Sentaurus™ Device QTX User Guide

Version N-2017.09, September 2017

SYNOPSYS®

Synopsys, Inc.
690 E. Middlefield Road
Mountain View, CA 94043
www.synopsys.com

Contents

About This Guide

This user guide describes the Synopsys® Sentaurus™ Device QTX tool that features the subband-based Boltzmann transport equation solver. This user guide is intended for users of TCAD Sentaurus involved in advanced CMOS transport modeling.

Conventions

The following conventions are used in Synopsys documentation.

Convention	Description
Blue text	Identifies a cross-reference (only on the screen).
Bold text	Identifies a selectable icon, button, menu, or tab. It also indicates the name of a field or an option.
`Courier font`	Identifies text that is displayed on the screen or that the user must type. It identifies the names of files, directories, paths, parameters, keywords, and variables.
Italicized text	Used for emphasis, the titles of books and journals, and non-English words. It also identifies components of an equation or a formula, a placeholder, or an identifier.

Customer Support

Customer support is available through the Synopsys SolvNet customer support website and by contacting the Synopsys support center.

Accessing SolvNet

The SolvNet support site includes an electronic knowledge base of technical articles and answers to frequently asked questions about Synopsys tools. The site also gives you access to a wide range of Synopsys online services, which include downloading software, viewing documentation, and entering a call to the Support Center.

To access the SolvNet site:

1. Go to the web page at https://solvnet.synopsys.com.
2. If prompted, enter your user name and password. (If you do not have a Synopsys user name and password, follow the instructions to register.)

If you need help using the site, click **Help** on the menu bar.

Contacting Synopsys Support

If you have problems, questions, or suggestions, you can contact Synopsys support in the following ways:

- Go to the Synopsys Global Support Centers site on synopsys.com. There you can find e-mail addresses and telephone numbers for Synopsys support centers throughout the world.
- Go to either the Synopsys SolvNet site or the Synopsys Global Support Centers site and open a case online (Synopsys user name and password required).

Contacting Your Local TCAD Support Team Directly

Send an e-mail message to:

- support-tcad-us@synopsys.com from within North America and South America.
- support-tcad-eu@synopsys.com from within Europe.
- support-tcad-ap@synopsys.com from within Asia Pacific (China, Taiwan, Singapore, Malaysia, India, Australia).
- support-tcad-kr@synopsys.com from Korea.
- support-tcad-jp@synopsys.com from Japan.

CHAPTER 1 Introduction to Sentaurus Device QTX

This chapter describes the features of the subband-based Boltzmann transport equation solver of Sentaurus Device QTX and the physical models it uses for modeling quasi-ballistic transport.

Features of Sentaurus Device QTX

Sentaurus Device QTX features the subband-based Boltzmann transport equation solver (hereafter, referred to as the *Subband-BTE solver*) that takes into consideration the effects of strong quantum confinement as well as quasi-ballistic transport along the channel of small FinFETs and nanowires. Both purely ballistic transport and scattering due to nonpolar phonon, polar optical phonon, surface roughness, alloy, and Coulomb scattering can be treated.

The subbands used by the Subband-BTE solver are provided by the solution of the Schrödinger equation on 2D slices along the channel. The solution of the Subband-BTE solver provides the distribution functions of carriers in each subband. This information, combined with the subband dispersion and the wavefunctions from the Schrödinger solver, provides the carrier density used in the Poisson equation. As shown in Figure 1, at each bias point, the Schrödinger equation, the subband-based Boltzmann transport equation (subband-BTE), and the Poisson equation are iterated until convergence.

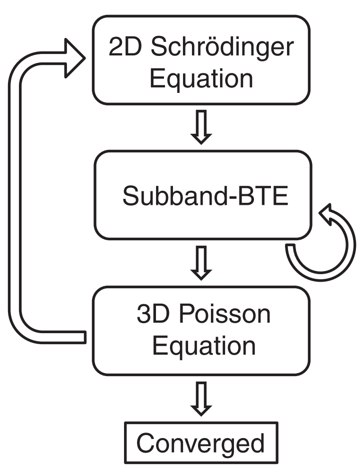

Figure 1 Schematic of the iteration procedure for the self-consistent solution of the Schrödinger equation, the subband-BTE, and the Poisson equation

The main output of the solution of the subband-BTE is the current flowing through the channel. In addition, several internal quantities related to the solution of the subband-BTE, such as the distribution function, the current spectrum, and the carrier velocity, can be saved to TDR files.

Similarities to Sentaurus Band Structure

The Subband-BTE solver shares many commands and physical models with Sentaurus Band Structure.

For descriptions of common commands, models, and parameters, refer to Part II of the *Sentaurus™ Device Monte Carlo User Guide*.

Device Structure

The simulation of carrier transport using the solution of the subband-BTE can be performed only for 3D devices. The use of Sentaurus Device QTX is most appropriate for devices with a well-defined channel, in which geometric quantum confinement plays a strong role. This is typically the case in small FinFETs and nanowire devices. The device structure is loaded from a TDR file using the `LoadDevice` command.

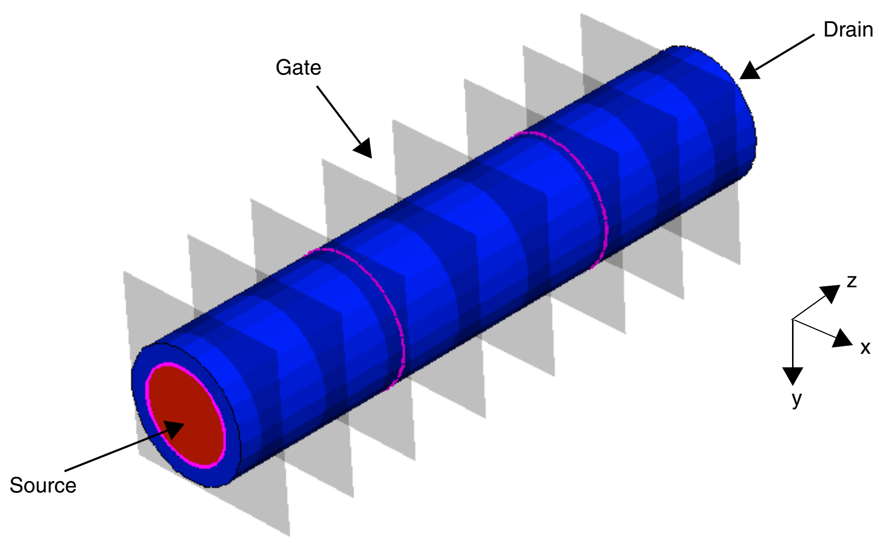

Figure 2 Three-dimensional nanowire device showing slices and contacts

Requirements and Restrictions on Device Structure and Mesh

Sentaurus Device QTX has some requirements and restrictions on the type of device structure and mesh that can be treated, namely:

- The 3D mesh must be composed of all tetrahedra.
- The mesh within the channel should essentially be an extruded mesh from a 2D cross section. That is, along the channel direction, the mesh points should all fall on the slice locations.

Device Axes and Orientation

The device axis along the channel direction must be the z-axis. It must be set up correctly during structure generation. The confinement axes must be the x-axis and the y-axis. The orientation of the device relative to the crystallographic axes can be specified using the `xDirection` and `yDirection` parameters of the `Physics` command.

Contacts

It is expected that the left and right ends of the channel are terminated by contacts, as shown in Figure 2 on page 2. Additional contacts, typically one gate contact, can be included in the structure as well. However, the current is computed only for the left and right contacts of a channel.

During the solution of the Poisson equation, a special Neumann boundary condition should be used at the left and right contacts to enable charge neutrality there to be maintained, even in the presence of quantum confinement. To enable this boundary condition, the `Physics contact` command can be used along with `type=Neumann`, for example:

```
Physics contact=source type=Neumann
Physics contact=drain  type=Neumann
```

A lumped resistance in units of ohm (Ω) can be specified for the left and right contacts of a sliced channel. For example, to specify a lumped resistance of $1\,\mathrm{k}\Omega$ for contacts named `source` and `drain`, specify:

```
Physics contact=source resist=1.0e3
Physics contact=drain  resist=1.0e3
```

Sliced Channels

To solve the subband-BTE along the channel of a device, the subbands from the solution of the 2D Schrödinger equation at slices along the channel must be computed first. A group of these slices forms a so-called sliced channel. On each slice, a 2D Schrödinger equation is solved for the subband dispersions and wavefunctions. Based on these quantities, along with the distribution function for each subband, the carrier density on each node in the slice can be computed. Then, this is interpolated on to the 3D device mesh for solving the 3D Poisson equation.

