

Sentaurus™ Visual User Guide

Version T-2022.03, March 2022

SYNOPSYS®

Copyright and Proprietary Information Notice

© 2022 Synopsys, Inc. This Synopsys software and all associated documentation are proprietary to Synopsys, Inc. and may only be used pursuant to the terms and conditions of a written license agreement with Synopsys, Inc. All other use, reproduction, modification, or distribution of the Synopsys software or the associated documentation is strictly prohibited.

Destination Control Statement

All technical data contained in this publication is subject to the export control laws of the United States of America. Disclosure to nationals of other countries contrary to United States law is prohibited. It is the reader's responsibility to determine the applicable regulations and to comply with them.

Disclaimer

SYNOPSYS, INC., AND ITS LICENSORS MAKE NO WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, WITH REGARD TO THIS MATERIAL, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE.

Trademarks

Synopsys and certain Synopsys product names are trademarks of Synopsys, as set forth at <https://www.synopsys.com/company/legal/trademarks-brands.html>.

All other product or company names may be trademarks of their respective owners.

Free and Open-Source Licensing Notices

If applicable, Free and Open-Source Software (FOSS) licensing notices are available in the product installation.

Third-Party Links

Any links to third-party websites included in this document are for your convenience only. Synopsys does not endorse and is not responsible for such websites and their practices, including privacy practices, availability, and content.

www.synopsys.com

Contents

Conventions	20
Customer Support	20
<hr/>	
1. Introduction to Sentaurus Visual	22
Functionality of Sentaurus Visual.	22
Starting Sentaurus Visual	22
From the Command Line	22
From Sentaurus Workbench	24
As a D-Bus Session	24
Accelerating the Rendering of Graphics	25
TCAD Sentaurus Tutorial: Simulation Projects	25
User Interface of Sentaurus Visual	26
Menu Bar	27
Toolbars	27
Plot Area.	28
Tcl or Python Console	28
Data Selection and Properties Panels	29
Quick Access to Tabs of Plot Properties and Axis Properties Panels	29
Filtering Names on the Data Selection Panel.	30
Interface to Sentaurus Process and Sentaurus Interconnect	32
Setting Up the Interface	35
Loading Command Files	37
Inserting and Deleting Breakpoints in the Flow	37
Inserting and Deleting Checkpoints in the Flow	38
Indicating Status of Steps	39
Updating the Structure	39
Multithreading Support.	39
<hr/>	
2. Basic Operations	41
Loading Files	41
Supported File Formats	41

Contents

Loading Scripts	42
Reloading Plot Files	43
Automatically Reloading Datasets	43
Managing Loaded Information	45
Customizing Settings	46
Creating Custom Buttons to Access Scripts	47
Working With Plots	48
Modes When Interacting With Plots	48
Common Modes	49
XY Plot–Only Modes	50
2D Plot–Only Modes	50
3D Plot–Only Modes	51
Linking Plots	51
Automatically Linking Plots	52
Undoing Operations	54
Displaying Multiple Plots	54
Grid Orientation	54
Vertical Orientation	56
Horizontal Orientation	57
Managing Frames	58
Drawing Inside Plots	59
Inserting Text	59
Drawing Rectangles	59
Drawing Ellipses	60
Drawing Lines	60
Exporting Plots	61
Exporting Movies	62
Starting a New Movie	62
Adding Frames in a Movie	62
Exporting a Movie	63
Printing Plots	64
Zooming and Panning	64
Zoom Tool	64
Reset Tool	64
Deleting Plots	64
Performance Options	65
Fast Draw (3D Plots Only)	65

Contents

Subsampling (2D and 3D Plots Only)	65
Advanced Options (XY Plots Only)	65
Advanced Options (2D and 3D Plots Only)	67
Selecting Log Files.....	69
3. Working With XY Plots	71
Loading XY Plots	71
Plotting One Curve.....	72
Plotting Multiple Curves	73
Visualizing Multiple TDR States.....	77
Cutline Plots.....	79
Curve Properties	80
Modifying Properties in Multiple Curves.....	81
Plot Area Properties.....	82
Legend Properties	83
Axis Properties.....	83
Changing the Axis Padding	84
Changing the Axis Precision	84
Duplicating XY Plots.....	85
Using Symbols and Scientific Notation in Plots	85
Best Look Option	89
Plotting Edges of Depletion Regions	89
Plotting Band Diagrams.....	91
Visualizing Discrete Traps in Band Diagrams	92
Visualizing Data as a Step-Like Plot	93
Saving the Plot to a Tcl File	93
Probing.....	94
Probe Options	94
Performing Complex Mathematical Operations on 1D Data	95
Performing Complex Mathematical Operations on Curves	96
Computing Electrical Characteristics	97
Exporting Data From Variables and Curves.....	97

Contents

4. Working With 2D and 3D Plots	100
Visualizing 2D and 3D Plots	100
Changing Plot Properties	101
Changing the Color Scheme for Materials	103
Displaying Region or Material Names	106
Visualizing Fields	107
Visualizing Fields Defined on Interface Regions	108
Visualizing Automatically Generated Regions	110
Junction Lines	110
Depletion Regions	110
Visualizing Multiple TDR States	111
Visualizing Regions With Multiple Parts	115
Visualizing Discrete Traps	117
3D View	119
Interacting With 3D Plots	121
Editing Camera Properties	122
2D View	123
Interacting With 2D Plots	123
Light Kit	124
Editing the Properties of Lights	126
Loading Options for Data	127
Rendering Options	129
Materials and Regions	130
Showing or Hiding Properties for Multiple Materials and Regions	130
Modifying Properties in Multiple Materials and Regions	131
Modifying the List of Initially Hidden Materials	132
Contact Regions	133
Contour Plots	136
Contour Legend Settings	136
Displaying Contour Plots	138
Converting Data to Nodal	138
Creating New Scalar Fields	139
Vector Plots	140
Importing an Image as a Background Field	141
Adjusting Magnification of an Image	142
Visualizing Point Boundary Conditions	144
Scaling and Shifting 2D and 3D Geometries	144
Rotating Structures (3D Plots Only)	145

Contents

Rotation Point	146
Customizing the Rotation Point	147
Using the Rotation Point as a Reference Point	147
Rotating Plots Using Exact Values	148
Overlays	150
Showing Differences Between Plots	152
Measuring Distances	153
Integration Tool	155
Using a Custom Integration Domain	157
Integrating Only a Defined Set of Regions or Materials	157
Probing	157
Accessing Dataset Information	161
Maximum and Minimum Locations of Fields	162
Changing Properties of Markers	165
Value Blanking	166
Choosing Constraints	167
Options for Value Blanking	169
Visualizing Deformation of Structures	171
Cutting Structures	172
Generating Precise Cutlines and Cutplanes	173
Cutlines in 2D Plots	176
Manipulating Cutlines	177
Polyline Cuts in 2D Plots	177
Manipulating the Polyline	179
Cutting Along Boundaries	179
Step 1: Selecting Regions or Materials	180
Step 2: Adding Vertex Points	181
Step 3: Choosing Segment Regions	182
Surface Cutlines From 3D Plots	184
Changing Properties of Cutline Along Boundaries	186
Two-Dimensional Projections	186
Cutplanes in 3D Plots	190
Extracting the Path of Minimum or Maximum Values of a Scalar Field	191
Surface Plots	197
Creating Surface Plots	198
Isosurfaces and Isolines	199

Contents

Creating Iso-Geometries	199
Modifying Iso-Geometries	201
Streamlines	202
Displaying Streamlines.....	203
Position Tab	203
Specifying Regions or Materials.....	204
Representing the Streamlines	204
Integration Settings	204
Integration Tab	205
Managing Created Streamlines	205
Configuring General Parameters of Streamlines	205
Extracting Data From Streamlines.....	206
5. Automated Tasks	209
Running Tcl or Python Scripts	209
Example: Plot Id–Vg Curve	209
Example: Create Cutline and Export Cutline Data to CSV File for Further Processing	211
Saving Command History	211
Running Inspect Command Files.....	211
Script Library	211
Restrictions	212
A. Tcl Commands	213
Syntax Conventions.....	213
Object Names: -name Argument	213
Common Properties.....	214
Colors.....	214
Fonts	215
Lines.....	216
Markers	216
add_custom_button.....	217
add_frame	218
calculate.....	219
calculate_field_value	220

Contents

calculate_scalar	221
create_curve	222
create_cut_boundary	223
create_cutline	224
create_cutplane	226
create_cutpolyline	227
create_cutpolyplanes	228
create_field	229
create_iso	230
create_plot	231
create_projection	232
create_projection_path	234
create_ruler	236
create_streamline	237
create_surface	239
create_variable	240
diff_plots	241
draw_ellipse	242
draw_line	243
draw_rectangle	244
draw_textbox	245
echo	246
exit	246
export_curves	247
export_movie	248
export_settings	249
export_variables	249
export_view	250
extract_path	251
extract_streamlines	253
extract_value	254

Contents

find_values.....	256
get_axis_prop	257
get_camera_prop.....	260
get_contour_labels_prop.....	261
get_curve_data	263
get_curve_prop	264
get_cutline_prop	266
get_cutplane_prop.....	268
get_ellipse_prop.....	269
get_field_prop	270
get_grid_prop.....	272
get_input_data.....	273
get_legend_prop	274
get_line_prop.....	276
get_material_prop	277
get_plot_prop.....	279
get_rectangle_prop	282
get_region_prop.....	283
get_ruler_prop	285
get_streamline_prop	286
get_textbox_prop.....	288
get_variable_data	290
get_vector_prop.....	291
get_vertical_lines_prop	293
help	294
import_settings.....	295
integrate_field	296
link_plots	298
list_curves	300
list_custom_buttons.....	301
list_cutlines	302

Contents

list_cutplanes	303
list_datasets	304
list_ellipses	305
list_fields	306
list_files	307
list_lines	308
list_materials	309
list_movie_frames	310
list_plots	310
list_rectangles	311
list_regions	312
list_streamlines	313
list_tdr_states	314
list_textboxes	315
list_variables	316
list_vertical_lines	317
load_file	318
load_file_datasets	319
load_library	320
load_script_file	321
move_plot	322
overlay_plots	323
probe_curve	324
probe_field	325
reload_datasets	326
reload_files	326
remove_curves	327
remove_custom_buttons	327
remove_cutlines	328
remove_cutplanes	328
remove_datasets	329

Contents

remove_ellipses	329
remove_lines	330
remove_plots	331
remove_rectangles	331
remove_rulers	332
remove_streamlines	333
remove_textboxes	334
render_mode	335
reset_settings	335
rotate_plot	336
save_plot_to_script	337
select_plots	338
set_axis_prop	339
set_band_diagram	343
set_best_look	343
set_camera_prop	344
set_contour_labels_prop	345
set_curve_prop	347
set_cutline_prop	349
set_cutplane_prop	350
set_deformation	351
set_ellipse_prop	352
set_field_prop	353
set_grid_prop	355
set_legend_prop	356
set_line_prop	359
set_material_prop	360
set_plot_prop	362
set_rectangle_prop	366
set_region_prop	367
set_ruler_prop	369

Contents

set_streamline_prop	370
set_tag_prop	372
set_textbox_prop	373
set_transformation	375
set_value_blanking	376
set_vector_prop	377
set_vertical_lines_prop	379
set_window_full	380
set_window_size	380
show_msg	381
start_movie	382
stop_movie	382
undo	383
unload_file	383
version	384
windows_style	385
zoom_plot	386
B. Python Commands	387
General Information	387
Accessing Documentation for Python Commands	387
Syntax Conventions	387
Get Property Commands	388
Set Property Commands	388
Common Properties	388
Colors	388
Fonts	390
Lines	390
Markers	391
C. Menus and Toolbars of User Interface	392
Menus	392
File Menu	392

Contents

Edit Menu	393
View Menu	393
Tools Menu	395
Data Menu	397
Window Menu	398
Help Menu	399
Toolbars	400
File Toolbar	400
Edit Toolbar	400
Draw Toolbar	400
View Toolbar	401
Tools Toolbar	401
Movies Toolbar	402
Look Toolbar	402
Additional Keyboard Shortcuts (2D and 3D Plots)	403
D. Available Formulas	405
Creating a New Variable	405
Creating a New Curve	405
Applying Functions to a Curve	406
Creating a New Field	407
Available Functions	407
E. Inspect Support in Sentaurus Visual	413
Fully Supported Commands	413
Partially Supported Commands	414
Not Supported Commands	415
Script Library Support	415
Extraction Library	415
Curve Comparison Library	415
The extend Library	416
F. Extraction Library	417
Syntax Conventions	417

Contents

Help for Procedures	419
Output of Procedures	419
ext::AbsList	421
ext::DiffForwardList	422
ext::DiffList	423
ext::ExtractBVi	424
ext::ExtractBVv	425
ext::ExtractEarlyV	427
ext::ExtractExtremum	428
ext::ExtractGm	429
ext::ExtractIoff	431
ext::ExtractRdiff	433
ext::ExtractRsh	435
ext::ExtractSS	437
ext::ExtractSsub	439
ext::ExtractValue	440
ext::ExtractVdlin	442
ext::ExtractVdlog	443
ext::ExtractVglin	444
ext::ExtractVglog	446
ext::ExtractVtgm	447
ext::ExtractVti	448
ext::ExtractVtsat	450
ext::FilterTable	451
ext::FindExtrema	455
ext::FindVals	456
ext::LinFit	458
ext::Linspace	460
ext::LinTransList	461
ext::Log10List	462
ext::RemoveDuplicates	463

Contents

ext::RemoveZeros	464
ext::SubLists	465
lib::SetInfoDef	467
References	467
G. Impedance Field Method Data Postprocessing Library	468
Overview	468
Syntax Conventions	469
Help for Procedures	470
Output of Procedures	471
ifm::Gauss	472
ifm::GetDataQuantiles	473
ifm::GetGaussian	474
ifm::GetHistogram	476
ifm::GetMoments	477
ifm::GetMOSIVs	478
ifm::GetMOSWeights	483
ifm::GetNoiseStdDev	484
ifm::GetQQ	486
ifm::GetsIFMStdDev	487
ifm::GetSNM	488
ifm::GetSRAMVTC	490
ifm::ReadCSV	494
ifm::ReadsIFM	495
ifm::WriteCSV	498
lib::SetInfoDef	499
H. Two-Port Network RF Extraction Library	500
Syntax Conventions	501
Help for Procedures	503
Output of Procedures	503

Contents

Overview of RF Extraction Library Procedures	503
Equations Used in RF Extraction Library	506
A-Matrix, C-Matrix, and Y-Matrix	506
Tcl Arrays rfx::AC and rfx::Y	507
Power Spectral Density Matrices	507
Power Spectral Densities	508
PSDs Computed by Sentaurus Device and RF Extraction Library	510
Power Spectral Density Tcl Arrays	511
Device Width Scaling for 2D Structures	512
Matrix Conversions	513
Converting Y-Matrix to h-Matrix	513
Converting Y-Matrix to S-Matrix	513
Converting Y-Matrix to Z-Matrix	514
Gains, Amplifier Stability, and Unilateralization	514
Small-Signal Current Gain	514
Amplifier Stability	514
Maximum Stable Gain and Maximum Available Gain	515
Unilateral Amplifier Design	515
Converting Gain Units to Decibels	515
Transistor Figures of Merit	516
ft and fmax	516
Extraction Methods for ft and fmax	517
Cutoff Frequency for Stability	519
Noise Figure of a Linear Two-Port Network	519
rfx Namespace Variables	523
Characteristic Impedance and Source Impedance	525
rfx::CreateDataset	526
rfx::Export	531
rfx::GetFK1	532
rfx::GetFmax	534
rfx::GetFt	536
rfx::GetNearestIndex	538
rfx::GetNoiseFigure	540
rfx::GetParsAtPoint	543
rfx::GetPowerGain	544
rfx::Load	547
rfx::NoiseFigure	549

Contents

rfx::PolarBackdrop	551
rfx::PowerGain	552
rfx::RFCList	553
rfx::SmithBackdrop.	555
rfx::Y2H	556
rfx::Y2S	557
rfx::Y2Z	558
Complex Arithmetic Support	559
rfx::Abs_c	560
rfx::Abs_v	561
rfx::Abs2_c	562
rfx::Abs2_v	562
rfx::Add_c	563
rfx::Add_v	564
rfx::Cart2Polar_c	565
rfx::Cart2Polar_v	565
rfx::Conj_c	566
rfx::Conj_v	567
rfx::Div_c	568
rfx::Div_v	568
rfx::Im_c	569
rfx::Mul_c	570
rfx::Mul_v	570
rfx::Mulsc_c	571
rfx::Phase_c	572
rfx::Phase_v	573
rfx::Polar2Cart_c	574
rfx::Polar2Cart_v	574
rfx::Re_c.	575
rfx::Sign	576
rfx::Sub_c.	577
rfx::Sub_v	577
lib::SetInfoDef	579
References.	579

Contents

I. PhysicalConstants Library	581
Major Physical Constants	581
References	583

About This Guide

The Synopsys® Sentaurus™ Visual tool is part of Sentaurus Workbench Visualization. It is a plotting software for visualizing data from simulations and experiments. Sentaurus Visual enables users to work interactively with data using both a user interface and a scripting language for automated tasks.

