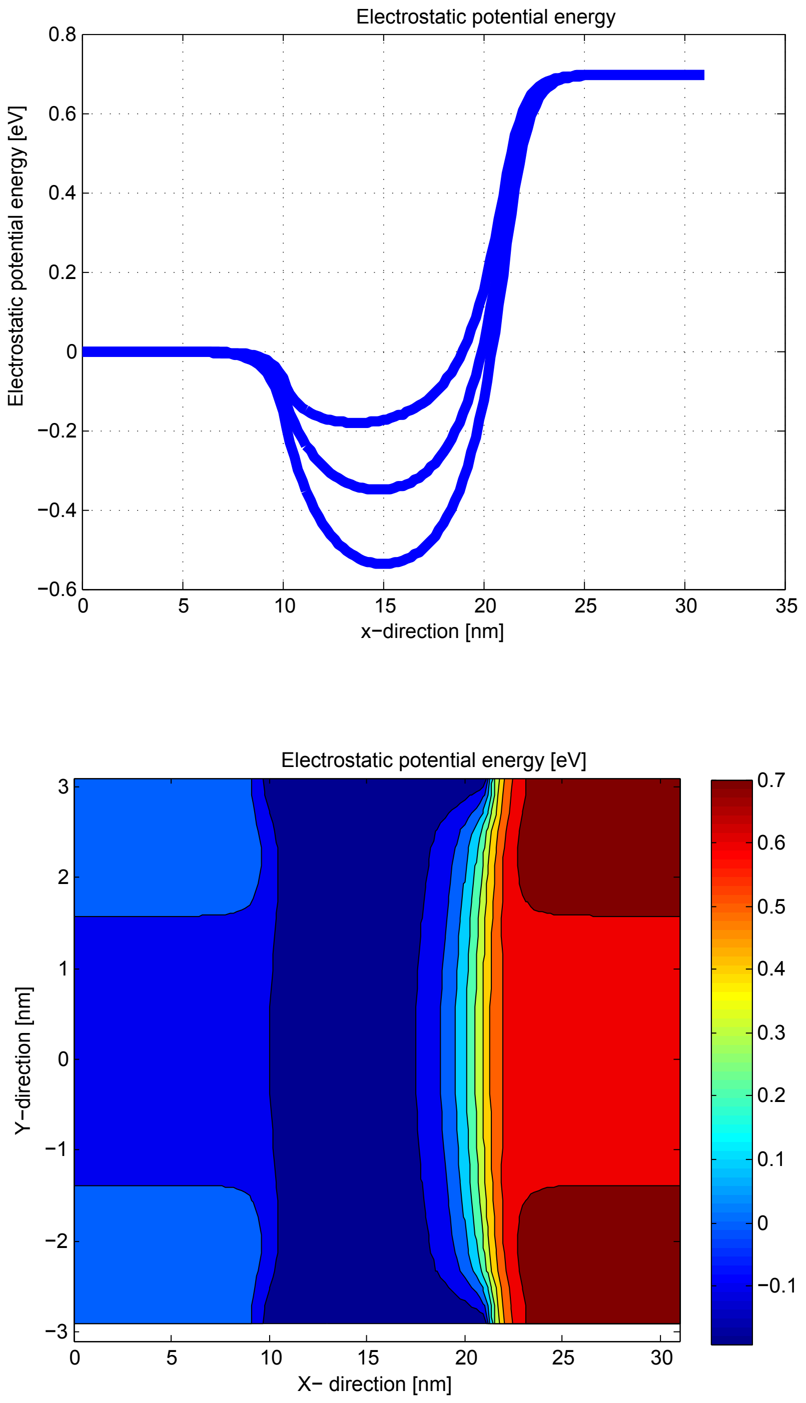


Figure 4.27: **Top: Average electrostatic potential energy along the transport direction at different Vgs in mode-space. Bottom: 2-D contour plot of the electrostatic potential energy at Vgs=0.1 V, Vds=0.7 V, and z=2.5 nm.**