Creating a Sliced Channel

You create a sliced channel using the `Math slicedChannel` command (see Math slicedChannel on page 25). This is similar to the `Math nonlocal` command used in other parts of Sentaurus Band Structure. The parts of the device structure to be included in the sliced channel can be specified by using either a list of regions, or a bounding box, or both.

For example, to create a sliced channel consisting of only the silicon region, named `si1`, of a silicon nanowire, use the following command:

```
Math slicedChannel name=channel1 regions=[list si1]
```

During the creation of a sliced channel using the `Math` command, two key operations are performed automatically:

- First, the 2D slices along the channel are extracted automatically.
- Second, a nonlocal area is created on each slice with the name given by the sliced channel name appended by `.NL`, for example, `channel1.NL`.

Specifying a Schrödinger Solver for a Sliced Channel

Specifying a Schrödinger solver for a sliced channel is very similar to specifying a Schrödinger solver for a nonlocal area for mobility calculations in Sentaurus Band Structure. For a sliced channel, the `Physics slicedChannel` command is used, and the specified Schrödinger solver is used for all slices in the sliced channel (see Physics slicedChannel on page 27).

Examples:

```
Physics slicedChannel=channel1 eSchrodinger=Parabolic \
  valleys=[list Delta1 Delta2 Delta3] Nsubbands=8 Nk=16 Kmax=0.6 \
  a0=5.43e-8 correction=3
```

```
Physics slicedChannel=channel1 hSchrodinger=6kp \
   valleys=[list Gamma] Nsubbands=32 Nk=16 Kmax=0.6 a0=5.43e-8
```

Model and Parameter Specification

Similar to the specification of models and parameters for the subband and mobility calculations in Sentaurus Band Structure, for the Subband-BTE solver, models and parameters can be specified on a material or region basis.

These specified models and parameters will be used on all slices that contain the specified material or region. In the following sections, two model types are highlighted that are particularly important for the Subband-BTE solver: valley models and scattering models.

Valley Models

Valley models are used to specify the band model and parameters to use when solving the Schrödinger equation.

The following valley models can be used for the subband-BTE: `ConstantEllipsoid`, `2kpEllipsoidal`, `6kp`, and `3kp`. The first two models can be used for the description of electrons, the `6kp` valley model gives an accurate description of holes, and the `3kp` valley model is an approximation of `6kp`, where the split-off band is not captured due to the neglect of spin-orbit coupling.

Scattering Models

Scattering models are used to model specific, microscopic scattering transitions such as phonon and surface roughness scattering. All of the scattering models that can be used for mobility calculations in Sentaurus Band Structure on 2D structures can also be used when solving the subband-BTE. These models include elastic and inelastic nonpolar phonon scattering, polar optical phonon scattering, alloy scattering, surface roughness scattering, and Coulomb scattering from bulk impurities, interface charge, and traps. In addition, the surface roughness and the Coulomb scattering models can be screened using either the scalar or tensor dielectric function models.

A standard set of scattering models is set up for silicon regions by default, as described in the documentation of Sentaurus Band Structure in the *Sentaurus™ Device Monte Carlo User Guide*.

For purely ballistic simulations, all scattering models must be removed. To remove all scattering models, use the following command:

```
Physics material=all ScatteringModel removeAll
```

RTA Scattering Model

The relaxation time approximation (RTA) scattering model is available for solving the subband-BTE. This model is implemented as an intra-subband scattering mechanism using the transition rate given in [1]:

$$S_{\mathrm{RTA}} = \frac{f_n(z,\varepsilon) - f_n^{\mathrm{leq}}(z,\varepsilon)}{\tau} \tag{1}$$

where ε is the total energy and f_n^{leq} is an equilibrium Fermi–Dirac distribution function defined in terms of a local Fermi level, E_F^{leq}:

$$f_n^{\mathrm{leq}}(z,\varepsilon) = \frac{1}{1 + e^{(\varepsilon - E_F^{\mathrm{leq}})/k_{\mathrm{B}}T}} \tag{2}$$

where T is the ambient temperature. The only adjustable parameter in the RTA scattering model is τ, the relaxation time which is taken as a constant. The local Fermi level, E_F^{leq}, is determined by enforcing a carrier conservation condition on the equilibrium Fermi–Dirac distribution function integrated over energy. This produces an additional equation that must be solved along with the subband-BTE. The local Fermi level can be saved to a TDR file for visualization using the `BTESaveCC` command (see BTESaveCC on page 34).

The RTA scattering model serves two purposes:

- First, the RTA scattering model helps to stabilize the numeric solution of the subband-BTE. By default, an RTA scattering model is always used with a built-in τ = 1e-10 s, which is a long enough relaxation time so as not to impact the computed current and is a short enough relaxation time so as to provide efficient numeric stability. This value of τ for the RTA scattering model can be changed when selecting the numeric Subband-BTE solver as described next (see Eq. 3).
- Second, the RTA scattering model provides an efficient model that can be used to mimic more complicated mechanisms that might incur significant runtime or are not yet available. For this purpose, additional RTA scattering models can be specified using the `Physics ScatteringModel` command on a per-region and per-valley basis, for example:

```
Physics region=si1 ScatteringModel=RTA name=rta1 valleys=[list Delta1] \
   tau=1.0e-13
```

The final value of τ that is used in the subband-BTE is computed using Mathiessen's rule considering the built-in RTA scattering model and all user-defined RTA scattering models for a particular region and valley, that is:

$$\frac{1}{\tau} = \frac{1}{\tau_{\text{built-in}}} + \sum_i \frac{1}{\tau_i} \tag{3}$$

Surface Roughness

Surface roughness scattering is activated by specifying the `SRFor2D` scattering model. This model is used because it is applied to 2D slices. The `SRFor2D` model applies surface roughness scattering only at semiconductor–insulator interfaces.

During the creation of the sliced channel, Sentaurus Device QTX automatically extracts any insulator regions that are adjacent to the specified semiconductor regions. This does not affect the solution of the Schrödinger equation on the sliced channel, but enables the `SRfor2D` model to work correctly even if no insulator regions are specified explicitly.

Coulomb Scattering

When used with Sentaurus Device QTX, Coulomb scattering can be applied to both bulk doping and interface charge.

To speed up the calculation of the matrix elements for Coulomb scattering, the `whenToCompute` parameter can be used to specify when the matrix elements should be recomputed. The options for this parameter are:

- The `EveryIteration` option causes the matrix elements to be evaluated for each Poisson Newton iteration. This option provides the best accuracy but requires the most runtime. This is the default value.
- The `FirstIteration` option causes the matrix elements to be evaluated only for the first Poisson Newton iteration during a `Solve` command. Essentially, the solution from the previous bias point is used to compute the matrix elements. For subsequent Newton iterations, the matrix elements are reused when computing the scattering rate. This approach can potentially affect accuracy, but it greatly reduces the runtime. For bias ramps, this approach has worked well. If accuracy is important, a bias point can be repeated.

The following example shows how to specify the `whenToCompute` parameter:

```
Physics material=Silicon ScatteringModel=Coulomb name=eCoulomb \
   transitionType=Intravalley valleys=[list Delta1 Delta2 Delta3] \
   screening=Lindhard whenToCompute=FirstIteration
```

Screening

Some of the scattering models, such as surface roughness and Coulomb, can be screened using one of the available screening models. By default, no screening is used. The required screening model can be selected using the `screening` parameter for each scattering model.

This parameter can be set to one of the following values:

- `none` for no screening
- `Lindhard` for the scalar Lindhard screening model
- `LindhardTDF` for the tensor Lindhard screening model

The screening models utilize the polarization (Π), which characterizes the strength of the screening, and the dielectric form-factor (F), which characterizes the screening effectiveness of each subband pair in terms of the wavefunction overlap with the Coulomb Green's function.