For additional information, see:

- The TCAD Sentaurus release notes, available on the Synopsys SolvNetPlus support site (see [Accessing SolvNetPlus on page 21](#))
 - Documentation available on the SolvNetPlus support site
-

Conventions

The following conventions are used in Synopsys documentation.

Convention	Description
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.
<i>Italicized text</i>	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.
Key+Key	Indicates keyboard actions, for example, Ctrl+I (press the I key while pressing the Control key).
Menu > Command	Indicates a menu command, for example, File > New (from the File menu, choose New).

Customer Support

Customer support is available through the Synopsys SolvNetPlus support site and by contacting the Synopsys support center.

Accessing SolvNetPlus

The SolvNetPlus 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 SolvNetPlus site:

1. Go to <https://solvnetplus.synopsys.com>.
 2. Enter your user name and password. (If you do not have a Synopsys user name and password, follow the instructions to register.)
-

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

Contacting Your Local TCAD Support Team Directly

Send an email 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

1

Introduction to Sentaurus Visual

This chapter presents basic information about Sentaurus Visual.

Functionality of Sentaurus Visual

Sentaurus Visual allows you to visualize complex simulation results generated by physical simulation tools in one, two and three dimensions. You can visualize data for an initial understanding and analysis, and then modify the plots to gain a new perspective.

Sentaurus Visual can be used to create plots that display fields, geometries, and regions, including results such as p-n junctions and depletion layers. It also allows you to view I-V curves and doping profiles, and provides tools to zoom, pan, and rotate images. You also can extract data using measure and probe tools.

The user interface provides direct and easy-to-use functionality, as well as advanced controls for expert users. With the user interface of Sentaurus Visual, you can systematically visualize devices as xy, 2D, and 3D plots.

Starting Sentaurus Visual

Sentaurus Visual can be started either from the command line or from Sentaurus Workbench.

From the Command Line

To start Sentaurus Visual from the command line, enter:

```
svisual
```

The following example loads the dataset associated with a file and generates its plot:

```
svisual n2_fps.tdr
```

Chapter 1: Introduction to Sentaurus Visual

Starting Sentaurus Visual

When starting Sentaurus Visual from the command line, you can use the following options (which are displayed by entering `svisual -h`):

Usage: `svisual [options] [FILES]`

Description:

Sentaurus Visual is a tool to display and analyze structures and curves.

Options:

<code>-h[elp]</code>	: Display this help message.
<code>-v[ersion]</code>	: Print the Sentaurus Visual version.
<code>-m[esa]</code>	: Run Sentaurus Visual with Mesa driver.
<code>-glx</code>	: Run Sentaurus Visual with GLX driver.
<code>-vgl</code>	: Run Sentaurus Visual in VirtualGL environment.
<code>-noavx</code>	: Run Sentaurus Visual without AVX support (by default, AVX support is active when Mesa is switched on).
<code>-b[atch]</code>	: Run in pure batch mode (requires a script file).
<code>-batchx -bx</code>	: Run in batch mode that allows export of graphics (virtual X server, requires a script file).
<code>-retry_count <integer></code>	: Number of attempts to find a free virtual X screen buffer. Specify 0 to retry indefinitely.
<code>-retry_time <integer></code>	: The maximum time to wait (in seconds) for a free virtual X screen buffer to become available.
<code>-diagnostics</code>	: Execute the diagnostics process, analyzing the environment, the system, and the setup.
<code>-p[ython]</code>	: Run Sentaurus Visual in Python mode, overriding user preferences.
<code>-t[c1]</code>	: Run Sentaurus Visual in Tcl mode, overriding user preferences.
<code>-s[cript] <string></code>	: Execute listed files as Sentaurus Visual scripts in Tcl or Python mode, according to user preferences, or <code>-t[c1]</code> or <code>-p[ython]</code> option.
<code>-i[nspect] -f <string></code>	: Execute listed files as Inspect scripts.
<code>-librarypath <string></code>	: Add library path to default path.
<code>-nolibrary</code>	: Switch off Tcl library auto-loading.
<code>-nowait</code>	: Do not wait for a license to become available.
<code>-verbose</code>	: Print every Tcl or Python command executed to the log file.
<code>-slowscript -ss</code>	: Redraw plots automatically after each command (script execution is slower).
<code>-spi</code>	: Launch Sentaurus Process interface.
<code>-geoms <string></code>	: Load only the geometries listed.
<code>-alldata</code>	: Load all data from a file, overriding user preferences.

Chapter 1: Introduction to Sentaurus Visual

Starting Sentaurus Visual

```
--threads <integer>      : Number of threads to use, overriding user
                             preferences. Specify 0 to yield automatic
                             detection.
--max_threads <integer>   : Maximum number of threads to use, overriding
                             --threads option and user preferences if
                             they are higher. Specify 0 to yield
                             automatic detection.
```

Note:

Sentaurus Visual can run solely in batch mode, that is, no display is required and scripts can be run using a shell. This mode is fast but has some disadvantages, for example, exporting graphics only works in the GUI mode. To overcome this, use the `-batchx` option.

From Sentaurus Workbench

Sentaurus Visual is integrated in Sentaurus Workbench. You can start Sentaurus Visual in one of the following ways:

- Click a node, which displays the Node Explorer. In the Node Explorer, in the **Viewer** box, select **svisual** and click the **Launch** button next to it.
- Click the Visualize toolbar button and select **Sentaurus Visual**.

Sentaurus Visual can receive node data and can be inserted into tool flows.

Note:

Sentaurus Visual can run in batch mode (`-b` option), which is especially useful when used within tool flows. In this context, the use of macro files is also of interest (see [Chapter 5 on page 209](#)).

As a D-Bus Session

Sentaurus Visual can be run in a special mode in which it creates its own D-Bus session. This feature is useful, for example, for certain grid configurations that cannot close D-Bus sessions, even if Sentaurus Visual has already exited.

To switch on this feature, set the `SVISUAL_DBUS_RUN_SESSION` environment variable to 1 before starting Sentaurus Visual as follows:

```
# For C shell
setenv SVISUAL_DBUS_RUN_SESSION 1

# For bash
export SVISUAL_DBUS_RUN_SESSION=1
```

Accelerating the Rendering of Graphics

In Sentaurus Visual, 2D and 3D plots are rendered using OpenGL® acceleration, which can produce significant differences in performance depending on the configuration of the machine where Sentaurus Visual runs.

By default, Sentaurus Visual always runs in the best supported graphics mode it can find, using the graphics card of the machine to render plots. If Sentaurus Visual cannot detect a graphics card, then it searches for a valid VirtualGL environment and uses it if available. If there are no compliant renderers, then Sentaurus Visual reverts to a generic Mesa driver for graphics rendering.

If your machine has a graphics card or VirtualGL environment but Sentaurus Visual runs with Mesa rendering, then you can force Sentaurus Visual to run with a GLX driver by using the `-glx` option or to run in a VirtualGL environment by using the `-vgl` option. If Sentaurus Visual cannot find a GLX driver or a valid VirtualGL environment, it exits with an error message. To force Mesa rendering, specify the `-mesa` option.

When using Mesa rendering, Sentaurus Visual automatically detects whether it can utilize Advanced Vector Extensions (AVX) instructions to speed up visualization. AVX significantly increases rendering performance when using a Mesa driver. To force Mesa rendering without the use of AVX, specify the `-noavx` option.

Note:

You cannot use both the `-glx` and `-mesa` options simultaneously.

TCAD Sentaurus Tutorial: Simulation Projects

The TCAD Sentaurus Tutorial provides various projects demonstrating the capabilities of Sentaurus Visual.

To access the TCAD Sentaurus Tutorial:

1. Open Sentaurus Workbench by entering the following on the command line: `swb`
2. From the menu bar of Sentaurus Workbench, choose **Help > Training** or click  on the toolbar.

Alternatively, to access the TCAD Sentaurus Tutorial:

1. Go to the `$STROOT/tcad/current/Sentaurus_Training` directory.

The `STROOT` environment variable indicates where the Synopsys TCAD distribution has been installed.

2. Open the `index.html` file in your browser.

Chapter 1: Introduction to Sentaurus Visual

User Interface of Sentaurus Visual

User Interface of Sentaurus Visual

The user interface of Sentaurus Visual has different areas (see [Figure 1](#)).

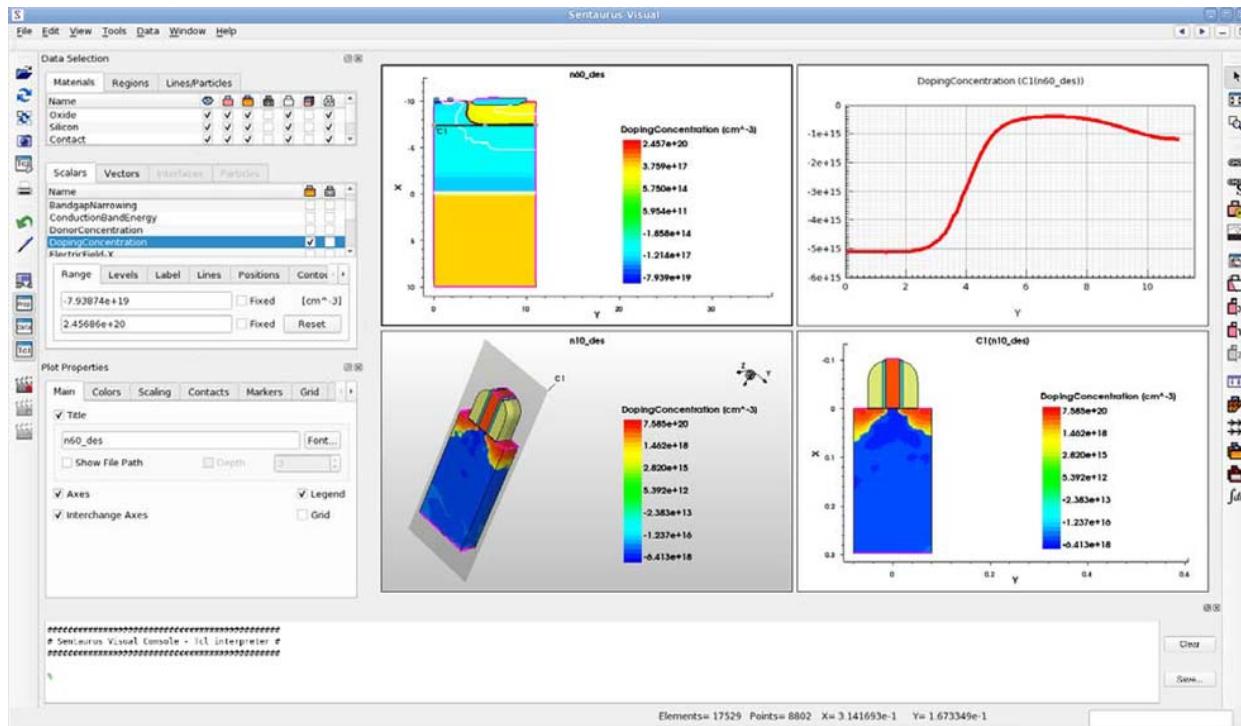
The Data Selection panel and properties panel are located to the left of the main window, the plot area displays the different visualizations, the Tcl or Python Console is in the lower part, and the toolbars are located on the sides of the main window.

For detailed information about menus and toolbars, see [Appendix C on page 392](#).

Note:

You can customize the user interface of Sentaurus Visual. Different options are available, for example, you can detach the panels, adjust their size, and move the toolbars to another part of the main window.

Figure 1 Main window of Sentaurus Visual showing different plots in the plot area



Menu Bar

You use the menu bar to access the main operations of Sentaurus Visual such as opening files, showing and hiding toolbars, configuring Sentaurus Visual, manipulating loaded data, and organizing plots in the plot area.

Table 1 Menus available from user interface

Menu	Description
File	Loads plots and scripts, reloads data, and exports and prints plots.
Edit	Selects plots, and selects settings for Sentaurus Visual.
View	Shows and hides toolbars and panels; plot settings and performance options.
Tools	Accesses analysis tools.
Data	Views loaded datasets, and deletes selected plots.
Window	Organizes and manages active plots.
Help	Provides information about Sentaurus Visual.

Toolbars

Toolbars offer quick access to commonly used functions that are also available from the different menus (see [Toolbars on page 400](#)).

Note:

One toolbar is always visible to allow you to show or hide the Tcl or Python Console and to organize the data selection and properties panels into tabs.

Table 2 Toolbars

Toolbar	Description
File	Loads plots and scripts, reloads data, and exports and prints plots.
Edit	Undoes operations, and displays toolbar for drawing shapes and inserting text onto plots.
View	Accesses zoom operations and subsampling.
Tools	Accesses analysis tools.

Table 2 Toolbars (Continued)

Toolbar	Description
Custom Buttons	Accesses custom buttons. See Creating Custom Buttons to Access Scripts on page 47 .
Movies	Records animated images.
Look	Shows or hides panels.

Plot Area

The plot area displays the active plots. You can select or deselect plots as follows:

- To select one plot, click inside the plot.
- To select multiple plots, hold the Shift key while you click inside the plots.
- If you click a selected plot while holding the Shift key, then that plot is deselected.

Toolbars and panels change depending on the types of plot selected.

Tcl or Python Console

The Tcl or Python Console shows information about the commands used to manipulate and display data in Sentaurus Visual. The Console has different areas:

- The main pane allows you to input the available commands from the respective interpreter, to output the command results, and to register the history of every action performed in the Sentaurus Visual GUI since the session started.

You can enter commands manually at the command prompt. This is helpful when running complex calculations on datasets and displaying results.
- On the right side, click **Clear** to delete the command history from the main pane, and click **Save** to store everything that was executed into a script file, so that it can be run without repeating all the operations.

For details about Tcl commands and scripting, see [Chapter 5 on page 209](#) and [Appendix A on page 213](#).

For details about Python commands, see [Appendix B on page 387](#).

Note:

The Python mode is activated when starting Sentaurus Visual with the `-p` or `-python` command-line option.

Data Selection and Properties Panels

Two panels are located by default at the left side of the main window:

- The Data Selection panel displays the data to visualize and shows which data is already displayed. This panel differs for xy plots, and 2D and 3D plots. The differences are explained in [Chapter 3 on page 71](#) and [Chapter 4 on page 100](#).
- The Properties panel lists the selected object properties. By default, it shows the plot properties. It will change after an object is selected. This panel can also be displayed (if it is arranged behind the Data Selection panel or if it is hidden) at its last position in the main window if you double-click the object.

If the plot area is empty (no plots are created or all plots are hidden), then these panels are always hidden. After a plot is created, by default, both panels open even if both were closed by the user in the last session.

To change this default behavior:

1. Choose **Edit > Preferences**.
The User Preferences dialog box is displayed.
2. Expand **Application > Common**.
3. Under Force Panels to Show, deselect **Data Selection Panel** or **Properties Panel** or both options.
4. Click **Save**.

Quick Access to Tabs of Plot Properties and Axis Properties Panels

Note:

The quick access operation for axes applies to xy and 2D plots only.

The Plot Properties panel or the Axis Properties panel *must be already open* for these quick access operations to work. To open the Plot Properties panel, double-click an empty part of the plot if another panel is active. To open the Axis Properties panel, double-click any axis in the plot area.

You can quickly access different tabs of the Plot Properties panel or the Axis Properties panel by clicking particular parts of a plot in the plot area:

- Click the plot title to display the **Main** tab of the Plot Properties panel.
- Click an axis title (for example, X) to display the **Title/Scale** tab of the Axis Properties panel.

Chapter 1: Introduction to Sentaurus Visual

User Interface of Sentaurus Visual

- Click any tick label on an axis (for example, -5) to display the **Ticks** tab of the Axis Properties panel. This operation applies only to 2D plots.
- Click an axis line to display the **Main** tab of the Axis Properties panel.

Filtering Names on the Data Selection Panel

You can filter the visible names of the Data Selection panel and xy plot data selection. This works for all data selection items, such as materials, region, lines/particles, or fields such as scalars or vectors.

The filter box opens when you start typing with the focus on the Data Selection panel. It automatically filters the view of the selection based on the filter expression. It supports case sensitivity, wildcards, and regular expression filtering.

The filter box is available in the Data Selection panel for the selection of rendering options for materials, regions, fields, and so on. It is also available for the data selection of xy plots.

The filter box contains the following buttons:

- Click the  button to activate case sensitive filtering.
- Click the  button to activate regular expression filtering. When it is deactivated, wildcard filtering is used.
- Click the  button to clear the filter expression.