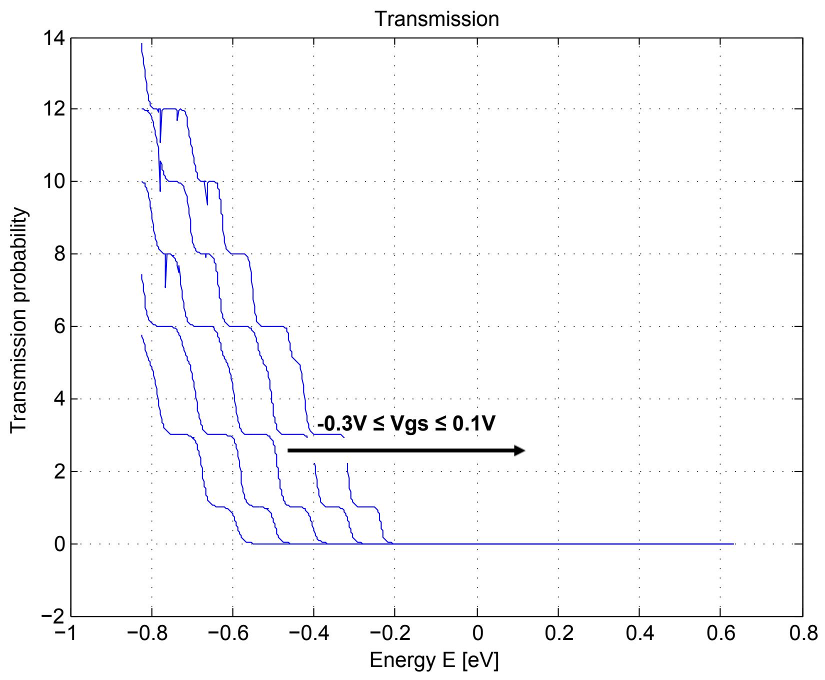


Figure 4.28: **Transmission probability through the transistor shown in Figure 4.21 at different Vgs.**

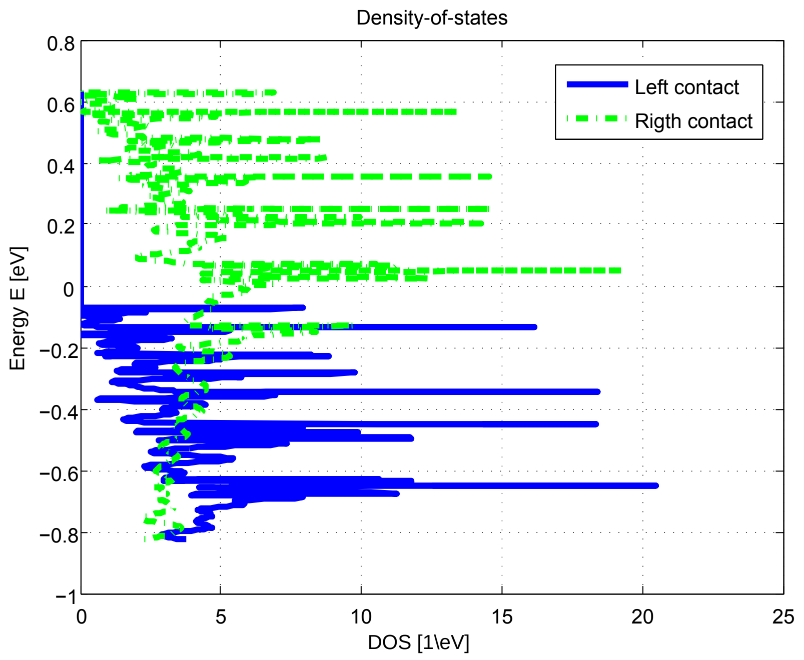


Figure 4.29: **Energy resolved density-of-states in the source and drain region of the 3D Silicon Nanowire.**