The polarization is given by:

$$\Pi_{nn'}(q) = \frac{g_s \cdot g_v}{2\pi}\int dk \frac{f_{n'}(k+q) - f_n(k)}{E_n(k) - E_{n'}(k+q)} \tag{4}$$

where:

- g_s and g_v are the spin and valley degeneracy, respectively.
- E_n and f_n are the dispersion and distribution function for subband n, respectively.

The dielectric form-factor is given by:

$$F^{nn'}_{mm'}(q) = \iint d\boldsymbol{x}d\boldsymbol{x}' \; \psi_m(\boldsymbol{x}) \cdot \psi_{m'}{}^{\dagger}(\boldsymbol{x})G(q, \boldsymbol{x}, \boldsymbol{x}')\psi_n{}^{\dagger}(\boldsymbol{x}') \cdot \psi_{n'}(\boldsymbol{x}') \tag{5}$$

where ψ_m is the wavefunction for subband m, and G is the Coulomb Green's function.

Tensor Dielectric Function

The tensor dielectric function (TDF) in the TDF screening model is given by:

$$\varepsilon^{nn'}_{mm'}(q) = \delta_{mn}\delta_{m'n'} + \Pi_{nn'}(q)F^{nn'}_{mm'}(q) \tag{6}$$

The tensor Lindhard screening model (`LindhardTDF`) is time consuming to compute, particularly when a large number of subband pairs is used for the screening calculation. The number of subband pairs can be controlled using the `tdfDegenTol` parameter of the `Physics slicedChannel` command when specifying the Subband-BTE solver. Subband pairs are used in the screening calculation if their subband edges are within `tdfDegenTol` of each other. It is recommended to set this to a value of 1.0e-4 eV.

Scalar Dielectric Function

The scalar dielectric function (SDF) in the SDF screening model is given by:

$$\varepsilon(q) = 1 + \sum_{n} \Pi_{nn}(q) F_{nn}^{nn}(q) \quad (7)$$

The sum is over the subbands. While it is expensive to compute, the SDF screening model is much faster than the TDF screening model and provides a good trade-off between accuracy and runtime.

Runtime Speedup With Screening

The screening models are computationally expensive. They are heavily parallelized and you should use as many threads as possible. However, even with many threads, for a structure with many slices, the overall wallclock time when using screening may still be quite long.

To provide a trade-off between accuracy and runtime, when specifying the Subband-BTE solver, the `whenToComputeScreening` parameter of the `Physics slicedChannel` command can be used to control how often the screening model is evaluated. The options for this parameter are:

- The `EveryIteration` option causes the screening model to be evaluated for each Poisson Newton iteration. This option provides the best accuracy but requires the most runtime. This is the default value.
- The `FirstIteration` option causes the screening model to be evaluated only for the first Poisson Newton iteration during a `Solve` command. Essentially, the solution from the previous bias point is used to compute the screening dielectric function. For subsequent Newton iterations, the dielectric function is reused. This approach can potentially affect accuracy, but it greatly reduces the runtime. For bias ramps, this approach has worked well. If accuracy is important, a bias point can be repeated.

The following example shows how to specify the `whenToComputeScreening` parameter and the `tdfDegenTol` parameter:

```
Physics slicedChannel=channel1 eBTESolver=Numeric tdfDegenTol=1.0e-4 \
  whenToComputeScreening=FirstIteration
```

Limiting the Subband Scattering Range

For transition types such as `Intravalley`, `Intervalley`, and `gIntervalley`, a state in the initial subband is allowed, by default, to scatter to all valid final states in all valid final subbands. When using a large number of subbands and a fine energy grid for the solution of the subband-BTE, this produces a very large system matrix that can be very time consuming to solve. Restricting the number of final subbands to which an initial state scatters can help to reduce the overall system size and to improve runtime performance. This can be achieved by using the `subbandScatteringRange` parameter of the `Physics slicedChannel` command (see Physics slicedChannel on page 27).

Subband-Based Boltzmann Transport Equation

With quantum confinement in two dimensions, the subband-BTE is reduced to one dimension along the channel within each subband. The solution of the subband-BTE gives the distribution function as a function of the k-vector and the channel coordinate for each subband. Using z for the channel axis, for subband n, the subband-BTE is [2]:

$$-\frac{\partial f_n}{\partial k}\frac{1}{\hbar}\frac{\partial E_n}{\partial z}+\frac{\partial f_n}{\partial z}\frac{1}{\hbar}\frac{\partial E_n}{\partial k} = S_{n,\,\mathrm{in}}-S_{n,\,\mathrm{out}} \tag{8}$$

where:

- f_n is the distribution function for subband n.
- E_n is the dispersion for subband n.

With Fermi–Dirac statistics, the in-scattering term ($S_{n,\,\mathrm{in}}$) and the out-scattering term ($S_{n,\,\mathrm{out}}$) are given by:

$$\begin{aligned} S_{n,\,\mathrm{in}} &= \sum_{n'}\frac{1}{2\pi}\int dk' S_{n'n}(k',k)f_{n'}(z,k')[1-f_n(z,k)] \\ S_{n,\,\mathrm{out}} &= \sum_{n'}\frac{1}{2\pi}\int dk' S_{nn'}(k,k')f_n(z,k)[1-f_n'(z,k')] \end{aligned} \tag{9}$$

where $S_{n'n}(k',k)$ is the total transition rate due to scattering.

As boundary conditions on the distribution functions, equilibrium Fermi–Dirac distribution functions at the left and right contacts are applied to carriers being injected into the channel.

Selecting a Subband-BTE Solver

Two Subband-BTE solvers are available for solving the subband-BTE.

The faster and simpler solver is called `AnalyticBallistic`. As its name suggests, it can be used only for purely ballistic simulations. This solver uses the analytic solution to the subband-BTE, which is determined by projecting the injecting, equilibrium Fermi–Dirac distribution functions from the contacts throughout the device as determined by the top of each subband barrier.

The second solver is called `Numeric`. It performs an actual numeric solution to the subband-BTE. This solver can be used for purely ballistic transport as well as with all of the available scattering models including the RTA scattering model.

For example, to activate the `AnalyticBallistic` subband-BTE for electron transport for a sliced channel named `channel1`, you can use the following command:

```
Physics slicedChannel=channel1 eBTESolver=AnalyticBallistic
```

For a subband-BTE simulation of holes, you can use the `hBTESolver` argument:

```
Physics slicedChannel=channel1 hBTESolver=AnalyticBallistic
```

Energy Grid

The subband-BTE is solved on a 2D tensor-product grid. One axis of this grid is along the channel direction and is referred to as the *channel coordinate axis*. The grid points along this axis are given by the slice locations along the channel. The other axis of the BTE solution grid is the *total energy axis*. Here, *total energy* refers to the total energy of a carrier in a particular subband as determined by the sum of the minimum subband energy and the kinetic energy of the carrier.

The minimum and maximum energies used in the energy grid are determined automatically. The nominal energy grid spacing is set using the `deltaE` argument of the `Physics slicedChannel` command when selecting a Subband-BTE solver (see Physics slicedChannel on page 27).

In addition, a special refinement around the top of each subband can be activated to improve the accuracy of the calculation and to improve convergence. This is activated using the `resolveTOBEnergy` argument. The smallest energy spacing used for this refinement is set using the `minDeltaE` argument.

For example, a typical energy grid setup for a sliced channel named `channel1` is:

```
Physics slicedChannel=channel1 eBTESolver=Numeric deltaE=4e-3 \
  resolveTOBEnergy=1 minDeltaE=1.0e-4
```

Numeric Options

When specifying the Subband-BTE solver (see Physics slicedChannel on page 27), several options related to numeric details can be set, in particular:

- `useParabolicFitForToB=`*`Boolean`*

  When true, this argument activates a special algorithm to more accurately locate the top of the source–drain barrier. This mainly applies to FET devices. Default: true.