You can use wildcard or regular expression filtering. Pressing the Esc key clears any active filter and shows all items again; any selected items remain selected. Pressing the Esc key again clears the selection.

Wildcard Filtering

Wildcard filtering is based on command shell filtering to filter files. Wildcard filtering is simpler than full regular expression filtering. Wildcard filtering is selected when the regular expression button () is deselected.

Syntax	Description
c	Any character represents itself apart from those mentioned in the following table entries. Therefore, c matches the character c. Note: If the character c appears at the start of the filter expression, then the regular expression only matches from the start.
?	Matches any single character. It is the same as . in full regular expressions.

Chapter 1: Introduction to Sentaurs Visual

User Interface of Sentaurs Visual

Syntax	Description
*	Matches zero or more characters, of any character. It is the same as <code>.*</code> in full regular expressions.
[...]	Sets of characters can be represented in brackets, similar to full regular expressions. Within the character class, the backslash has no special meaning.

Regular Expression Filtering

For most use cases, wildcard filtering should be sufficient. However, if you want more control, then you can use regular expressions. The regular expression is modeled on the Perl regexp language. Regular expressions are built from expressions, quantifiers, and assertions.

The simplest expression is a character, for example, `x` or `5`. An expression can also be a set of characters enclosed in brackets. For example, `[ABCD]` will match an `A`, or a `B`, or a `C`, or a `D`. You can write this same expression as `[A-D]`, and an expression to match any capital letter in the English alphabet is written as `[A-Z]`.

You use the vertical bar (`|`), which means *or*, to include multiple words for matching. For example, you might want to find all fields containing the words *band gap*. You enclose the expression in parentheses `(band|gap)` to group expressions together, and they identify a part of the regular expression that you want to capture. Enclosing the expression in parentheses allows you to use it as a component in more complex regular expressions.

Note:

A filter expression starting with a character `c`, which is not part of the syntax, results in the regular expression only matching from the start. This has been done to allow for a more intuitive search experience, but it differs from standard regular expression filtering.

A *quantifier* specifies the number of occurrences of an expression that must be matched. For example:

- `x{1,1}` means match one and only one `x`.
- `x{1,5}` means match a sequence of `x` characters that contains at least one `x` but no more than five.

Suppose you want a regular expression to match integers in the range from 0 to 99. At least one digit is required, so you start with the expression `[0-9]{1,1}`, which matches a single digit exactly once.

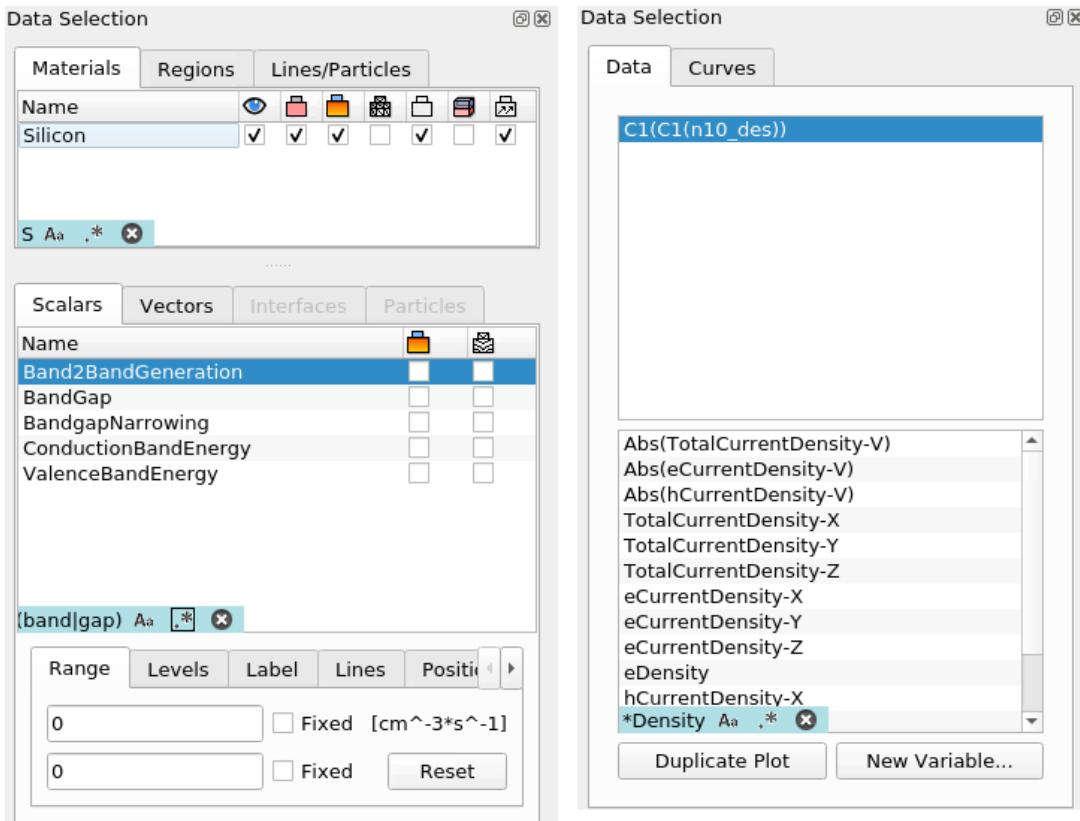
With regard to *assertions*, when `$` is the last character of a regular expression, it means the regular expression must match to the end of the string.

Chapter 1: Introduction to Sentaurus Visual

Interface to Sentaurus Process and Sentaurus Interconnect

For more information about regular expressions, see J. E. F. Friedl, *Mastering Regular Expressions*, Sebastopol, California: O'Reilly Media, 3rd ed., 2006.

Figure 2 Data Selection panel showing filtered names



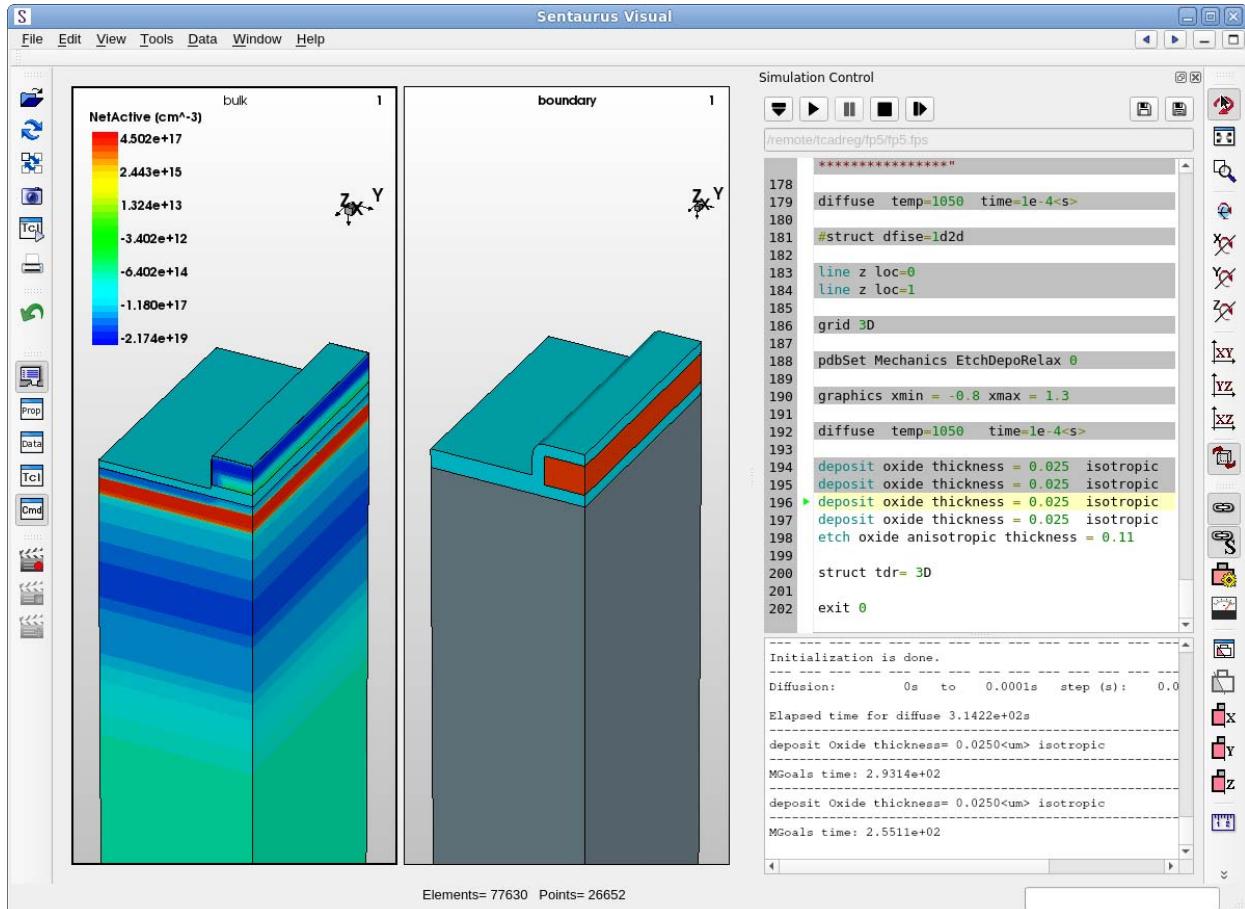
Interface to Sentaurus Process and Sentaurus Interconnect

The interface to Sentaurus Process and Sentaurus Interconnect visualizes 1D, 2D, and 3D structures and data evolution while the simulation progresses. The interface is initiated and controlled from Sentaurus Visual, including control of the simulation.

Chapter 1: Introduction to Sentaurus Visual

Interface to Sentaurus Process and Sentaurus Interconnect

Figure 3 Interface to Sentaurus Process: upper part of Simulation Control panel shows command file and lower part of panel shows log file



Chapter 1: Introduction to Sentaurus Visual

Interface to Sentaurus Process and Sentaurus Interconnect

Figure 4 Interface to Sentaurus Interconnect: upper part of Simulation Control panel shows command file and lower part of panel shows log file

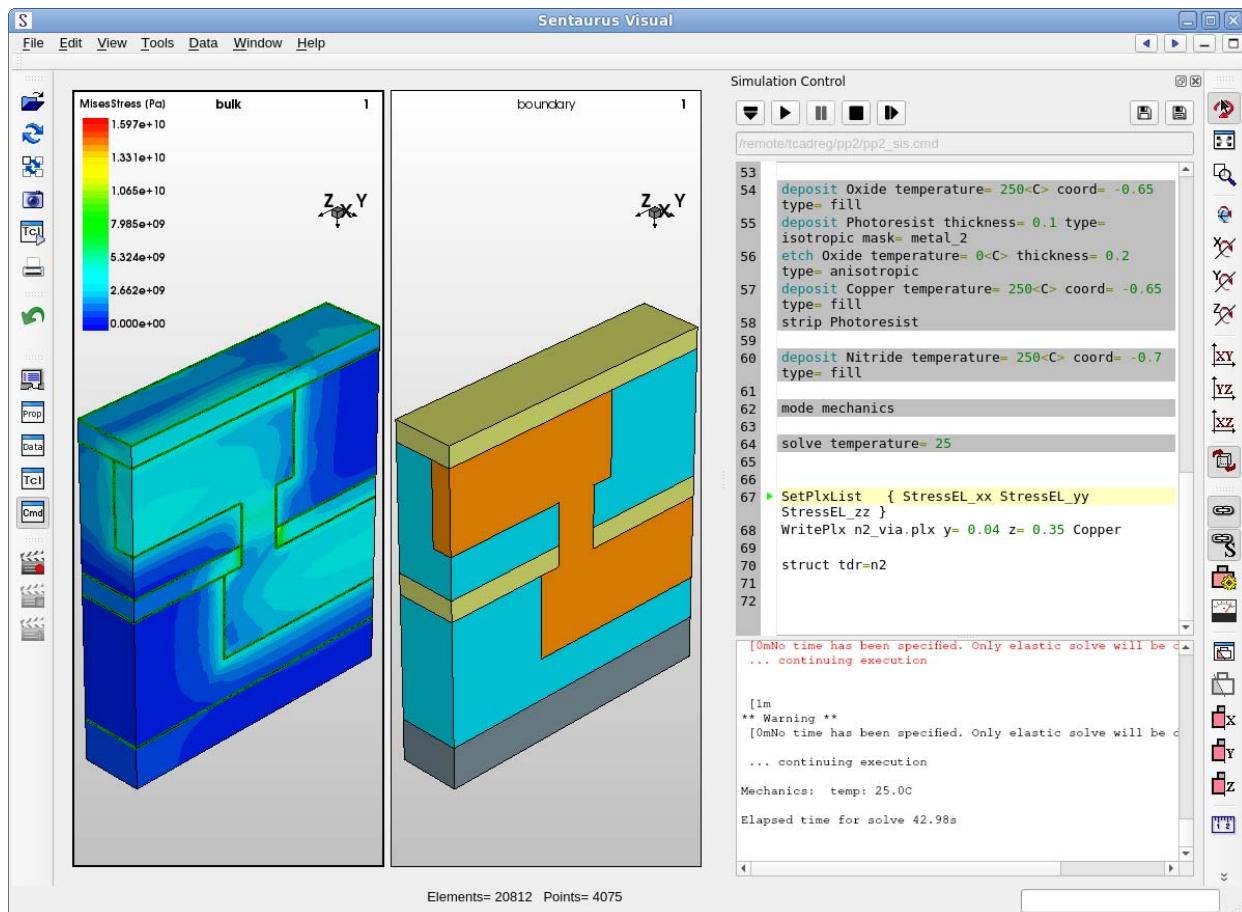


Table 3 Buttons of Simulation Control panel

Button	Description
▼	The Load button loads a command file. When a command file is loaded, it is thereafter referred to as the <i>flow</i> .
▶	The Run button runs the flow. This button either starts running the flow or continues execution after pausing the flow. The simulation continues at the location of the cursor (line with a light-yellow background).

Table 3 Buttons of Simulation Control panel (Continued)

Button	Description
	The Pause button pauses the running flow. The pause occurs either when the currently executing step (command) is finished or, for a long-running step, when the current time step is completed.
	The Reset button resets the running flow. The running flow stops, the connection to the simulator is terminated, and you return to the start of the flow.
	The Run Step button runs the next step in the flow. When you click this button, either a single step (command) is executed or a group of commands enclosed in braces is executed. You must repeatedly click this button to execute the next steps.
	The Save button saves the flow.
	The Save As button saves the flow under a new name.
	The Undo button restores the previously saved simulation state. This button is available only when Breakpoint Behavior is set as Breakpoint as Checkpoint or Breakpoint and Checkpoint in user preferences (see Setting Up the Interface).

Setting Up the Interface

To set up and run the interface:

1. Launch Sentaurus Visual with the following command-line options:

```
> svisual -spi &
```

2. From the Sentaurus Visual GUI, choose **Edit > Preferences**.

The User Preferences dialog box opens.

3. Expand **Common > Miscellaneous**.

4. Under Simulator, in the **Command** field, enter the tool binary as well as any command-line options required. For example:

```
sprocess -n
```

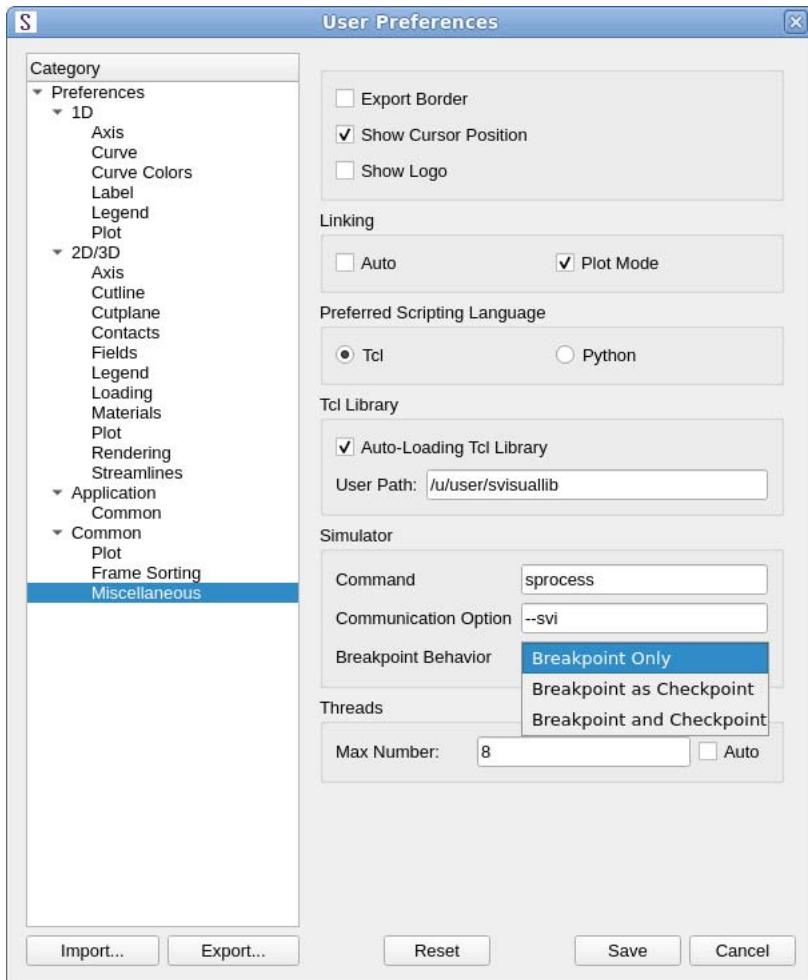
5. Under Simulator, from the **Breakpoint Behavior** list, select one of the following:

- **Breakpoint Only**: Pauses the flow execution at the required step.