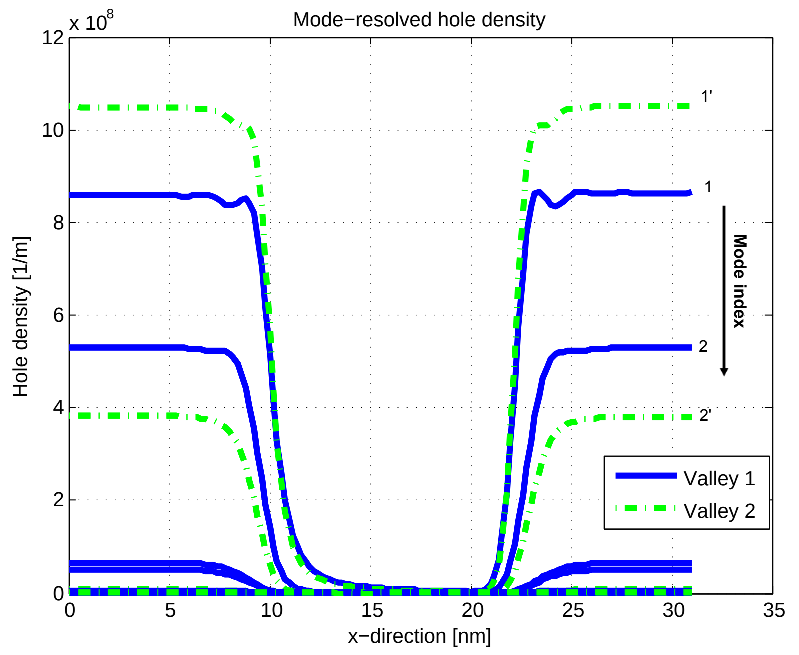


Figure 4.30: **Mode-resolved hole density at Vgs=-0.3 V and Vds=-0.7V.** Both valleys of the circular nanowire are shown. The hole density is decreasing with increasing the mode index.

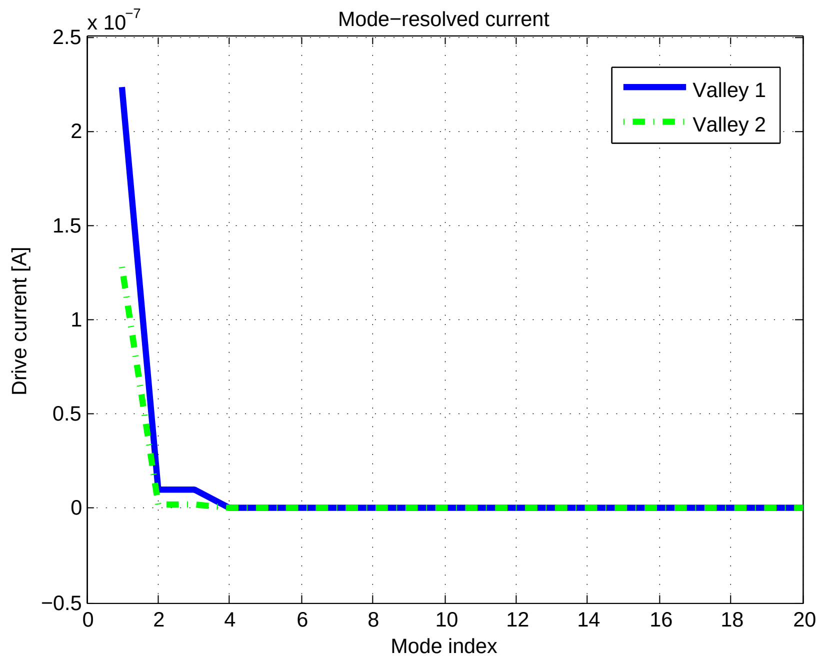


Figure 4.31: **Mode-resolved current at Vgs=-0.3 V and Vds=-0.7V.** Both valleys of the circular nanowire are shown. The current is decreasing with increasing the mode index.

# Bibliography

[1] W. Fichtner, D. J. Rose, and R. E. Bank, "Semiconductor device simulation," *SIAM Journal on Scientific and Statistical Computing*, vol. 4, no. 3, pp. 391–415, 1983.