- `approximateJacobian=`*`Boolean`*

  When true, this argument activates a special algorithm to approximate the Jacobian that is used when solving the BTE system. This can be used to greatly reduce memory consumption with a small impact on convergence. Default: true.

- `useKdependentWFForBTEDensity=`*`Boolean`*

  When true, this argument computes the carrier density based on k-dependent wavefunctions. When this argument is false, Γ-point wavefunctions are used. By default, this argument is deactivated to improve convergence.

  NOTE If `useKdependentWFForBTEDensity=0` and `useKdependentWF=1` are set when specifying the Schrödinger solver, k-dependent wavefunctions will still be used when computing the scattering rates.

  NOTE If `reorderDispersion` is switched on, you should also switch on `useKdependentWFForBTEDensity` and `useKdependentWF` to have good convergence behavior.

- `NkForScreening=`*`Integer`*

  This argument sets the number of k-points to use when evaluating screening. Default: 32.

Solving at One Bias

The solution of the subband-BTE at a specific bias is initiated using the `Solve` command. Similar to calculations using Sentaurus Band Structure, the bias on each contact is set using the `V(<contact>)` syntax, where `<contact>` refers to the contact name. For example, to initiate a solve at specific biases on the source, drain, and gate contacts, use the following command:

```
Solve V(gate)=1.0 V(drain)=1.0 V(source)=0.0
```

If the bias on a particular contact is not specified explicitly in the `Solve` command, the previously specified bias is used. By default, the bias on all contacts is set to 0 at the start of the simulation.

Convergence Criteria

During the solution of the subband-BTE, three equations are solved in an iterative fashion: the Schrödinger equation, the subband-BTE, and the Poisson equation. The overall iteration of these equations is divided into an outer iteration of the Poisson equation and an inner iteration of the subband-BTE after a single solve of the Schrödinger equation. This is shown in Figure 1 on page 1.

The convergence criteria for the outer Poisson equation are set by the `currentConvTol` and `potentialUpdateTolerance` arguments of the `Math` command. The argument `potentialUpdateTolerance`, set in units of $k_B T/q$, determines the maximum-allowed change in the potential below which the Poisson equation is considered to have converged. For the subband-BTE system of equations, it is recommended to set this argument to a value of approximately 3.0e-3.

The `currentConvTol` argument determines the relative change in terminal currents between subsequent Poisson iterations below which the subband-BTE system is considered to have converged.

The subband-BTE system is considered to have converged when either of these two convergence criteria are met. For example, if `potentialUpdateTolerance=3.0e-3` and `currentConvTol=1.0e-3`, a bias point for the subband-BTE will be considered converged if the maximum change in the potential at a particular Poisson Newton iteration is below $3.0\text{e-}3 \times (k_B T/q)$ or the change in terminal current relative to the previous Newton iteration is below 1.0e-3. By default, `currentConvTol` is set to 0.0, meaning that convergence is completely controlled by the `potentialUpdateTolerance` argument.

The convergence criteria of the inner solution of the subband-BTE are set using the following arguments:

- `distFuncUpdateTolerance` for the distribution functions.
- `rtaUpdateTolerance` for the local Fermi level used in the RTA scattering model.

It is recommended to use the default values for these arguments.

The solution of the subband-BTE requires an initial guess for the distribution functions. For the first several Poisson iterations, this guess is supplied by solving the subband-BTE using only the RTA scattering model. After these initial Poisson iterations, the initial guess is supplied by the solution of the subband-BTE from the previous Poisson iteration. The exact number of Poisson iterations for which the RTA solution is used as an initial guess is set by the `iterationsForRTAGuess` parameter of the `Math` command. It is recommended to use its default value of 5.

For details about these arguments, see Math on page 23.

Output

The primary output of the solution of Sentaurus Device QTX is the current, in ampere, flowing through the left and right contacts at the end of the channel. These current values are printed to the screen at the end of a `Solve` command, and the values also are stored in the bias log file. Furthermore, in the bias log file, when a nonzero lumped resistance is specified, in addition to the applied bias that is stored under `V(<contact>)`, the so-called inner voltage on the inner end of the lumped resistor is stored under `Vinner(<contact>)`.

In addition to the current, several different TDR files with data from the solution of the Subband-BTE solver can be saved. These are described in the next sections. See Chapter 3 on page 23 for full details. Chapter 2 on page 17 presents an example.

2D TDR File With Fields Over Channel Coordinate–Energy Space

You can use the `BTESaveE` command to save a 2D TDR file with fields that are defined on the channel coordinate–energy grid. These fields include quantities such as the distribution function, the current spectrum, and the density-of-states (DOS) in each subband, or in each valley, or in total (see BTESaveE on page 30).

2D TDR File With Fields Over Channel Coordinate–k-Space

You can use the `BTESaveK` command to save a 2D TDR file with fields that are defined over channel coordinate–k-space (see BTESaveK on page 32).

TDR (XY) File With Fields Versus Channel Coordinate

You can use the `BTESaveCC` command to save a TDR (xy) file with fields versus the channel coordinate. The fields that can be saved include the total inversion charge, the total current, the total carrier velocity, as well as these quantities per subband. In addition, the occupancy of each subband can be saved (see BTESaveCC on page 34).

Slice-Related Fields

Each slice that is used in the solution of the subband-BTE is basically a 2D cross section. Similar to the direct treatment of a 2D device structure using Sentaurus Band Structure, various fields across this 2D cross section, as well as fields in 1D k-space, can be saved using two commands.

2D TDR File With Fields Over XY Real Space

You can use the `SaveSlice` command to save quantities from the Schrödinger solver on a slice such as the subband energy and the wavefunctions, as well as quantities such as the conduction band energy (see SaveSlice on page 36).

TDR (XY) File With Fields Over 1D k-Space

You can use the `SaveSliceK` command to save the 1D k-space dispersion for each subband computed by the Schrödinger solver (see SaveSliceK on page 37).

References

[1] S. Jin, T.-W. Tang, and M. V. Fischetti, "Simulation of Silicon Nanowire Transistors Using Boltzmann Transport Equation Under Relaxation Time Approximation," *IEEE Transactions on Electron Devices*, vol. 55, no. 3, pp. 727–736, 2008.

[2] D. Esseni, P. Palestri, and L. Selmi, *Nanoscale MOS Transistors: Semi-Classical Transport and Applications*, Cambridge: Cambridge University Press, 2011.

CHAPTER 2 Getting Started Example

This chapter presents an example that uses Sentaurus Device QTX to compute an I_d–V_g curve for a silicon NMOS nanowire.

This chapter describes the command file for simulating the ballistic transport in a silicon nanowire device using Sentaurus Device QTX. The various sections of the command file used to load the device structure, to set up the models, to specify the sliced channel, and to perform the I_d–V_g simulation are described.

Device Structure

Figure 3 shows the device structure for this example. The device is a cylindrical silicon NMOS nanowire with a diameter of 5 nm, a gate length of 13 nm, and source and drain extension regions of length 10 nm. The source and drain are uniformly doped n-type to 2e20 cm^{-3}. This example computes an I_d–V_g curve at $V_d = 0.6$ V in the ballistic limit using the `Numeric` Subband-BTE solver.

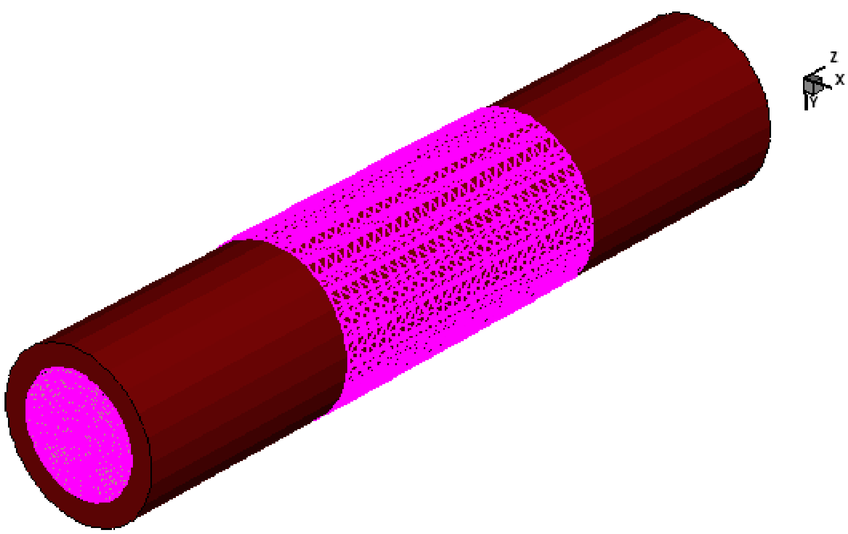

Figure 3 Silicon nanowire device for the example

Loading the Device Structure

```
LoadDevice tdrFile=nanowire3D_0.tdr
```

The device structure, `nanowire3D_0.tdr`, is loaded using the `LoadDevice` command. Upon loading, the tool automatically sets up the default models and parameters in the silicon and oxide regions.