Chapter 1: Introduction to Sentaurus Visual

Interface to Sentaurus Process and Sentaurus Interconnect

- **Breakpoint as Checkpoint:** This is the same as **Breakpoint Only**, but as well as pausing the flow, it also saves the step result. This allows the **Undo**  button to reach the last stored checkpoint.
- **Breakpoint and Checkpoint:** With this option, the **Undo** button functionality is separated from breakpoints. A first click creates a breakpoint without saving the step result, behaving like the **Breakpoint Only** option. A subsequent click transforms this step into a checkpoint, saving the step result but not stopping the flow execution.



6. Click **Save**.

Loading Command Files

To load a command file:

1. In the Simulation Control panel, click the **Load**  button.
The Load File dialog box opens.
2. Select a command file.
3. Click **Open**.

In addition, a command file can be loaded from the command line using:

```
svisual -spi <filename>
```

Inserting and Deleting Breakpoints in the Flow

To set breakpoints to pause the simulation at a particular step in the flow:

1. Click in the left margin of the line corresponding to the step (command) where you want the flow to stop.
A red circle indicates the location of the breakpoint (see [Figure 5 on page 38](#)).
2. Click the **Run** button to execute the entire flow, or click the **Run Step** button to execute individual steps.

Note:

You can set multiple breakpoints in a flow.

Step results are saved if **Breakpoint as Checkpoint** is selected as the breakpoint behavior in user preferences (see [Setting Up the Interface on page 35](#)).

To delete a breakpoint:

- Click the red circle in the left margin until the circle disappears.

Chapter 1: Introduction to Sentaurus Visual

Interface to Sentaurus Process and Sentaurus Interconnect

Figure 5 Simulation Control panel showing (left) Sentaurus Process command file and (right) Sentaurus Interconnect command file with breakpoints and checkpoints

```

Simulation Control
remote/tcadreg/pp38/pp38-2_fps.cmd
126 mgoals on min.normal.size= 0.001$/f normal.growth.ratio= 2.0
128 refinebox interface.materials= $Sub
129
130 icwb.create.mask name= Active layer.name= "Active" polarity= positive
131
132 refinebox name= WellVt mask= Active \
133     min= "0.0 $Ymin" max= "0.4 $Ymax" \
134     extend= 0.004 extrusion.min= 0.0 extrusion.max= 0.4 \
135     xrefine= "$.05/$f" yrefine= "$PolyPitch/(8.0*$f)" all add
136
137 grid remesh
138
139 ● deposit material= {Oxide} type= isotropic rate= {1.0} time= 0.005
140
141 icwb.create.mask layer.name= "pWELL" name= PWELL info= 1 polarity= negative
142 photo.mask= PWELL thickness= 1.5
143
144 > implant Boron dose= 2e+13 energy= 300.0 tilt= 0.0
145 implant Boron dose= 2e+13 energy= 200.0 tilt= 0.0
146 implant Boron dose= 1e+13 energy= 70.0 tilt= 0.0
147 implant Boron dose= 1e+13 energy= 25.0 tilt= 0.0
148 strip Photoresist
149
150 icwb.create.mask layer.name= "pWELL" name= NWELL info= 1 polarity= positive
151 photo.mask= NWELL thickness= 1.5
152
153 ● implant Phosphorus dose= 1.0e13 energy= 140 tilt= 0.0
154 implant Phosphorus dose= 7.5e12 energy= 100 tilt= 0.0
155 implant Phosphorus dose= 5.0e12 energy= 60 tilt= 0.0
156
157 implant Arsenic dose= 1.0e13 energy= 90 tilt= 0.0
158 implant Arsenic dose= 2.5e12 energy= 40 tilt= 0.0
159 implant Arsenic dose= 4.0e+12 energy= 30 tilt= 0.0
160 strip Photoresist
161
162 SetPtxList { Boron_Implant Arsenic_Implant Phosphorus_Implant }
163 WritePtx n38.well_imp ptx y= $Ymid
164 temp_ramp name= well time= 1.8<s> temp= 600.0 ramprate= 250.0000
165 temp_ramp name= well time= 20.0<s> temp= 1050.0 ramprate= 0.0000
166

```



```

Simulation Control
remote/tcadreg/dual_damasc_3d/dual_damasc_3d.sis
1 > # Testcase: SI Current Analysis, fixed potential biases
2
3 # Define regional contacts
4 contact region=contact1 name=ct1
5 contact region=contact2 name=ct2
6
7 # Bias the contacts
8 supply contact.name=ct1 voltage=1.0e-4<V>
9 supply contact.name=ct2 voltage=0<V>
10
11 # Grid settings
12 pdbSet Grid No3DMerge 1 ;# Test continuous BC's between same material interfaces
13 refinebox clear
14 refinebox interface.materials = {Copper Metal Oxide} min.normal.size = 0.02
normal.growth.ratio = 1.2
15
16 ● init tdr=dual_damasc_3d
17
18 mode current
19
20 # Start simulation
21 solve info=2
22
23 #struct tdr=res_new !Gas
24 struct tdr=res !Gas
25
26 set max_E_field [select name=ElectricField max]
27 set max_curr_dens [select name=ConductionCurrentDensity max]
28 Logfile "MAX(E-field) = $max_E_field V/cm"
29 Logfile "MAX(curr_dens) = $max_curr_dens A/cm^2"
30 set c1 [contact Potential name= "ct1" current]
31 set c2 [contact Potential name= "ct2" current]
32LogFile "$c1!$c2"
33

```

Inserting and Deleting Checkpoints in the Flow

To set checkpoints to save the simulation state at a particular step in the flow (only if **Breakpoint Behavior** is set to **Breakpoint and Checkpoint**):

1. Click in the left margin once, at the line corresponding to the step (command) where you want the flow to save the step result.
- A red circle indicates the location of the breakpoint (see Figure 5).
2. Click again over the same margin.
- A blue circle indicates the location of the checkpoint (see Figure 5).
3. Click the **Run** button to execute the entire flow, or click the **Run Step** button to execute individual steps.

Note:

You can set multiple checkpoints in a flow.

To delete a checkpoint:

- Click the blue circle in the left margin until the circle disappears.

Indicating Status of Steps

In the Simulation Control panel, as the flow is executed, a green triangle in the left margin (at the same location as breakpoints) indicates the step to be executed next. A red triangle indicates the step that is currently being executed.

Already executed steps are indicated by a gray background. This part of the flow cannot be changed further.

In addition, wherever the cursor is placed in the flow, its position is indicated by highlighting the line with a light-yellow background.

Updating the Structure

In 3D simulations, two plots are shown with the titles *bulk* and *boundary* (see [Figure 3](#) and [Figure 4](#)). This is because, in 3D simulations, both the bulk and boundary are not always up to date. The plot with its title in bold indicates the last updated information.

For example, if the `insert` command is called in a 3D simulation, then the boundary is updated; however, by default, the mesh is not updated. After the insertion operation is completed, the title of the *boundary* plot is bold, and the title of the *bulk* plot is not bold.

Multithreading Support

Sentaurus Visual provides multithreading capabilities for the following features to improve their performance:

- Loading new TDR format files
- Visualizing Sentaurus Topography 3D structures and Sentaurus Process Explorer structures
- Field integration (see [Integration Tool on page 155](#))
- Value blanking (see [Value Blanking on page 166](#))
- Cutting structures (see [Cutting Structures on page 172](#))
- Projection (see [Two-Dimensional Projections on page 186](#))
- Streamlines (see [Streamlines on page 202](#))

In addition, Sentaurus Visual provides multithreading support for some internal computations, improving the general performance of the application. Most of the features work on a region basis, which translates to better speedups when regions are similar in size or complexity.

Chapter 1: Introduction to Sentaurus Visual

Multithreading Support

You can configure the number of threads used by Sentaurus Visual in user preferences: expand **Common > Miscellaneous** and, under Threads, specify the maximum number of threads (see [Figure 137 on page 206](#)). You can select **Auto** next to the **Max Number** field to let Sentaurus Visual compute the ideal number of threads to use or set a preferred value. This number is usually the number of available CPUs on the machine. You also can select a fixed number of threads to use.

For convenience, you can specify the command-line options `--threads` and `--max_threads` to control the number of threads used. Both options accept an integer value and can be specified when launching Sentaurus Visual:

- `--threads <integer>`: The specified value overrides any setting specified in user preferences. This option is convenient when you want to use a different number of threads occasionally without modifying user preferences. Specifying 0 yields to automatic detection, which usually allocates one thread per available CPU on the machine.
- `--max_threads <integer>`: The specified value enforces an overall limit on either user preferences or `--threads` usage. In any case, the number of threads to run concurrently is constrained by the specified value. If you specify 0, then it does not limit user preferences or the `--threads` option.

Note:

Sentaurus Visual might use more threads than specified for other internal functionalities. For example, the Mesa driver uses threads for each 2D and 3D plot. Nevertheless, the number of concurrent threads in execution is constrained by user preferences or the `--threads` or `--max_threads` option.

2

Basic Operations

This chapter describes the basic operations that are common to all types of plot in Sentaurus Visual.

Loading Files

You can load files from the user interface or the command line. For example:

```
svisual [file1.tdr file2.tdr ...]
```

To load a file from the user interface:

1. Choose **File > Open**.
2. In the dialog box that opens, browse to the file you want to open, or enter the file name in the **File name** field.
3. Click **Open**.

An opened file consists of datasets. A dataset is a structure containing data that is plotted on xy, 2D, or 3D space. For example, a **.plt**, or **.plx** file can consist of one or more datasets, and **.tdr** files usually consist of only one dataset.

Note:

To select multiple files, hold the Ctrl key when you click the required files to load.

Supported File Formats

Sentaurus Visual supports the most commonly used file formats, including: **.csv**, **.plt**, **.plx**, **.tdr**, and **.tif**.

Chapter 2: Basic Operations

Loading Files

Loading Scripts

Sentaurus Visual can load scripts from the command line. For example, you can simply enter svisual with the path of a Tcl (.tcl) or Python (.py) script, and Sentaurus Visual detects the script mode automatically and starts with that scripting language.

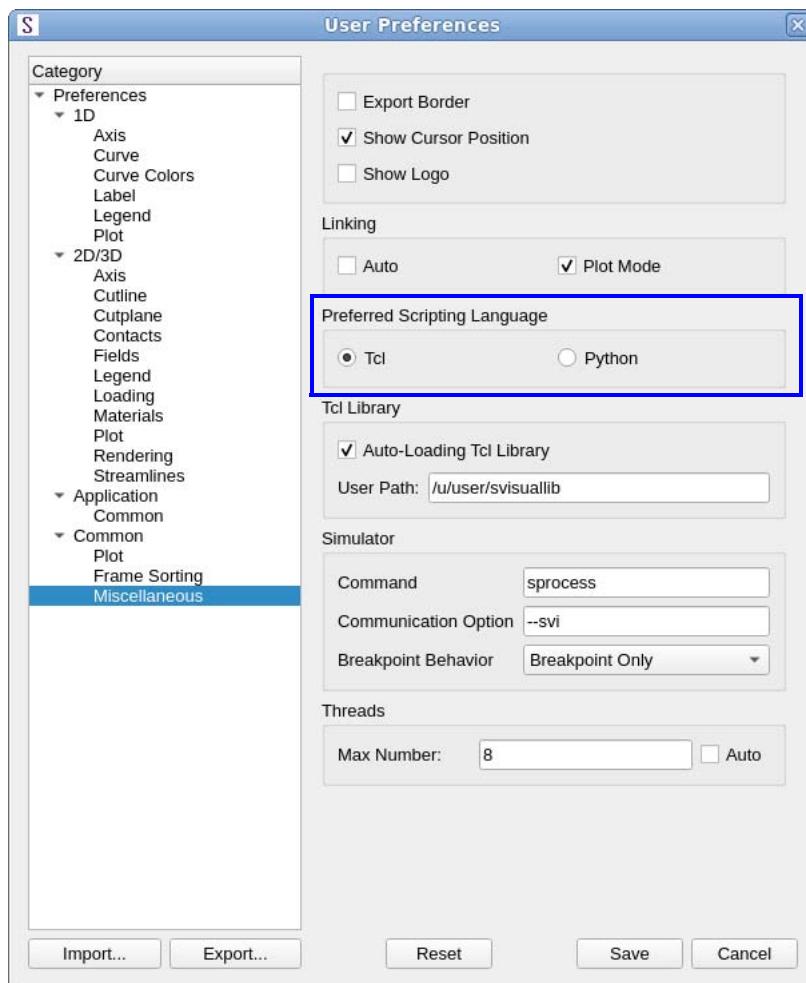
You can set a preferred scripting language in user preferences if you do not start Sentaurus Visual with the path of a script:

1. Choose **Edit > Preferences**.

The User Preferences dialog box opens.

2. Expand **Common > Miscellaneous**.

3. Under Preferred Scripting Language, select the required language.



4. Click **Save**.

Chapter 2: Basic Operations

Reloading Plot Files

Note:

If you start Sentaurus Visual with the `-t[c1]` or `-p[ython]` command-line option, then this enforces the selection, ignoring the preferred scripting language selected in user preferences.

Tcl and Python scripts run native Sentaurus Visual commands, while Inspect scripts need the `-inspect` or `-f` option to run Inspect commands in compatibility mode.

Most Inspect commands are fully supported, although some commands have only partial support and some commands are not supported at all. For detailed information about support for Inspect libraries and commands, see [Appendix E on page 413](#). For detailed information about Inspect commands, see the *Inspect User Guide*.

From the user interface, choose **File > Run Script**. A dialog box is displayed where you can select the script to load. The scripts loaded using the user interface run native commands only.

Reloading Plot Files

Sometimes, there are changes to the datasets from outside Sentaurus Visual. These changes can be shown without closing Sentaurus Visual.

To reload a specific dataset:

- ▶ Choose **File > Reload Selected** or press Shift+F5.

To reload all datasets:

- ▶ Choose **File > Reload All** or press the F5 key.

Note:

Not all changes to a dataset can be reloaded. For example, if the original structure was two dimensional, the reloaded data is expected to belong to a 2D plot. If, after changes, the dataset now contains data for a 3D structure, Sentaurus Visual cannot reload this plot.

Automatically Reloading Datasets

You can reload 1D datasets automatically periodically after a certain number of seconds.

To reload datasets automatically using user preferences (applies to all plots):

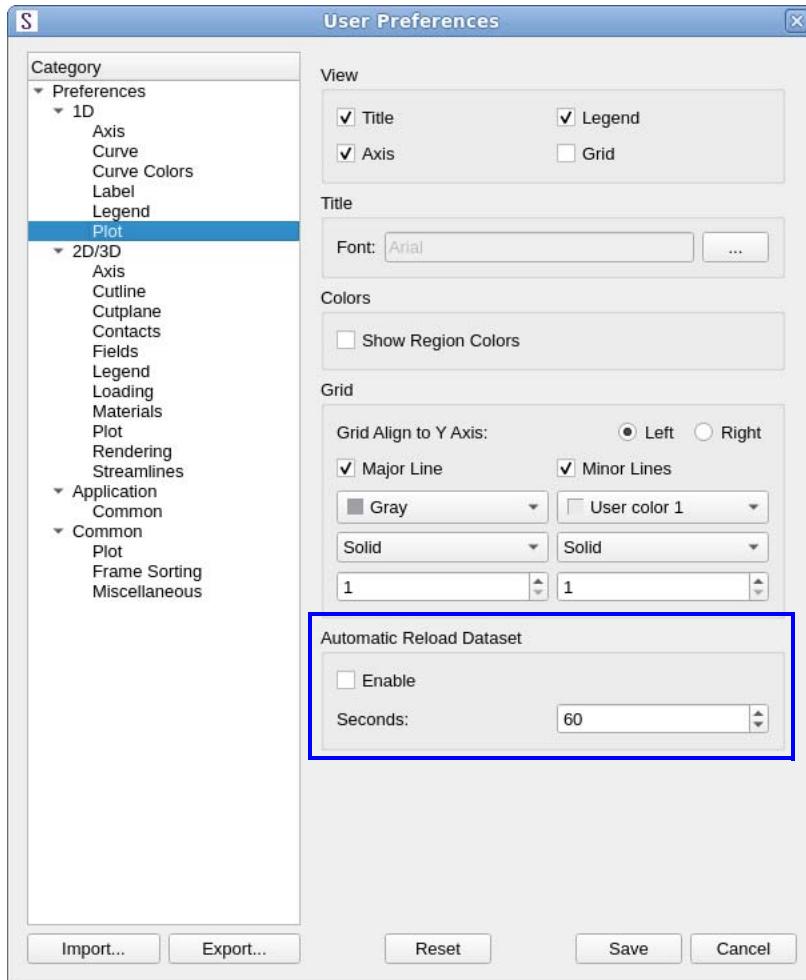
1. Choose **Edit> Preferences**.