[2] C. S. Rafferty, M. R. Pinto, and R. W. Dutton, "Iterative methods in semiconductor device simulation," *Electron Devices, IEEE Transactions on*, vol. 32, no. 10, pp. 2018–2027, 1985.

[3] S. Selberherr, A. Schutz, and H. W. Potzl, "Minimos-a two-dimensional mos transistor analyzer," *Solid-State Circuits, IEEE Journal of*, vol. 15, no. 4, pp. 605–615, 1980.

[4] M. Luisier, *Quantum Transport Beyond the Effective Mass Approximation.* Series in microelectronics, Hartung-Gorre, 2007.

[5] M. Luisier, A. Schenk, W. Fichtner, and G. Klimeck, "Atomistic simulation of nanowires in the $sp^3d^5s^*$ tight-binding formalism: From boundary conditions to strain calculations," *Physical Review B*, vol. 74, no. 20, p. 205323, 2006.

[6] M. Luisier and A. Schenk, "Atomistic simulation of nanowire transistors," *Journal of Computational and Theoretical Nanoscience*, vol. 5, no. 6, pp. 1031–1045, 2008.

[7] M. Luisier, "Full-band quantum transport in nanowire transistors," *Journal of Computational Electronics*, vol. 7, no. 3, pp. 309–314, 2008.

[8] M. Luisier and G. Klimeck, "A multi-level parallel simulation approach to electron transport in nano-scale transistors," in *High Performance Computing, Networking, Storage and Analysis, 2008. SC 2008. International Conference for*, pp. 1–10, IEEE, 2008.

[9] M. Luisier, A. Schenk, and W. Fichtner, "Quantum transport in two-and three-dimensional nanoscale transistors: coupled mode effects in the nonequilibrium greenÕs function formalism," *Journal of applied physics*, vol. 100, no. 4, p. 043713, 2006.

[10] W. R. Frensley, "Boundary conditions for open quantum systems driven far from equilibrium," *Reviews of Modern Physics*, vol. 62, no. 3, p. 745, 1990.

[11] A. Svizhenko, M. Anantram, T. Govindan, B. Biegel, and R. Venugopal, "Two-dimensional quantum mechanical modeling of nanotransistors," *Journal of Applied Physics*, vol. 91, no. 4, pp. 2343–2354, 2002.

[12] T. A. Davis, "A column pre-ordering strategy for the unsymmetric-pattern multifrontal method," *ACM Transactions on Mathematical Software (TOMS)*, vol. 30, no. 2, pp. 165–195, 2004.

[13] P. R. Amestoy, I. S. Duff, and J.-Y. L'Excellent, "Multifrontal parallel distributed symmetric and unsymmetric solvers," *Computer methods in applied mechanics and engineering*, vol. 184, no. 2, pp. 501–520, 2000.

[14] X. S. Li and J. W. Demmel, "Superlu_dist: A scalable distributed-memory sparse direct solver for unsymmetric linear systems," *ACM Transactions on Mathematical Software (TOMS)*, vol. 29, no. 2, pp. 110–140, 2003.

[15] O. Schenk and K. Gärtner, "Solving unsymmetric sparse systems of linear equations with pardiso," *Future Generation Computer Systems*, vol. 20, no. 3, pp. 475–487, 2004.

[16] W. Gropp, E. Lusk, N. Doss, and A. Skjellum, "A high-performance, portable implementation of the mpi message passing interface standard," *Parallel computing*, vol. 22, no. 6, pp. 789–828, 1996.

[17] http://www.cise.ufl.edu/research/sparse/amd/.

[18] http://www.netlib.org/.

[19] http://www.cs.sandia.gov/CRF/aztec1.html.

[20] http://people.sc.fsu.edu/~jburkardt/c$_$src/metis/metis.html.

[21] http://graal.ens-lyon.fr/MUMPS/.

[22] http://www.pardiso-project.org/.

[23] http://crd-legacy.lbl.gov/~xiaoye/SuperLU/.

[24] http://www.cise.ufl.edu/research/sparse/UFconfig/.

[25] http://www.cise.ufl.edu/research/sparse/umfpack/.