Setting Up the Model and the Classical Solve

```
# Simple model for hole density
Physics material=Silicon hBulkDensity=hFermiDensity Nv=3.10e19

# Remove all scattering models for ballistic simulation
Physics material=all ScatteringModel removeAll

# Use Neumann boundary condition on source and drain
Physics contact=source type=Neumann
Physics contact=drain  type=Neumann

# Set workfunction
Physics contact=gate workfunction=4.25

# Set orientation for <110> channel
Physics xDirection=[list 1 1 0] yDirection=[list 0 0 1]

# Classical solve
Solve
```

These commands set up particular models and parameters, starting with specifying a simple model for the hole density. Then, all of the scattering models are removed so that a ballistic simulation can be performed. The boundary condition type for the source and drain contacts is set to Neumann. Finally, the gate workfunction is set to 4.25 V.

After the required models are specified, a classical solve of only the Poisson equation is performed in equilibrium, that is, zero bias on all contacts. A *classical solve* means that the Schrödinger equation is not used. The resulting classical solution will serve as the initial guess for the first solve using the Schrödinger equation.

Specifying the Sliced Channel

```
# Create sliced channel over all device for si1 region
Math slicedChannel name=channel1 regions=[list si1]

# Set up parabolic Schrodinger on the sliced channel
Physics slicedChannel=channel1 eSchrodinger=Parabolic \
   valleys=[list Delta1 Delta2 Delta3] Nsubbands=8 Nk=16 Kmax=0.6 \
   a0=5.43e-8 correction=3

# Select Numeric Subband-BTE solver and set energy grid parameters
Physics slicedChannel=channel1 eBTESolver=Numeric deltaE=4.0e-3 \
   resolveTOBEnergy=1 minDeltaE=1.0e-4
```

This section of the command file specifies the sliced channel, the Schrödinger solver, and the Subband-BTE solver.

The `Math slicedChannel` command creates a sliced channel named `channel1` over the region named `si1` throughout the entire device.

The first `Physics slicedChannel` command specifies that a parabolic Schrödinger solver must be used on all slices of the sliced channel. This command refers to the `Delta1`, `Delta2`, and `Delta3` valley models that were created by default during the `LoadDevice` command.

The second `Physics slicedChannel` command changes the Subband-BTE solver to `Numeric` and specifies some energy grid parameters.

Performing the I_d–V_g Simulation

```
# Set some convergence parameters. It continues to next bias if not converged
Math potentialUpdateTolerance=6.0e-3 iterations=30 doOnFailure=0

# Solve at equilibrium and then at required drain bias
Solve V(gate)=0.0 V(drain)=0.0

Solve V(drain)=0.6

# Ramp gate using a Tcl foreach loop
foreach Vg [list 0.0 0.1 0.2 0.3 0.4 0.5 0.6] {
   Solve V(gate)=$Vg
   AddToLogFile name=Id value=[GetLast name=I(drain)]
}
```

This section of the command file computes the I_d–V_g curve. First, the `Math` command specifies the convergence tolerance for the Poisson equation. The number of allowed iterations for each bias is set to 30, and the `doOnFailure` argument is set such that the tool will continue to the next bias, even if the convergence criteria are not met.

This series of `Solve` commands will write data to the default bias log file named `subbandBTE_Quickstart.plt`. An extra user-defined quantity named `Id` is added to the bias log file to make plotting the I_d–V_g curve easier in Sentaurus Visual. Figure 4 shows the resulting I_d–V_g curve.

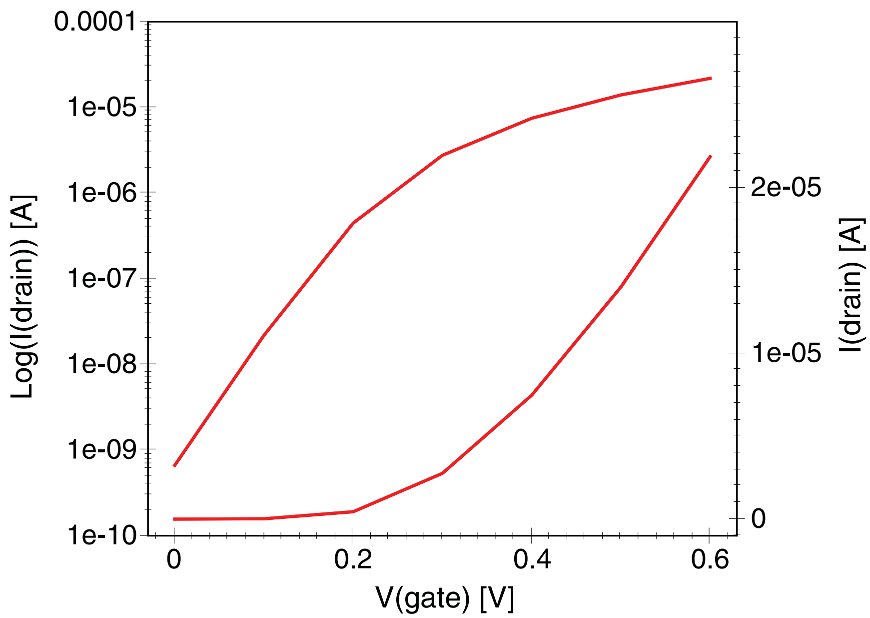

Figure 4 Ballistic I_d–V_g curve from the example

Saving TDR Files

```
# Save 3D TDR
Save tdrFile=subbandBTE_Quickstart.tdr \
   models=[list DopingConcentration eDensity hDensity ConductionBandEnergy \
           eQuasiFermiEnergy]

# Save fields over ChannelCoord-Energy
BTESaveE tdrFile=subbandBTE_Quickstart_CC_E.tdr \
   models=[list Delta1_0_DistributionFunction Delta1_0_CurrentSpectrum \
           Delta1_0_SubbandEnergy Delta1_0_DOS]

# Save distribution function over ChannelCoord-k
BTESaveK tdrFile=subbandBTE_Quickstart_CC_K.tdr \
   models=[list Delta1_0_DistributionFunction]
```

```
# Save subband quantities over ChannelCoord
BTESaveCC tdrFile=subbandBTE_Quickstart_CC.tdr \
   models=[list NinvTotal CurrentTotal VelocityTotal \
           Delta1_0_Occupancy Delta3_0_Occupancy]

# Save fields over a slice
SaveSlice tdrFile=subbandBTE_Quickstart_Slice.tdr channelCoord=20.0e-3 \
   models=[list eDensity Delta1_0_Wavefunction]

# Save dispersion at a slice
SaveSliceK tdrFile=subbandBTE_Quickstart_SliceK.tdr channelCoord=20.0e-3 \
   models=[list Delta1_0_Dispersion]
```

This section of the command file writes various TDR files with different types of data from the solution of the subband-BTE:

- The `Save` command writes a 3D TDR file with the 3D device structure and the selected models.
- The `BTESaveE` command writes a 2D TDR file over channel coordinate–energy space. In this example. the distribution function, the current spectrum, the subband energy, and DOS for the `Delta1_0` subband are included. Figure 5 on page 22 shows the plot of the resulting distribution function.
- The `BTESaveK` command saves a TDR file over channel coordinate–k-space with only the distribution function for the `Delta1_0` subband.
- The `BTESaveCC` command saves a TDR (xy) file with the selected models versus the channel coordinate. Figure 6 on page 22 shows the plot of the resulting total inversion change (NinvTotal) and the carrier velocity (VelocityTotal) along the channel.
- The final two commands, `SaveSlice` and `SaveSliceK`, save real-space and k-space quantities, respectively, from the slice located closest to the channel coordinate of 20 nm.