The User Preferences dialog box opens.

Chapter 2: Basic Operations

Reloading Plot Files

2. Expand **1D > Plot**.
3. Under Automatic Reload Dataset, select **Enable**.



4. Specify the number of seconds.
5. Click **Save**.

Chapter 2: Basic Operations

Managing Loaded Information

To reload datasets automatically from the **File** menu:

1. Choose **File > Automatic Reload Dataset**.

The Automatic Reload Dataset dialog box opens.

2. Select **Enable**.



3. Specify the number of seconds between each reload.
4. Select whether the reload applies to the current selected plots or all plots.
5. Click **OK**.

Managing Loaded Information

Sentaurus Visual provides a dialog box to manage the information loaded in the current session.

Choose **Data > View Info Loaded** to display the Manage Loaded Data dialog box (see [Figure 6](#)) with all the data that is currently active and the option of removing plots and datasets.

To delete all xy plots:

1. In the Dimension pane, click **XY**.
2. Under Datasets, click **Remove**.

To delete a plot:

1. In the Plots pane, click the plot to be deleted.
2. Under Plots, click **Remove**.

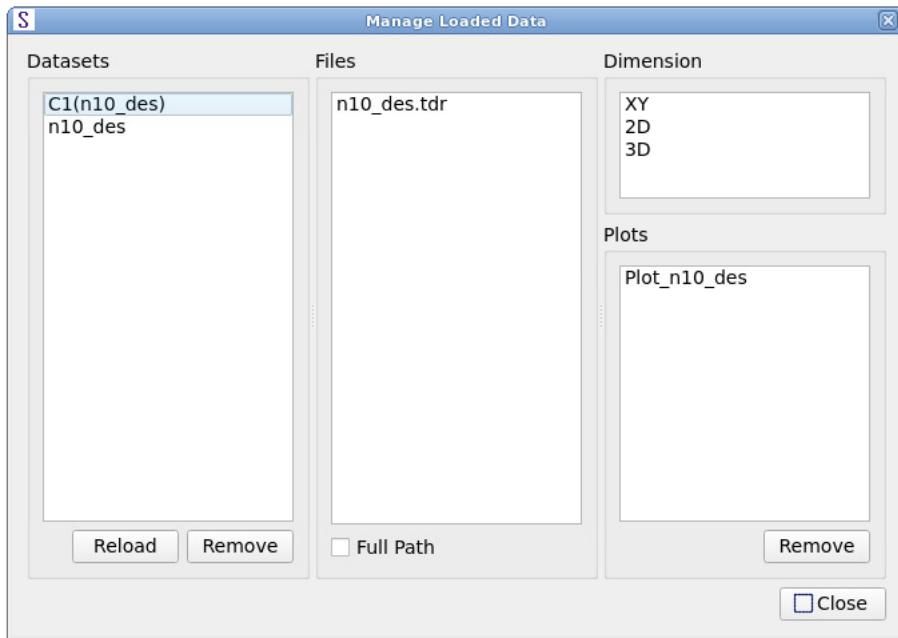
Chapter 2: Basic Operations

Customizing Settings

Note:

Deleting a plot does not delete the datasets associated with it. However, deleting a dataset removes the associated 2D or 3D plots. For xy datasets, only the curves that use the datasets are deleted.

Figure 6 Manage Loaded Data dialog box showing active data



Customizing Settings

You can customize the settings of Sentaurus Visual using the User Preferences dialog box (see [Figure 7](#)).

To display the User Preferences dialog box, choose **Edit > Preferences**.

You can also import or export settings to a file by clicking **Import** or **Export**. To restore the preferences to their defaults, click **Reset**.

Alternatively, you can import and export settings using Tcl commands:

- To import previously saved settings, use the `import_settings` command (see [import_settings on page 295](#)).
- To export the current settings, use the `export_settings` command (see [export_settings on page 249](#)).

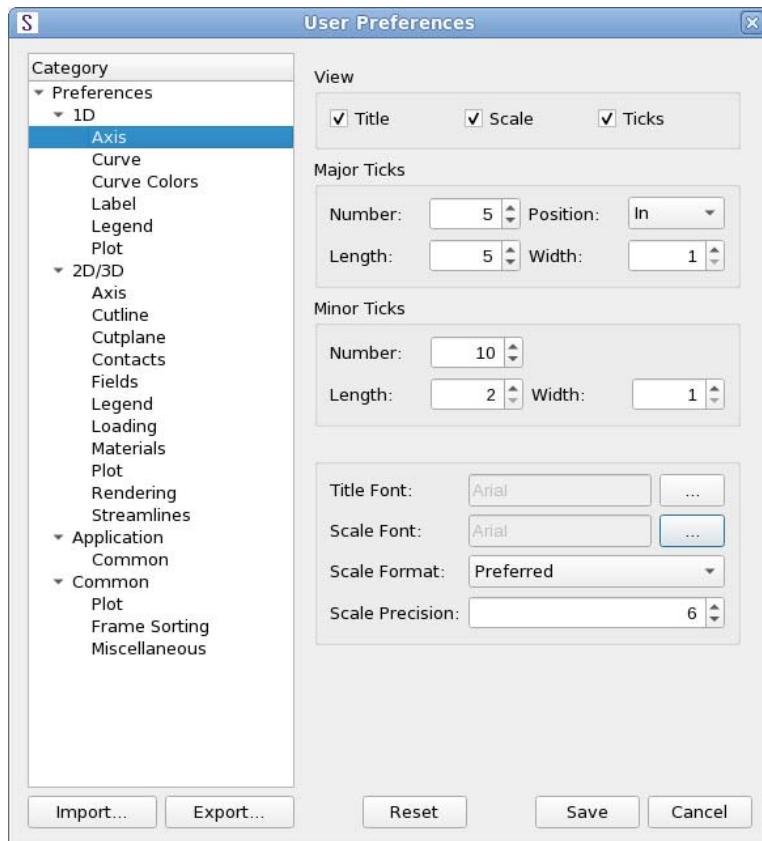
Chapter 2: Basic Operations

Creating Custom Buttons to Access Scripts

Note:

Settings are applied the next time you launch Sentaurus Visual.

Figure 7 User Preferences dialog box showing selected settings



An additional option that can be modified in the configuration file converts the old coordinate system to the unified coordinate system (UCS). The Boolean parameter is called `convert/oldCoordinateSystemToUCS` and belongs to the `PlotHD` group.

On Linux operating systems, user preferences are stored in the following file:

`~/.config/Synopsys/SVisual.conf`

Creating Custom Buttons to Access Scripts

You can create buttons to make it easier to execute or load Tcl script files. Custom buttons are added to the Custom Buttons toolbar, which is located immediately below the menu bar and, by default, has the + and - buttons.

Chapter 2: Basic Operations

Working With Plots

Each new button can be set up to load a Tcl script file or to execute directly a Tcl script code. For each button, you can assign text or an icon to display as the button, as well as a tooltip.

Custom buttons can be loaded at the start of a Sentaurus Visual session when it loads scripts stored in the Tcl script library (see [Script Library on page 211](#)).

When you click the button, Sentaurus Visual displays a message before and after the script is executed, so you can identify the section of the commands that is executed using the button. The message identifies the button that has been clicked by its name and description.

For more information, see [add_custom_button on page 217](#), [get_input_data on page 273](#), [list_custom_buttons on page 301](#), and [remove_custom_buttons on page 327](#).

Working With Plots

Sentaurus Visual offers different modes when interacting with plots. These modes are independent for each plot instance and are enabled using toolbar buttons. However, you can apply mode changes to a group of plots by selecting the plots *before* applying the mode change.

Modes When Interacting With Plots

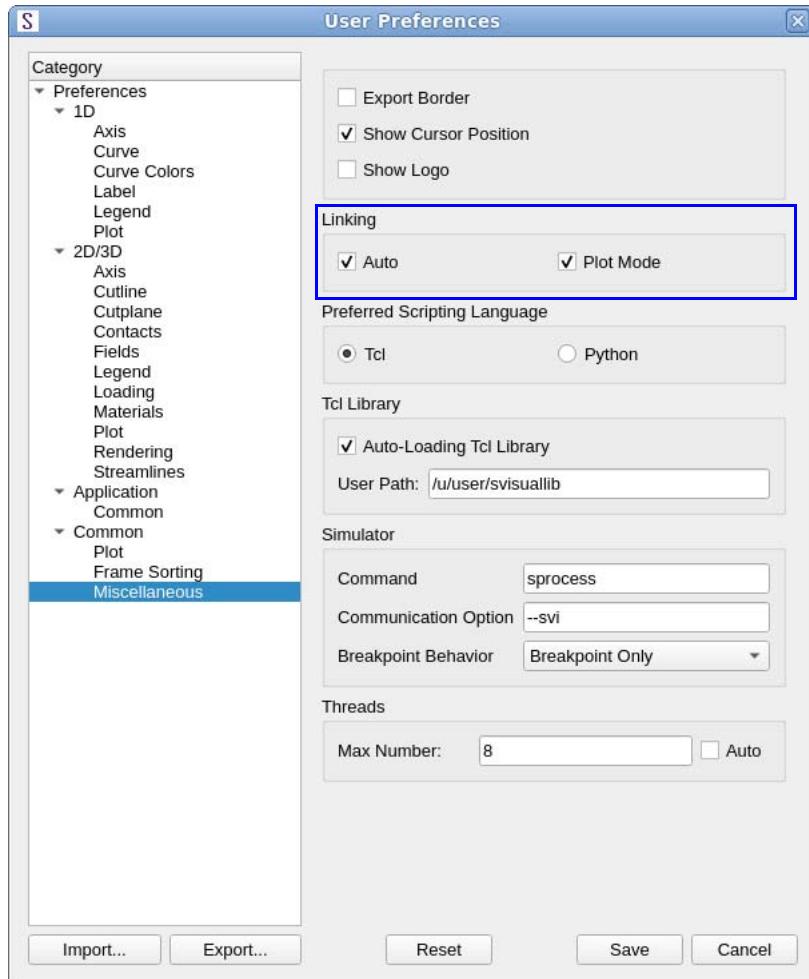
By default, a group of linked plots shares the same mode. This behavior can be switched off in the User Preferences dialog box (expand **Common > Miscellaneous**) by deselecting **Plot Mode** (see [8]). If this option is not selected, you can use special linking for a specific group of linked plots to change this behavior (see [Linking Plots on page 51](#)).

The modes temporarily modify the behavior of the left mouse button, allowing you to perform specific operations.

Chapter 2: Basic Operations

Working With Plots

Figure 8 User Preferences dialog box showing Plot Mode option selected (the default)



Common Modes

All modes are enabled by clicking a toolbar button. The current mode remains active until you select another mode:

- The Selection mode (the default mode) allows you to select and move all objects inside plots (such as curves, legend, rectangles, and ellipses).
- The Zoom mode allows you to drag the left mouse button to draw a box. When you release the mouse button, the area delimited by the box will be magnified.
- The Probe mode allows you to extract data by clicking in the plot. For xy plots, curve data is extracted (see [Probing on page 94](#)). For 2D and 3D plots, structure data is extracted (see [Probing on page 157](#)).

Chapter 2: Basic Operations

Working With Plots

XY Plot–Only Modes

All modes are enabled by clicking a toolbar button. The Drawing mode  displays a submenu of drawing options:

- The Draw Line mode  allows you to draw a line with the left mouse button.
- The Draw Rectangle mode  allows you to draw a rectangle with the left mouse button.
- The Draw Ellipse mode  allows you to draw an ellipse with the left mouse button.
- The Insert Text mode  allows you to insert a text box with the left mouse button at a specified position.

Note:

For all of these drawing options, when you release the mouse button, the current mode finishes and it changes to the Selection mode.

2D Plot–Only Modes

All modes are enabled by clicking a toolbar button. The modes specific to 2D plots only are:

- The Cut X mode , the Cut Y mode , and the Cut Z mode  allow you to generate an axis-aligned (x, y, or z) cutline at a specified position (see [Cutlines in 2D Plots on page 176](#)). When you release the mouse button, the current mode finishes and it changes to the Selection mode.
- The Cutline mode  allows you to draw a cutline with the left mouse button. When you release the mouse button, the current mode finishes and it changes to the Selection mode.
- The Ruler mode  allows you to draw a line with the left mouse button to perform a measurement. If you hold the Ctrl key while you click the mouse button, the snap-to-mesh mode is enabled (see [Measuring Distances on page 153](#)). This mode remains active until you select another mode.
- The Drawing mode  displays a submenu of drawing options:
- The Draw Line mode  allows you to draw a line with the left mouse button.
- The Draw Rectangle mode  allows you to draw a rectangle with the left mouse button.
- The Insert Text mode  allows you to insert a text box with an arrow that can be repositioned using the left mouse button.

Note:

For all of these drawing options, when you release the mouse button, the current mode finishes and it changes to the Selection mode.

Chapter 2: Basic Operations

Working With Plots

3D Plot–Only Modes

All modes are enabled by clicking a toolbar button. The modes specific to 3D plots only are:

- The Spherical Rotation mode  allows you to perform a rotation in spherical coordinates using the left mouse button. This mode overrides the Selection mode as the default mode and remains active until you select another mode.
- The Rotation Axis X mode , the Rotation Axis Y mode , and the Rotation Axis Z mode  allow you to perform a rotation around a fixed x-axis, y-axis, or z-axis with the left mouse button. This mode cannot select plot elements (aside from the legend) and remains active until you select another mode.
- The Cut X mode , the Cut Y mode , and the Cut Z mode  allow you to generate an axis-aligned (x, y, or z) cutplane at a specified position (see [Cutplanes in 3D Plots on page 190](#)). When you release the mouse button, the current mode finishes and it changes to the Selection mode or the Spherical Rotation mode (depending on which mode was last active).
- The Ruler mode  allows you to draw a line with the left mouse button to perform a measurement. If you hold the Ctrl key while you click the mouse button, the snap-to-mesh mode is enabled (see [Measuring Distances on page 153](#)). This mode remains active until you select another mode.

Linking Plots

The feature of linking plots can be used to compare similar models, as it allows you to manipulate elements from one plot of the group, and the linked elements will change on all plots of the group. Elements that can be linked include material/region selection, field selection and properties, movement and rotation, cutplanes and cutlines, axes properties (only in xy plots and 2D plots), legend properties, curves properties, grid properties, and plot properties.

To link plots:

1. Select the plots to be linked by holding the Shift key and clicking the required plots.
2. Click the  toolbar button.

The linking operation links all properties except for y-axes and y2-axes in xy plots and streamlines in 2D and 3D plots. For customized linking properties, special linking can be used to link only specified properties and to set the remaining properties individually.

Plot linking also links the plot mode. This behavior is switched on by default and can be changed in the User Preferences dialog box (expand **Common > Miscellaneous**) by deselecting **Plot Mode** (see [Figure 8 on page 49](#)). However, special linking can be used to change this behavior for particular groups of linked plots.

Chapter 2: Basic Operations

Working With Plots

Note:

All plot properties are linked by default, including the properties of the plot title, except the text of the title, which is independent of the other plots regardless of which linking option is selected.

To use special linking:

1. Select the plots to be linked by holding the Shift key and clicking the required plots.
2. Click the  toolbar button.

The properties that can be linked or unlinked with special linking include:

- Common properties:
 - Legend settings and movement
 - Plot properties and plot mode
- Only for xy plots:
 - Curve settings and grid settings
 - Axis settings (divided into x-, y-, and y2-settings)
- Only for 2D plots:
 - Axis properties
- Only for 2D and 3D plots:
 - Material or region selection
 - Field selection and field properties
 - Deformation and streamlines
 - Cuts

Automatically Linking Plots

Plot linking is switched off by default. You can activate this feature automatically in the User Preferences dialog box as follows:

1. Choose **Edit > Preferences**.
2. Expand **Common > Miscellaneous**.
3. Under Linking, select **Auto** (see [Figure 8 on page 49](#)).
4. Click **Save**.

Chapter 2: Basic Operations

Working With Plots

When linking is set to automatic, plots are linked only if the following criteria are met:

- Plots must be dimension compatible, that is, the plots must have the same dimensions. In addition, 2D plots can be linked to xy plots when generated by cutline operations (they will share the range of the common axis). Three-dimensional plots cannot be linked to xy plots.
- Plots must use the same coordinate system. For example, a 3D plot using UCS and a 3D plot using the DF–ISE coordinate system cannot be linked.
- Plots must use the same unit for axes. For example, if several 2D plots all use μm as the unit for their axes, then all these plots are linked. On the other hand, if one 2D plot uses μm and another 2D plot uses no units, then these two plots are not linked.

When linking is set to automatic, to distinguish plots, the following applies:

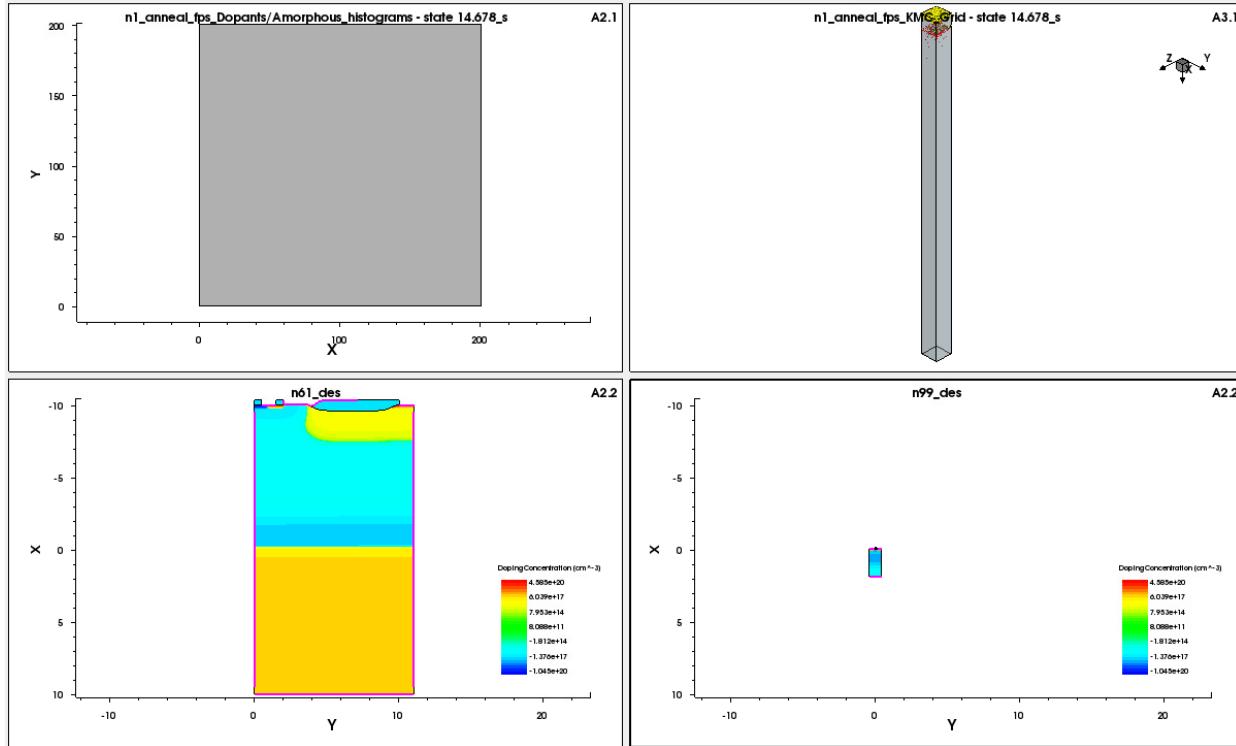
- The prefix A1 indicates an xy plot.
- The prefix A2 indicates a 2D plot.
- The prefix A3 indicates a 3D plot.
- Linked 2D and 3D plots have the same suffix, such as .1, .2, and .3.
- Linked xy plots have no suffix.

[Figure 9](#) shows two linked plots, indicated by A2.2 in their upper-right corners. Both are 2D plots and both use the same units for their axes (μm). The other 2D plot is not linked (A2.1) because it is unitless. It has the suffix .1 because it is the first 2D plot in the plot area.

Chapter 2: Basic Operations

Working With Plots

Figure 9 (Top row) Unitless 2D plot and 3D plot that are not linked to other plots, and (bottom row) two automatically linked plots with the same dimension and the same units used for their axes



Undoing Operations

Most user interaction commands in Sentaurus Visual have undo functionality that allows you to revert recent changes to the visualization.

To undo an operation, click the  toolbar button or use the `undo` Tcl command (see [undo on page 383](#)).

Displaying Multiple Plots

In the plot area, multiple plots can be displayed in a grid, with or without keeping the aspect ratio. In addition, you can arrange plots horizontally or vertically.

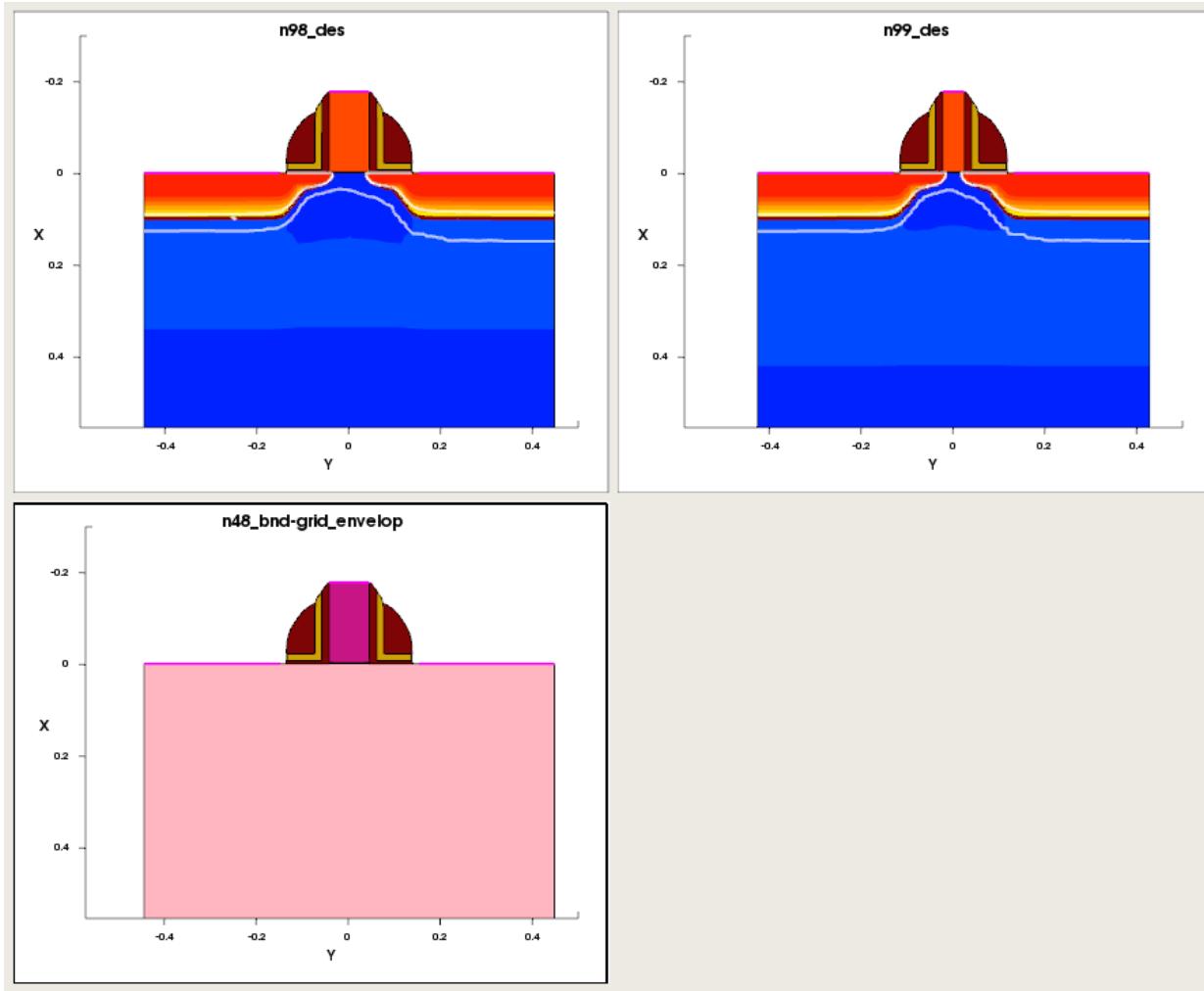
Grid Orientation

To display plots in a grid configuration (see [Figure 10](#)), choose **Window > Tile Grid**.

Chapter 2: Basic Operations

Working With Plots

Figure 10 Multiple plots keeping the same aspect ratio



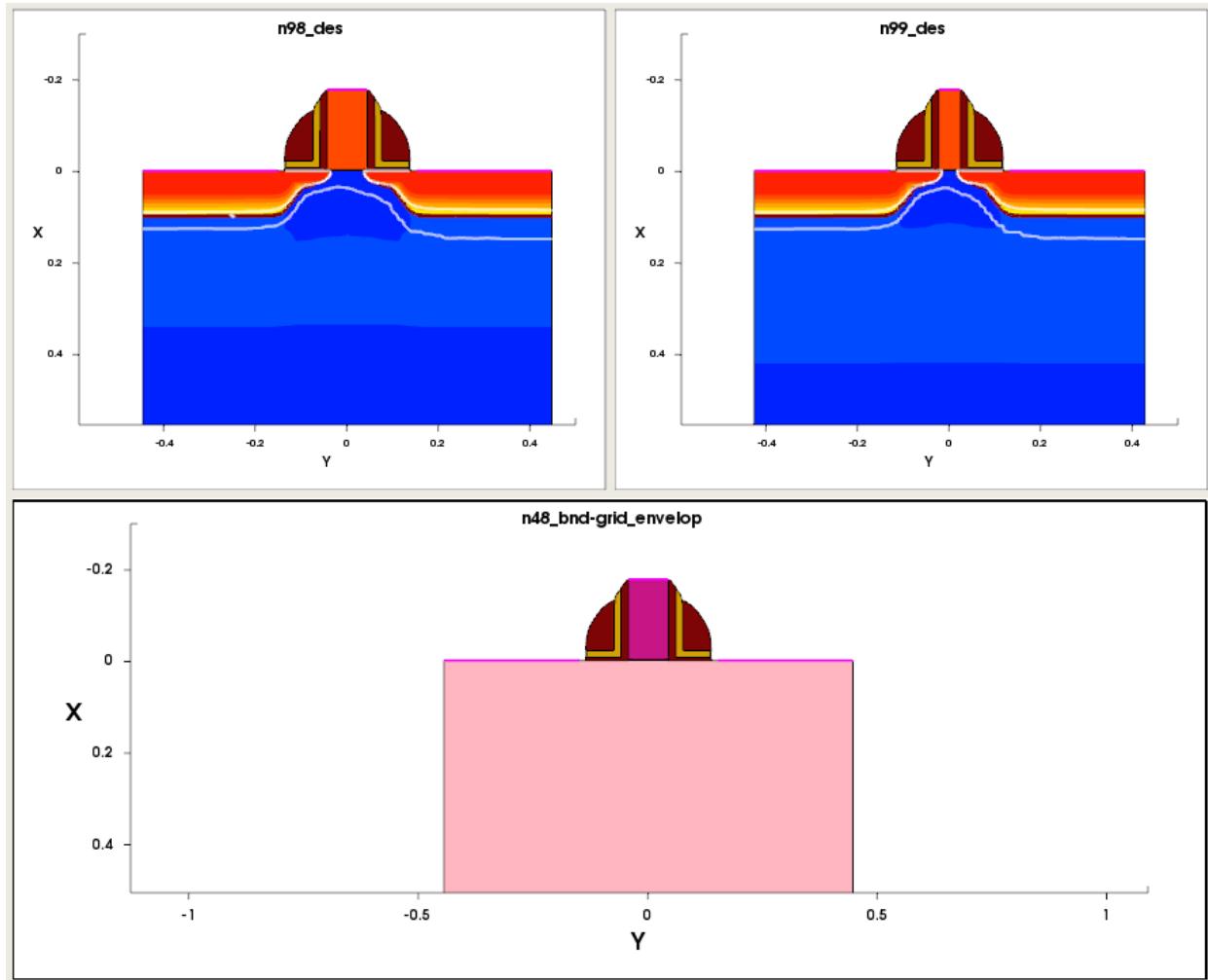
By default, the aspect ratio between plots is preserved, but this can be changed by deselecting **Keep Aspect Ratio** in the Manage Frames dialog box (see [Figure 14 on page 58](#)). [Figure 11](#) shows plots where the aspect ratio is not maintained.

When the aspect ratio is not maintained, the unused space is filled with the last plot frame, but the aspect ratio of the structure is preserved.

Chapter 2: Basic Operations

Working With Plots

Figure 11 *Multiple plots without keeping the same aspect ratio*



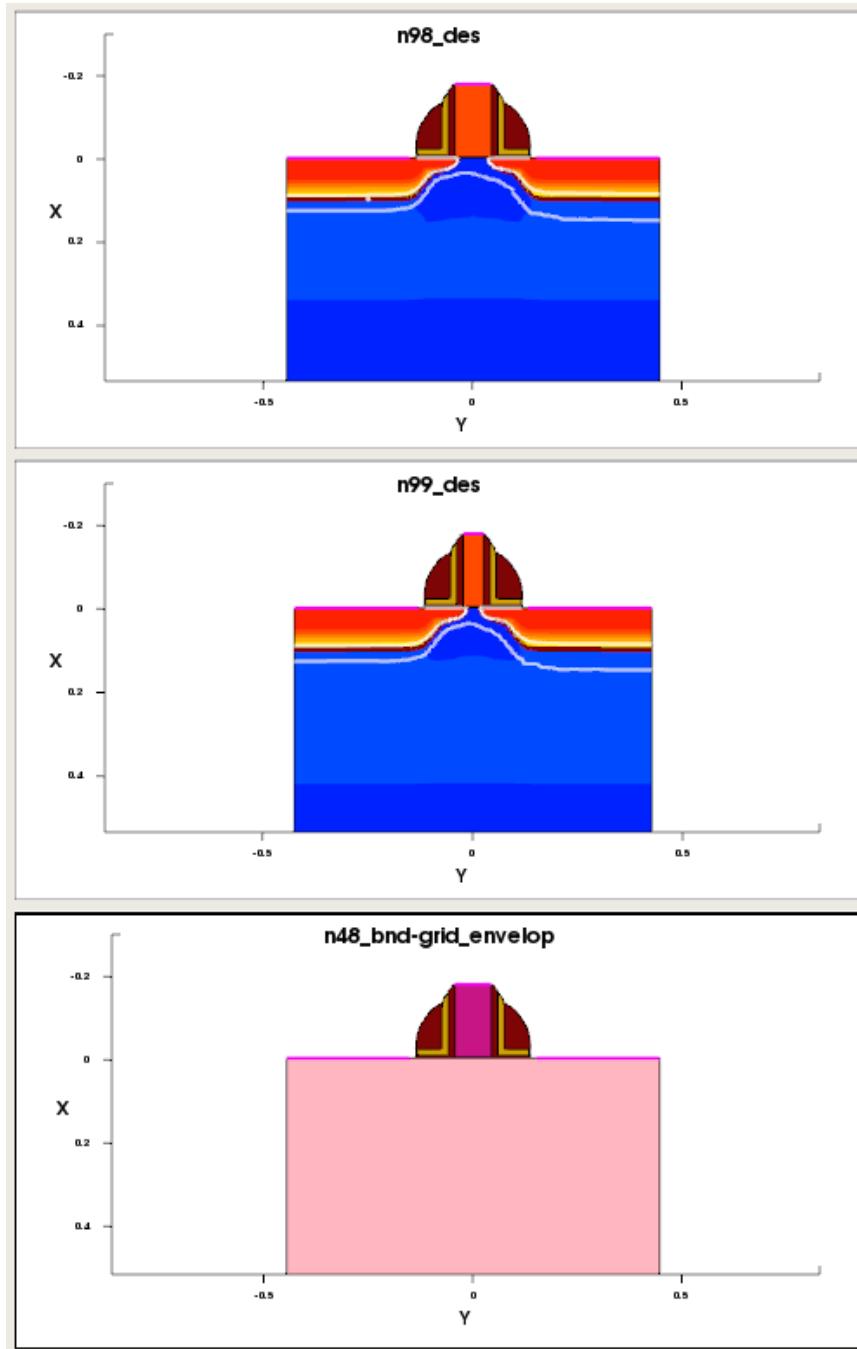
Vertical Orientation

To arrange plots vertically (see Figure 12), choose **Window > Tile Vertically**.