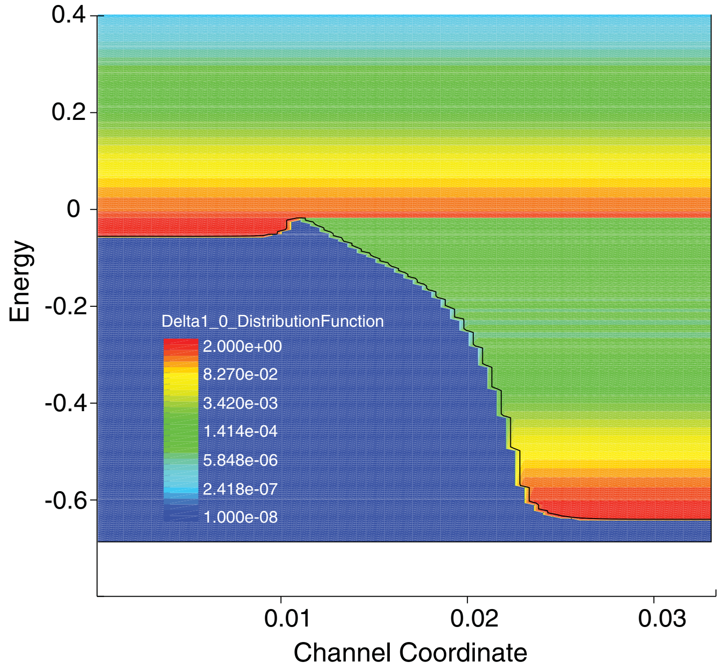

Figure 5 Plot of the distribution function in the Delta1_0 subband from the BTESaveE command

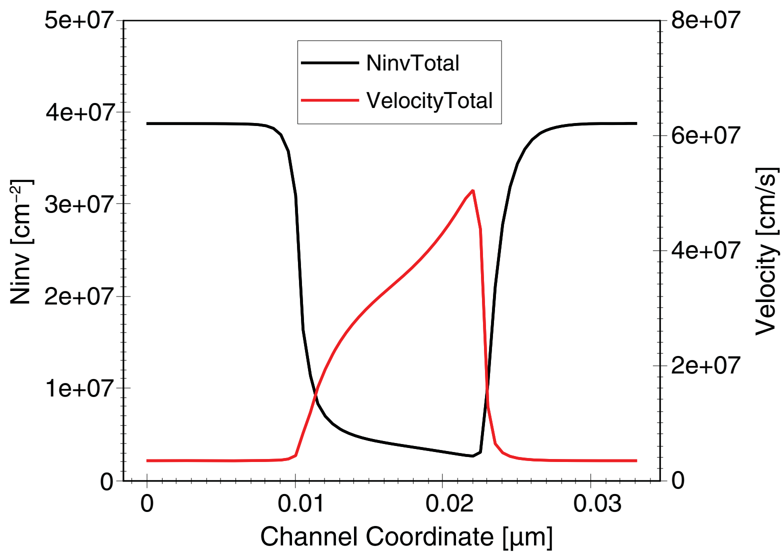

Figure 6 Plot of the total inversion charge and carrier velocity along the nanowire channel from the BTESaveCC command

CHAPTER 3 Commands of Sentaurus Device QTX

This chapter summarizes the commands used by Sentaurus Device QTX.

Math and Physics Commands

This section describes the commands used to specify the geometry and the physics of a sliced channel as well as the convergence criteria for the solution of the subband-BTE.

Math

Sets the convergence criteria for solving the subband-BTE.

Syntax

```
Math [currentConvTol=Double] [distFuncUpdateTolerance=Double] \
   [iterationsForRTAGuess=Integer] [potentialUpdateTolerance=Double] \
   [rtaUpdateTolerance=Double]
```

Arguments

Argument	Description	Default	Unit
currentConvTol	Tolerance on the relative error in the terminal current compared to the previous Newton iteration.	0.0	1
distFuncUpdateTolerance	Tolerance on the infinity norm for the update to the distribution functions.	1.0e-8	1
iterationsForRTAGuess	Number of iterations at the start of a solve for which the solution of the subband-BTE with only RTA scattering is used as an initial guess.	5	1
potentialUpdateTolerance	Convergence tolerance applied to the potential update in units of the thermal voltage.	1.0e-5	$\frac{k_B T}{q}$
rtaUpdateTolerance	Tolerance on the infinity norm for the update to the local Fermi levels for the RTA scattering model.	1.0e-2	eV

Description

The basic `Math` command sets the convergence criteria for solving the subband-BTE using the `Numeric` Subband-BTE solver along with the RTA scattering model. The subband-BTE is considered converged when both criteria specified by `distFuncUpdateTolerance` and `rtaUpdateTolerance` are met.

The `iterationsForRTAGuess` parameter can be used to control the initial guess for the solution of the subband-BTE. Setting this to a large value can help to improve convergence at the beginning of a `Solve`; however, it can result in a large overall number of Poisson iterations.

Examples

```
Math distFuncUpdateTolerance=1.0e-7 rtaUpdateTolerance=1.0e-3
```

This command sets the update tolerance on the distribution functions to 1.0e-7 and on the RTA Fermi levels to 1.0e-3 eV.

Math slicedChannel

Creates a sliced channel within a 3D device.

Syntax

```
Math slicedChannel name=String [regions=List] [channelAxis=String] \
   [minX=Double] [maxX=Double] [minY=Double] [maxY=Double] [minZ=Double] \
   [maxZ=Double]
```

Arguments

Argument	Description	Default	Unit
slicedChannel	Indicates the creation of a new sliced channel.	–	–
name	Name of the sliced channel.	–	–
regions	List of regions to be included in the sliced channel.	All regions	–
channelAxis	Device axis along the channel. The axis must be z.	z	–
minX	Minimum x-coordinate of the sliced channel.	–∞	µm
maxX	Maximum x-coordinate of the sliced channel.	∞	µm
minY	Minimum y-coordinate of the sliced channel.	–∞	µm
maxY	Maximum y-coordinate of the sliced channel.	∞	µm
minZ	Minimum z-coordinate of the sliced channel.	–∞	µm
maxZ	Maximum z-coordinate of the sliced channel.	∞	µm

Description

The `Math slicedChannel` command creates a sliced channel within the 3D device that has been already loaded using the `LoadDevice` command. This command automatically creates the 2D slices along the channel as well as a nonlocal area for each slice. The name of the nonlocal area is set automatically to `<slicedChannel>.NL`. For example, if the sliced channel is named `channel1`, the nonlocal will be named `channel.NL`. As in the `Math nonlocal` command, only mesh elements contained within the specified geometry are included.

Examples

```
Math slicedChannel name=channel1 regions=[list si1 ox1]
```

This command created a new sliced channel named `channel1`. Slices along the channel axis, which defaults to z, are extracted automatically within the `si1` and `ox1` regions.

Physics contact

Sets the properties of the specified contact.

Syntax

```
Physics contact=String [resist=Double] [type=String]
```

Arguments

Argument	Description	Default	Unit
`contact`	Name of the contact.	–	–
`resist`	Lumped resistance.	0.0	Ω
`type`	Type of boundary condition used for the Poisson equation at the contact. Specify `Neumann` to set Neumann boundary condition.	Dirichlet	–

Description

The `Physics contact` command sets the properties of the specified contact. For the Subband-BTE solver, the `type` argument can be used to set the required Neumann boundary condition for the solution of the subband-BTE on the left and right contacts of each sliced channel. A lumped resistance can be specified for a contact through which current flows using the `resist` argument.

Examples

```
Physics contact=source type=Neumann
Physics contact=drain  type=Neumann
```

These commands set the boundary condition type on the source and drain contacts to Neumann.