Chapter 2: Basic Operations

Working With Plots

Figure 12 Plots arranged vertically



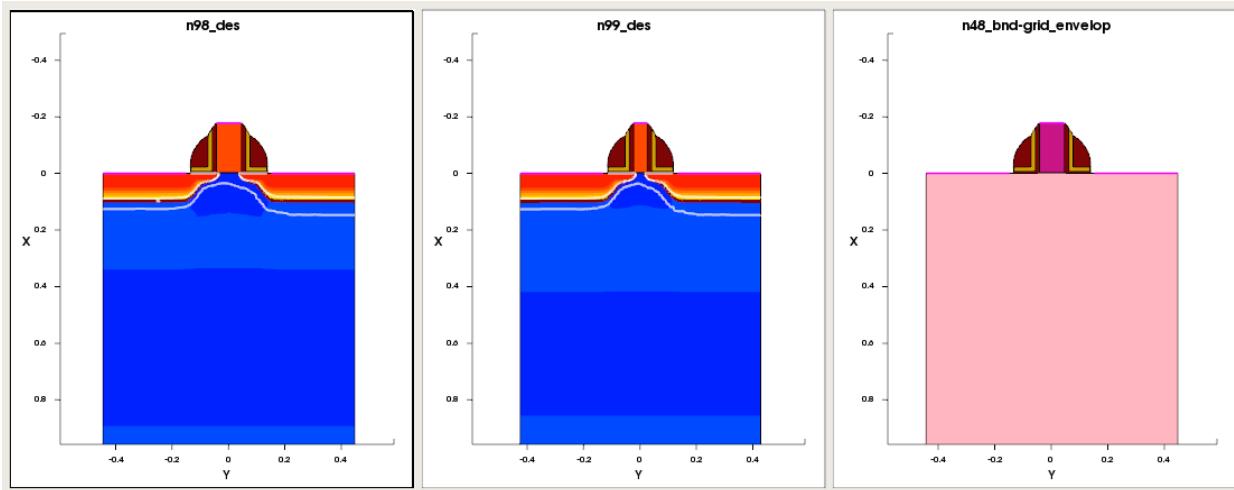
Horizontal Orientation

To arrange plots horizontally (see [Figure 13](#)), choose **Window > Tile Horizontally**.

Chapter 2: Basic Operations

Working With Plots

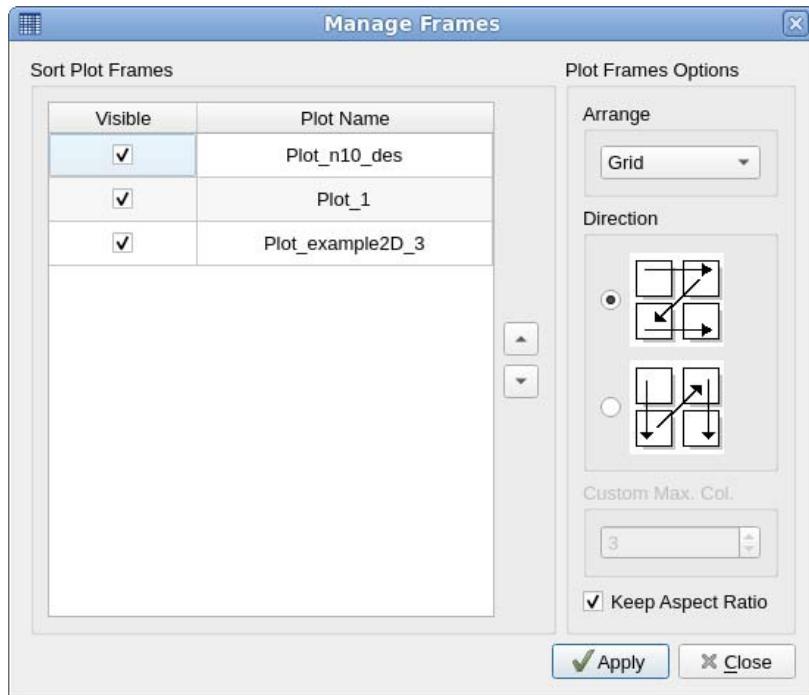
Figure 13 Plots arranged horizontally



Managing Frames

More advanced sorting options can be configured in the Manage Frames dialog box (see Figure 14). To display the dialog box, choose **Window > Manage Frames**.

Figure 14 Manage Frames dialog box



Chapter 2: Basic Operations

Drawing Inside Plots

Features available include setting a custom grid, sorting plots in the plot area, and changing the direction in which new plots are placed on the grid. In addition, you can manage plots by minimizing them or restoring them using the **Visible** option.

Drawing Inside Plots

Sentaurus Visual allows you to draw inside plots and to insert labels to allow plot customization. This feature is not available for 3D plots.

Inserting Text

To insert text inside a plot, click the  toolbar button, or use the `draw_textbox` command (see [draw_textbox on page 245](#)).

The text properties such as font size, font color, and position can be changed using the `set_textbox_prop` command (see [set_textbox_prop on page 373](#)). This feature is available for xy and 2D plots.

There is a difference between the behavior of text boxes in xy plots and 2D plots, which is related to the coordinate system:

- In xy plots, the lower-left corner of the text box is placed at a specified point using the world coordinate system {x, y}. The text box keeps its position even if you perform a panning operation.

The *world coordinate system* is a Cartesian coordinate system where the positions of objects, such as curves and drawn objects, are defined. The scale of the axes shows the world coordinate values.

- In 2D plots, the anchor arrow also exhibits this behavior, so it is placed at a specified point using the world coordinate system. However, the text box in 2D plots always retains its specified visual position even if you perform a panning operation. This is because the text box uses relative normalized screen coordinates, so its position is defined by numbers from 0.0 to 1.0, that is, {0.0, 0.0} for the lower-left corner of the plot area and {1.0, 1.0} for the upper-right corner of the plot area.

Drawing Rectangles

Note:

This feature is available only for xy and 2D plots.

To draw a rectangle, click the  toolbar button or use the `draw_rectangle` command (see [draw_rectangle on page 244](#)).

Chapter 2: Basic Operations

Drawing Inside Plots

You can edit a rectangle using the user interface or the `set_rectangle_prop` command (see [set_rectangle_prop on page 366](#)).

To delete a rectangle, select the rectangle and press the Delete key.

To delete multiple rectangles simultaneously, use the `remove_rectangles` command (see [remove_rectangles on page 331](#)).

To list the rectangles inside a plot, use the `list_rectangles` command (see [list_rectangles on page 311](#)).

Drawing Ellipses

Note:

This feature is available only for xy plots.

To draw an ellipse, click the  toolbar button or use the `draw_ellipse` Tcl command (see [draw_ellipse on page 242](#)).

You can edit an ellipse using the user interface or the `set_ellipse_prop` command (see [set_ellipse_prop on page 352](#)).

To delete an ellipse, select the ellipse and press the Delete key.

To delete multiple ellipses simultaneously, use the `remove_ellipses` command (see [remove_ellipses on page 329](#)).

To list the ellipses inside a plot, use the `list_ellipses` command (see [list_ellipses on page 305](#)).

Drawing Lines

Note:

This feature is available only for xy and 2D plots.

To draw a line, click the  toolbar button or use the `draw_line` Tcl command (see [draw_line on page 243](#)).

You can edit a line using the user interface or the `set_line_prop` command (see [set_line_prop on page 359](#)).

To delete a line, select the line and press the Delete key.

To delete multiple lines simultaneously, use the `remove_lines` command (see [remove_lines on page 330](#)).

Chapter 2: Basic Operations

Exporting Plots

To list the lines inside a plot, use the `list_lines` command (see [list_lines on page 308](#)).

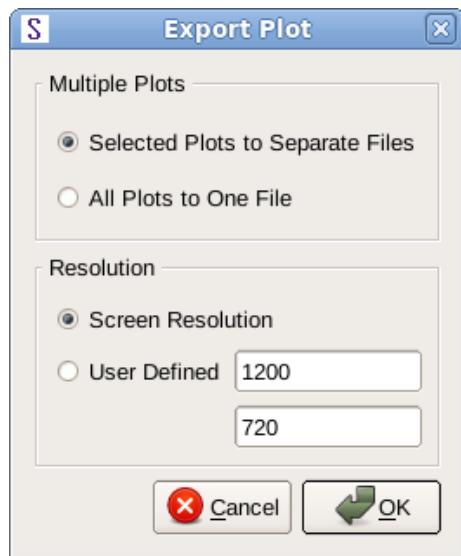
Exporting Plots

You can export plots to an image file. Sentaurus Visual supports exporting plots to the following file formats: BMP, EPS, JPG, JPEG, PNG, PPM, TIF, TIFF, XBM, and XPM.

To export plots:

1. Choose **File > Export Plot**, press **Ctrl+E**, or click the  toolbar button.

The Export Plot dialog box opens.



2. Select the option to export multiple plots.
3. Select the option for the resolution.
4. Click **OK**.

Note:

The **User Defined** option is not available if you select **All Plots to One File**.

If you specify a custom resolution rather than select **Screen Resolution**, then the exported plots might look different on-screen due to rescaling to the chosen resolution.

Exporting Movies

You can export several captures of one or more plots to generate an animated GIF file.

Starting a New Movie

To start a new movie:

1. Choose **Tools > Movies > Start Recording**, or click the  toolbar button.

The Start Recording dialog box opens, where you can generate frames for the movie.



2. Select the resolution:

- The **Screen Resolution** option keeps the size of the current view.
- The **User Defined** option allows you to specify the size of the capture in pixels.

3. Click **OK**.
-

Adding Frames in a Movie

To add a new frame in the movie:

1. Click the required plot to select it.

To select multiple plots, hold the Shift key while clicking the plots.

2. Choose **Tools > Movies > Add Frames**, or click the  toolbar button.

If several plots are selected, one frame for each plot is generated.

Chapter 2: Basic Operations

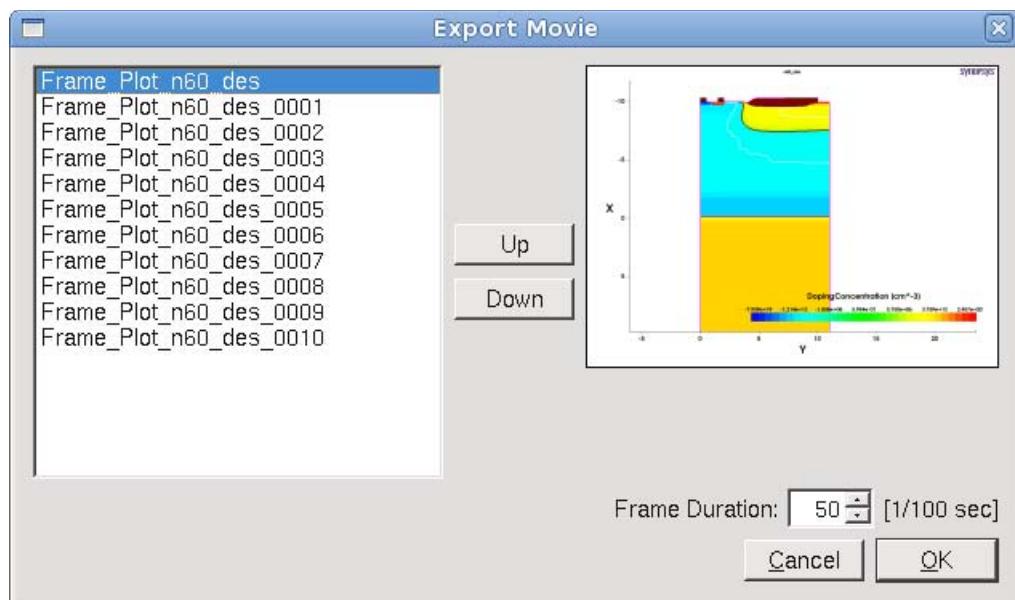
Exporting Movies

Exporting a Movie

To export a movie:

1. Choose **Tools > Movies > Stop Recording**, or click the  toolbar button.

The Export Movie dialog box opens.



2. To see a preview of a frame, click an item in the left pane.
3. Select the frames to export from the left pane.

To make multiple selections, drag to highlight the frames or hold the **Ctrl** key while clicking the frames. At least one frame must be selected.

4. If required, change the order of the frames by selecting a frame from the left pane, and clicking **Up** or **Down**.

Note:

The movie is recorded sequentially from the first frame to the last frame.

5. Set the duration of each frame in the **Frame Duration** field (the unit is 1/100 s).
 6. Click **OK** to save the file.
- Click **Cancel** to delete the entire frame buffer.
7. In the dialog box that opens, ensure that the file has the **.gif** extension. Add the extension if it is missing.

Chapter 2: Basic Operations

Printing Plots

Printing Plots

You can print selected plots by either clicking the  toolbar button, or choosing **File > Print Plots**, or pressing **Ctrl+P**.

The Printer dialog box is displayed, where you can select a printer and set print properties.

Note:

All plots are printed on one page.

Zooming and Panning

To take a closer look at significant details on a plot, there are various ways to zoom. On a selected plot, you can zoom by using the mouse wheel, or by clicking the middle mouse button and moving to the top or bottom of the screen. To pan, drag while holding the right mouse button.

Zoom Tool

The zoom tool is used to magnify a particular area of a plot.

To select the zoom tool:

1. Click the  toolbar button.
 2. Draw a rectangle by dragging the mouse over the area you want to magnify.
-

Reset Tool

The reset tool restores the selected plot position and zoom level. It does not restore the rotation on 3D plots.

To select the reset tool:

- Click the  toolbar button.
-

Deleting Plots

To delete selected plots:

- Choose **Data > Delete Selected Plots** or press **Ctrl+D**.

Chapter 2: Basic Operations

Performance Options

Note:

Deleting a plot does not delete the associated dataset. To delete the datasets or the plots or both, choose **Data > View Info Loaded** (see [Figure 6 on page 46](#)).

Performance Options

Working with complex 2D and 3D plots can be sometimes slow. To improve this, Sentaurus Visual provides two options to work faster with plots.

Fast Draw (3D Plots Only)

This option draws only the boundaries of a 3D plot when it is manipulated, which has a large impact on performance.

To activate fast draw:

- ▶ Choose **View > Fast Draw** or click the  toolbar button.

Subsampling (2D and 3D Plots Only)

This option reduces the data points when manipulating a plot, enabling better performance on slower computers.

To activate subsampling:

- ▶ Choose **View > Subsampling**.

Advanced Options (XY Plots Only)

You can customize the minimum value to be displayed in log scale for xy plots. This configuration can be changed in the User Preferences dialog box (choose **Edit > Preferences**) in the Curve pane (see [Figure 15](#)).

You can change the minimum value using the **Min Plot Value** field. By default, this value is 1e-20. The minimum value is 1e-300. If you specify a value less than the minimum, the value will be rounded up to the minimum value.

Under Performance, you can use the available fields to improve the performance of Sentaurus Visual when displaying large .plt files. You can reduce the number of points used to define a curve displayed in a plot, which decreases the time taken to draw a curve with a large number of points.

Chapter 2: Basic Operations

Performance Options

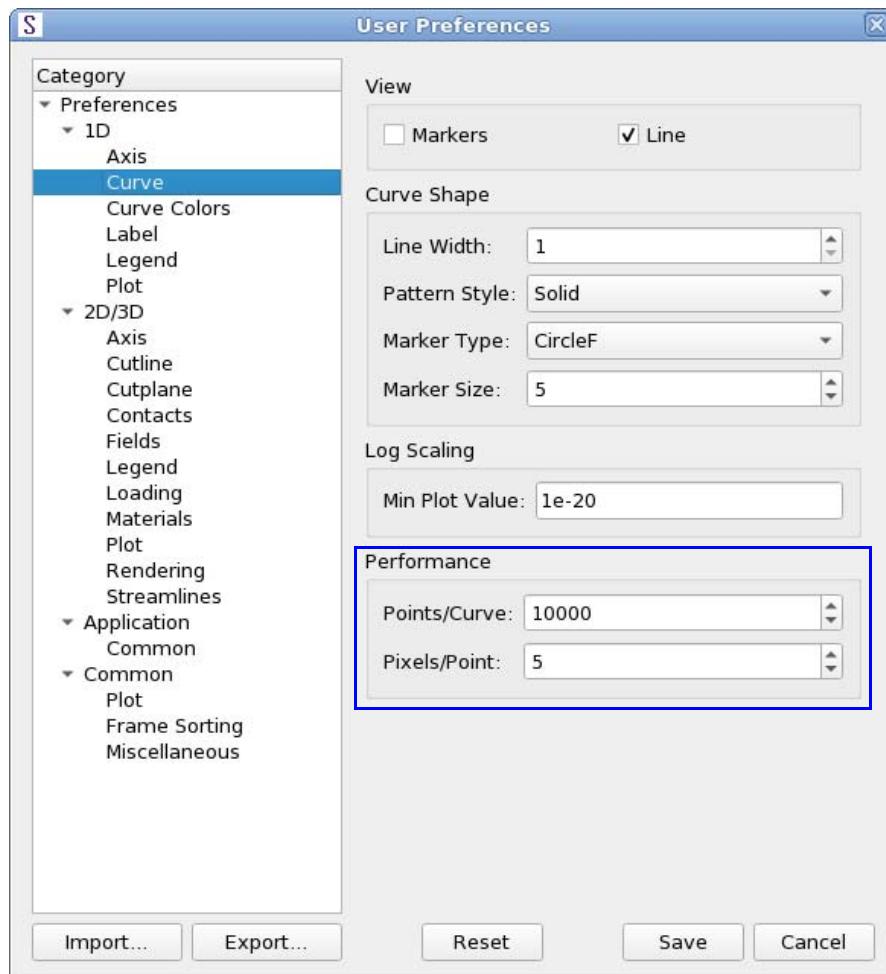
You must define the following fields:

- **Points/Curve** specifies the lower limit at which to activate the Level of Detail algorithm when displaying a curve. The default value is 5000. This means that any curve containing more than 5000 points will switch on this algorithm when displaying the curve. Any curve with fewer than 5000 points will not use this algorithm.
- **Pixels/Point** defines the distance in pixels where only one point will be displayed. The default value is 5. This means that any point that is *more than 5 pixels away* from an already plotted point will be plotted. Any point that is less than 5 pixels away from an already plotted point will not be plotted.