Physics slicedChannel

Specifies the Schrödinger solver or the Subband-BTE solver to use on a sliced channel.

Syntax

To specify the Schrödinger solver to use on all slices in a sliced channel, use:

```
Physics slicedChannel=String eSchrodinger=String ...
```

To specify the Subband-BTE solver to use on a sliced channel, use:

```
Physics slicedChannel=String (eBTESolver=String | hBTESolver=String) \
   [approximateJacobian=Boolean] [deltaE=Double] [minDeltaE=Double] \
   [NkForScreening=Integer] [reorderDispersion=Boolean] \
   [resolveTOBEnergy=Boolean] [subbandScatteringRange=Integer] \
   [tau=Double] [tdfDegenTol=Double] \
   [useKdependentWF=Boolean] [useKdependentWFForBTEDensity=Boolean] \
   [useParabolicFitForToB=Boolean] [whenToComputeScreening=String]
```

Arguments

Argument	Description	Default	Unit
`slicedChannel`	Name of the sliced channel.	–	–
`eSchrodinger`	Name of the Schrödinger solver to use on all slices in the sliced channel. For the Subband-BTE solver, this must be `Parabolic`.	–	–
`hSchrodinger`	Name of the Schrödinger solver to use on all slices in the sliced channel. For the Subband-BTE solver, available options are `6kp`, `3kp`, and `Parabolic`.	–	–
`eBTESolver`	The type of Subband-BTE solver to use for solving the subband-BTE on the sliced channel. Options are `AnalyticBallistic` and `Numeric`.	Numeric	–
`hBTESolver`	The type of Subband-BTE solver to use for solving the subband-BTE on the sliced channel. Options are `AnalyticBallistic` and `Numeric`.	Numeric	–
`approximateJacobian`	Specifies whether an approximate Jacobian will be used to solve the BTE, thereby decreasing memory consumption.	1	–
`deltaE`	Nominal energy grid spacing.	4.0e-3	eV
`minDeltaE`	When the top of the barrier for each subband is refined, this argument specifies the minimum energy spacing to use.	1.0e-4	eV

Argument	Description	Default	Unit
`NkForScreening`	The number of k -points to use when evaluating screening.	32	–
`reorderDispersion`	The default order of the subbands for each k -point is based on the energy value. If reordering of the subband dispersion is switched on, the order of the subbands is determined through wavefunction overlap in k -space. Options are: • `0` – Do not reorder subband dispersion. • `1` – Reorder subband dispersion.	0	–
`resolveTOBEnergy`	Specifies whether the energy grid near the top of each subband barrier is refined. Options are: • `0` – Do not refine • `1` – Refine	1	–
`subbandScatteringRange`	Specifies the allowed subband scattering range between the initial and final subbands. A value less than 0 indicates that scattering to all valid subbands is allowed. This is the case by default.	–1	–
`tau`	Relaxation time for the built-in RTA scattering model.	1.0e-10	s
`tdfDegenTol`	When you specify a positive value, this value sets the subband energy difference tolerance for screening inter-subband transitions with the tensor Lindhard screening model. When you specify a negative value, a special algorithm is used to automatically select the inter-subband transitions that are treated.	–1	eV
`useKdependentWF`	This argument is forwarded to the Schrödinger solver on each slice and specifies that the wavefunctions should be computed at each k -point in the k -space grid. Options are: • `0` – Only evaluate wavefunctions at the subband minimum. • `1` – Evaluate wavefunctions at each k -point.	0	–
`useKdependentWFForBTEDensity`	Specifies whether k -dependent wavefunctions will be used to compute the carrier density.	0	–
`useParabolicFitForToB`	Specifies whether a special algorithm will be used to more accurately locate the top of the source–drain barrier.	1	–
`whenToComputeScreening`	Controls when the selected screening model is computed during a bias solve. Options are: • `EveryIteration` – Compute for every Poisson Newton iteration • `FirstIteration` – Compute only during the first Poisson Newton iteration	Every Iteration	–

Description

The `Physics slicedChannel` command is used to specify the Schrödinger solver or the Subband-BTE solver to use on a sliced channel, depending on which parameters are used.

To specify the Schrödinger solver to use on all slices in a sliced channel, use the `eSchrodinger` or `hSchrodinger` argument. The use of this argument is identical to setting up a Schrödinger solver using the `Physics nonlocal` command, and all of the Schrödinger-related arguments can be used here as well.

To specify the Subband-BTE solver to use on a sliced channel, use the `eBTESolver` or `hBTESolver` argument.

Examples

```
Physics slicedChannel=channel1 eSchrodinger=Parabolic \
  valleys=[list Delta1 Delta2 Delta3] Nsubbands=8 Nk=16 Kmax=0.6 a0=5.43e-8
```

This command specifies that a parabolic Schrödinger solver must be used on all slices of the sliced channel named `channel1`. The usual set of arguments for setting up a Schrödinger solver is specified as well.

```
Physics slicedChannel=channel1 eBTESolver=AnalyticBallistic deltaE=5.0e-3
```

This command specifies that the Subband-BTE solver for `channel1` should change to the `AnalyticBallistic` Subband-BTE solver and that the nominal energy grid spacing should change to 5 meV.

Saving TDR Files

Several types of data and TDR files can be saved from Sentaurus Device QTX. Each type of TDR file is saved using a command specific to the type of data as described here.

Specifying Subbands and Subband IDs

Many of the quantities are defined on a subband basis. Therefore, you must specify exactly which subband and which quantity are required.

A particular subband is uniquely specified by its so-called subband ID, which is determined by the valley name of the subband and its subband index within this valley, separated by an underscore (_). For example, the subband ID of `Delta1_0` corresponds to the 0th subband of the `Delta1` valley.

NOTE Subband indexing starts from 0.

Therefore, a particular subband-based quantity is specified by giving the subband ID and then the quantity name, again separated by an underscore. For example, the name `Delta1_0_DistributionFunction` specifies the distribution function for the 0th subband of the `Delta1` valley.

In the following command sections, various subband-based quantities are described using names such as `<subbandID>_DistributionFunction`. Here, `<subbandID>` should be replaced by the subband ID such as `Delta1_0`.

BTESaveE

Saves the solution of the subband-BTE performed over the channel coordinate–energy space.

Syntax

```
BTESaveE tdrFile=String models=List [slicedChannel=String]
```

Arguments

Argument	Description	Default	Unit
tdrFile	Name of the TDR file to save.	–	–
models	List of the models or quantities to save.	–	–
slicedChannel	Name of the sliced channel from which the quantities will be extracted.	First sliced channel defined	–

The available quantities that can be included in the list of `models` to save are:

Quantity	Description	Unit
ChannelCoord	The position along the channel. Saved by default.	µm
Energy	The total carrier energy. Saved by default.	eV
<subbandID>_BackwardNetScatteringRate	The net in-scatter minus out-scatter rate into backward-going states.	s^{-1}
<subbandID>_CurrentSpectrum	The energy-resolved current spectrum used to compute the current.	$A*eV^{-1}$
<subbandID>_DensitySpectrum	The energy-resolved density spectrum used to compute the inversion density.	$(eV*cm)^{-1}$

`<subbandID>_DistributionFunction`	The carrier distribution function used to compute the carrier density.	1
`<subbandID>_DOS`	Density-of-states (DOS) per spin and direction.	$(eV*cm)^{-1}$
`<subbandID>_ForwardNetScatteringRate`	The net in-scatter minus out-scatter rate into forward-going states.	s^{-1}
`<subbandID>_SubbandEnergy`	Zero contour gives subband energy along the channel coordinate.	1

Description

The solution of the subband-BTE is performed over the channel coordinate–energy space. A few major quantities are solved or computed on this grid on a per-subband basis. The `BTESaveE` command allows you to save a 2D TDR file over the channel coordinate–energy space for these quantities.

For quantities other than `SubbandEnergy`, the `subbandID` can be used to save a quantity either per subband, or per valley, or as the total value summed over all subbands. To save a quantity per valley, the `subbandID` must contain only the valley name. To save the total value of a quantity, do not specify the `subbandID`. For example:

- To save the `DensitySpectrum` for the `Delta1_0` subband, specify:

  ```
  Delta1_0_DensitySpectrum
  ```

- To save the sum of the `DensitySpectrum` for all subbands in the `Delta1` valley, specify:

  ```
  Delta1_DensitySpectrum
  ```

- To save the sum of the `DensitySpectrum` over all subbands, specify:

  ```
  DensitySpectrum
  ```

Examples

```
BTESaveE tdrFile=FieldsOver_CC_E.tdr \
   models=[list Delta1_0_DistributionFunction Delta1_0_CurrentSpectrum]
```

This command saves a TDR file named `FieldsOver_CC_E.tdr` containing the distribution function and the current spectrum for the `Delta1_0` subband.

BTESaveK

Saves a 2D TDR file over the channel coordinate–k-space for the specified quantities.

Syntax

```
BTESaveK tdrFile=String models=List [slicedChannel=String] [Nk=Integer] \
   [Kmax=Double]
```

Arguments

Argument	Description	Default	Unit
tdrFile	Name of the TDR file to save.	–	–
models	List of the models or quantities to save.	–	–
slicedChannel	Name of the sliced channel from which the quantities will be extracted.	First sliced channel defined	–
Nk	The number of k-points to use in the k-grid.	101	1
Kmax	The k-grid is created for points between -Kmax and Kmax.	0.3	$\frac{2\pi}{a_0}$

The available quantities that can be included in the list of models to save are:

Quantity	Description	Unit
ChannelCoord	The position along the channel. Saved by default.	μm
k	The k-value along the k-grid. Saved by default.	$\frac{2\pi}{a_0}$
<subbandID>_Dispersion	The subband dispersion.	eV
<subbandID>_DistributionFunction	The carrier distribution function used to compute the carrier density.	1

Description

When the solution of the subband-BTE is performed over the channel coordinate–energy space, it is sometimes helpful to visualize some quantities over the channel coordinate–k-space instead, in particular, the distribution function. The BTESaveK command saves a 2D TDR file over channel coordinate–k-space for the specified quantities. The Nk and Kmax arguments are provided to specify the k-grid that should be used.

Examples

```
BTESaveK tdrFile=FieldsOver_CC_K.tdr \
   models=[list Delta1_0_DistributionFunction] Nk=101 Kmax=0.3
```

This command saves a TDR file named `FieldsOver_CC_K.tdr` containing the distribution function for the `Delta1_0` subband over channel coordinate–k-space. The k-grid that is saved extends from $-0.3*2\pi/a_0$ to $0.3*2\pi/a_0$ with 101 points.

BTESaveCC

Saves quantities from the solution of the subband-BTE that depend only on the channel coordinate.

Syntax

```
BTESaveCC tdrFile=String models=List [slicedChannel=String]
```

Arguments

Argument	Description	Default	Unit
`tdrFile`	Name of the TDR file to save.	–	–
`models`	List of the models or quantities to save.	–	–
`slicedChannel`	Name of the sliced channel from which the quantities will be extracted.	First sliced channel defined	–

The available quantities that can be included in the list of `models` to save are:

Quantity	Description	Unit
`BackwardCurrentTotal`	Total current flowing in the backward direction.	A
`ChannelCoord`	The position along the channel. Saved by default.	µm
`CurrentTotal`	Total current.	A
`DopingConcentration`	The integral of the doping concentration over each slice.	cm^{-1}
`ForwardCurrentTotal`	Total current flowing in the forward direction.	A
`NinvTotal`	Total inversion charge.	cm^{-1}
`VelocityTotal`	Total carrier velocity.	cm/s
`<subbandID>_BackwardCurrent`	Current flowing in the backward direction.	A
`<subbandID>_Current`	Current.	A
`<subbandID>_Efleq`	Local Fermi level for the RTA scattering model.	eV
`<subbandID>_ForwardCurrent`	Current flowing in the forward direction.	A
`<subbandID>_Ninv`	Inversion charge.	cm^{-1}
`<subbandID>_Occupancy`	Subband occupancy.	1

`<subbandID>_SubbandEnergy`	The minimum subband energy.	eV
`<subbandID>_Velocity`	Carrier velocity.	cm/s

Description

Several quantities from the solution of the subband-BTE depend only on the channel coordinate. These include the inversion charge, the current, and the carrier velocity. These quantities can be saved on a per-subband basis. In addition to the quantities mentioned, the total value from all subbands can be saved.

Examples

```
BTESaveCC tdrFile=FieldsOver_CC.tdr models=[list NinvTotal Delta1_0_Velocity]
```

This command saves a TDR (xy) file named `FieldsOver_CC.tdr` containing the total inversion charge along the channel as well as the velocity in the 0th subband of the `Delta1` valley. Since no sliced channel is specified, the first defined sliced channel is used automatically.

Saving Slice-Related Fields

Each 2D slice is like a 2D device with a 2D nonlocal area defined. Like a 2D device, real-space models over this 2D device can be saved as well as 1D k-space data similar to what is done for Schrödinger and mobility calculations on a single 2D device in Sentaurus Band Structure.

The model syntax is exactly the same. The key difference here is that a particular 2D slice from a sliced channel must be selected. A 2D slice has a unique *address* given by its sliced channel name and its channel coordinate. The user-specified `ChannelCoord` snaps to the nearest actual slice position.

SaveSlice

Saves real-space models over a 2D slice to a TDR file.

Syntax

```
SaveSlice tdrFile=String channelCoord=Double models=List \
   [slicedChannel=String]
```

Arguments

Argument	Description	Default	Unit
`tdrFile`	Name of the TDR file to save.	–	–
`channelCoord`	Channel coordinate of the required slice.The specified value snaps to the nearest slice position.	–	µm
`models`	List of the models or quantities to save. The usual set of real-space models over a 2D device can be specified.	–	–
`slicedChannel`	Name of the sliced channel from which the quantities will be extracted.	First sliced channel defined	–

Description

This command saves real-space models over the 2D slice specified with the `slicedChannel` and `channelCoord` arguments. The coordinates of the real-space mesh are saved in units of micrometer.

Examples

```
SaveSlice tdrFile=slice.tdr slicedChannel=channel1 channelCoord=20.0e-3 \
   models=[list ConductionBandEnergy Delta1_0_Wavefunction]
```

This command saves a 2D TDR file named `slice.tdr` containing the mesh and specified models over the 2D slice located in the sliced channel named `channel1` and closest to the slice channel coordinate of 20.0e-3 µm . Two models are saved: the relaxed conduction band energy and the norm of the wavefunction for the `Delta1_0` subband.

SaveSliceK

Saves a TDR (xy) file of k-space models over 1D k-space for the slice specified.

Syntax

```
SaveSliceK tdrFile=String channelCoord=Double models=List \
   [slicedChannel=String] [Nk=Integer] [Kmax=Double]
```

Arguments

Argument	Description	Default	Unit
tdrFile	Name of the TDR file to save.	–	–
channelCoord	Channel coordinate of the required slice. The specified value snaps to the nearest slice position.	–	μm
models	List of the models or quantities to save. Currently, only the dispersion can be saved.	–	–
slicedChannel	Name of the sliced channel from which the quantities will be extracted.	First sliced channel defined	–
Nk	The number of k-points to use in the k-grid.	101	1
Kmax	The k-grid is created for points between -Kmax and Kmax.	0.3	$\frac{2\pi}{a_0}$

Description

This command saves a TDR (xy) file of k-space models over 1D k-space for the slice specified with the slicedChannel and channelCoord arguments. The 1D k-space coordinates are saved in units of $2\pi/a_0$. The k-space grid that is used is specified using the Nk and Kmax arguments.

Examples

```
SaveSliceK tdrFile=slice_K.tdr slicedChannel=channel1 channelCoord=20.0e-3 \
   models=[list Delta1_0_Dispersion]
```

This command saves a TDR (xy) file named slice_K.tdr containing a 1D k-space grid and specified models over 1D k-space for the slice located in the sliced channel named channel1 and closest to the slice channel coordinate of 20.0e-3 μm. In this example, only the dispersion for the Delta1_0 subband is saved. The default values for Nk and Kmax are used.