Note:

For large .plt files, using the Level of Detail algorithm increases performance, but the accuracy of the drawn curves decreases.

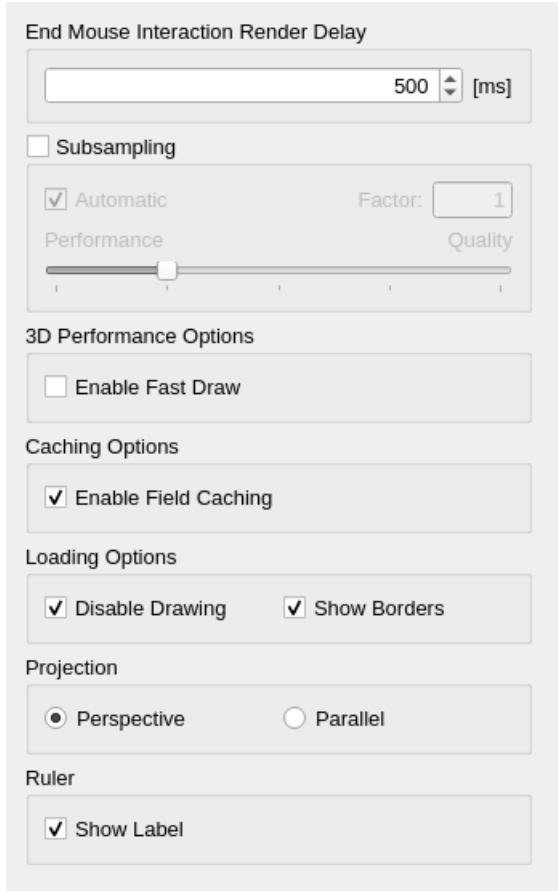
Figure 15 Options for xy plots in User Preferences dialog box



Advanced Options (2D and 3D Plots Only)

Advanced configuration of rendering can be changed in the User Preferences dialog box (choose **Edit > Preferences**). In this dialog box, in the Category pane, expand **2D/3D > Rendering**. [Figure 16](#) shows the rendering fields available.

Figure 16 Performance options



Advanced options include setting the rendering delay after a mouse operation, modifying the quality of the subsampled interactive structure, and enabling the caching functionality for fields:

- The **End Mouse Interaction Render Delay** field adjusts the delay after interacting with the structure in subsampling or fast draw mode, to redraw the detailed geometry.
- The **Subsampling** option enables the structure to be rendered with fewer points, which optimize the interactive performance with little degradation of the rendering quality.

Chapter 2: Basic Operations

Performance Options

- If you select the **Automatic** option, Sentaurus Visual automatically renders the subsampled structure. When you select the **Automatic** option, the value in the **Factor** field is used to fine-tune the algorithm to either performance or quality. A higher value means a higher quality subsampled structure, and a lower value means a lesser quality structure but with better interactive performance.
- If you do not select the **Automatic** option, you can use the Performance/Quality slider to manually choose the quality of the subsampled structure. Moving the slider to the left prioritizes interactive performance, or moving the slider to the right prioritizes rendering quality.
- The **Enable Fast Draw** option enables drawing of the boundaries only of 3D structures.
- The **Enable Field Caching** option helps you to obtain faster transitions between different field visualizations. When it is selected, this option avoids the recalculation of visualization field data that has already been loaded and for which its configuration has not changed.
- The **Disable Drawing** option helps you to improve the loading of files in Sentaurus Visual. This option switches off plot drawing when loading several files and then switches it on when the loading of files is finished.
- The **Show Borders** option switches on or off the default visualization of region border lines in 2D and 3D plots.
- The **Projection** options allow you to set the default value of the camera projection to either perspective or parallel coordinates (see [Interacting With 3D Plots on page 121](#)).
- The **Show Label** option sets the default behavior for the ruler to show the distance label in the plot or to not show the distance label (see [set_ruler_prop on page 369](#)).

The effects of rendering the structure with subsampling or with the **Enable Fast Draw** option selected are shown in [Figure 17](#).

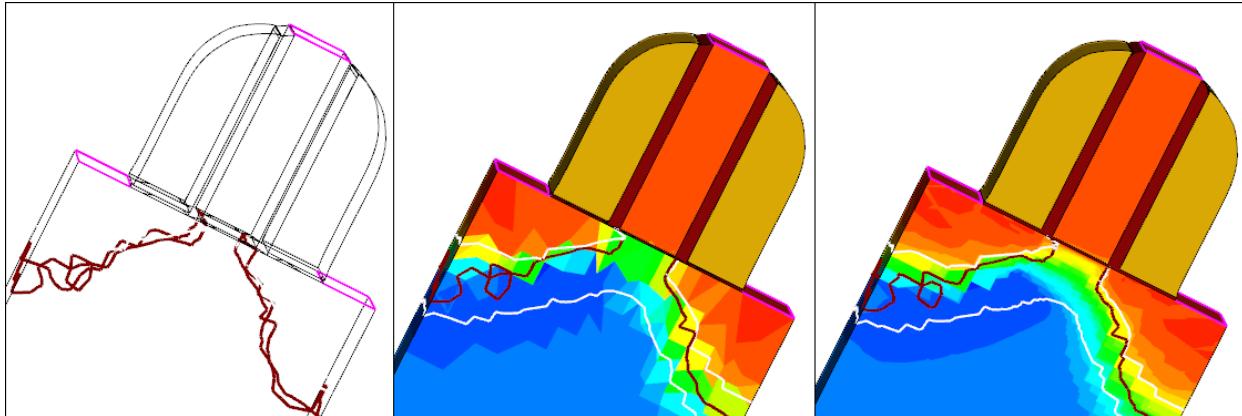
Note:

These changes are active only when in GUI mode. When the operation is completed, the full rendering is shown.

Chapter 2: Basic Operations

Selecting Log Files

Figure 17 (Left) Fast draw enabled, (middle) subsampling selected, and (right) original structure



Selecting Log Files

Note:

This feature applies to both Tcl and the Python scripting language.

By default, Sentaurus Visual generates a standard log file with all the commands executed during a session, named `SVisualTcl.log` for Tcl or `SVisualPy.log` for Python. This log file does not store commands executed by a script or procedure. If this log file already exists, then Sentaurus Visual renames this file by adding the `.BAK` extension.

In addition, by default, another Tcl log file is created if Sentaurus Visual is executed from the command line (or Sentaurus Workbench) with a script. For example:

```
% svisual scriptFile.tcl
```

This additional log file not only stores Tcl commands executed during the session, but also writes Sentaurus Visual Tcl commands executed from a script. In this case, the log file contains more detailed information than the standard Tcl log file. This additional log file is named according to the script executed from the command line (or Sentaurus Workbench), changing the file extension of the script from `.tcl` to `.log`, for example, `scriptFile.log`.

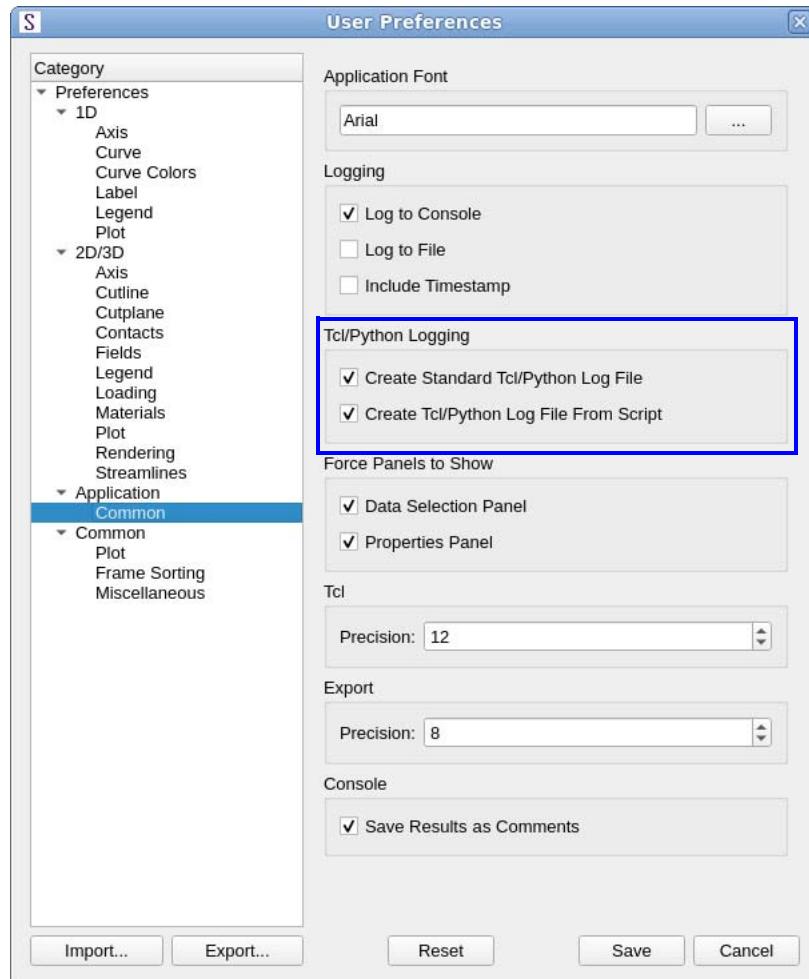
To change the selection of log files:

1. Choose **Edit > Preferences**.
2. In the User Preferences dialog box, expand **Application > Common**.

Chapter 2: Basic Operations

Selecting Log Files

3. Under Tcl Logging, change the selected options as required.



4. Click Save.

3

Working With XY Plots

This chapter presents specific topics about working with xy plots in Sentaurus Visual.

Loading XY Plots

Loading an xy file does not automatically plot the dataset associated with it. Instead, the loaded datasets appear in the Data Selection panel, and a blank plot is created as shown in [Figure 18](#).

The top pane corresponds to the datasets loaded, the middle pane shows the variables present in the selected dataset, and the bottom pane lists the composite variables available in the middle pane.

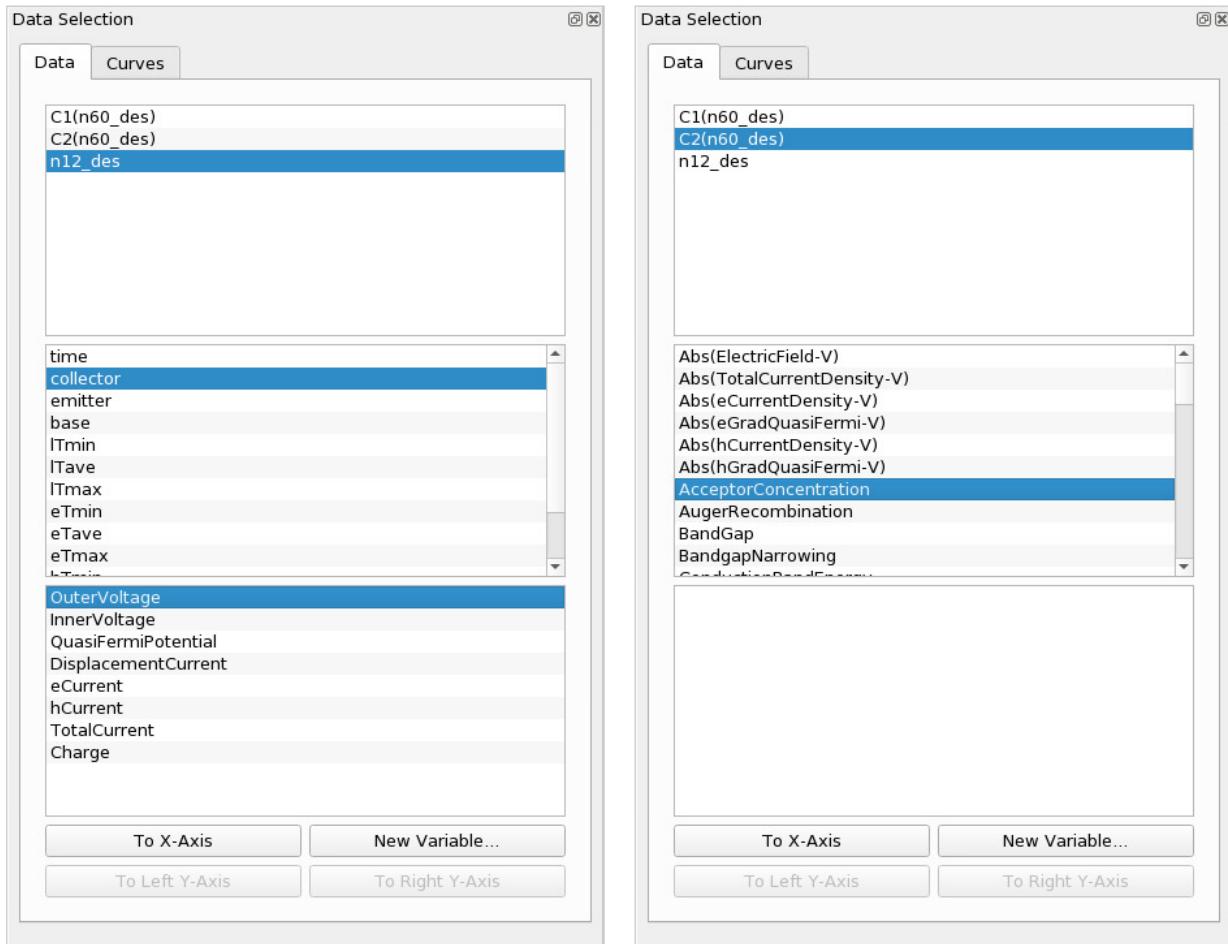
Note:

For .plx files and cutline plots, the x-axes and y-axes are assigned automatically, and the respective curve is generated onto the active plot.

Chapter 3: Working With XY Plots

Loading XY Plots

Figure 18 Data Selection panel showing (left) active datasets of xy plot and (right) active datasets of cutline plot



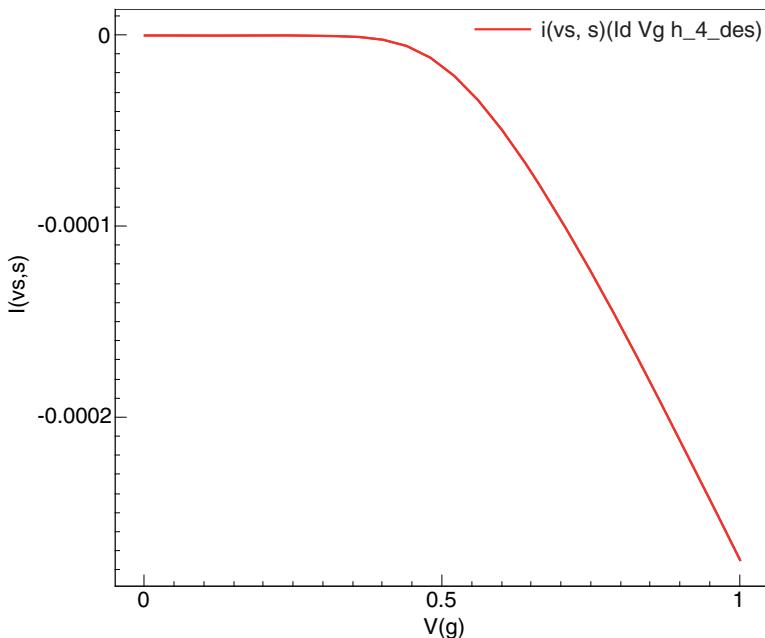
Plotting One Curve

To plot an xy curve, you must select a dataset, and then assign the x-axis and y-axis variables from the available options in the middle pane (or bottom pane if one variable is a composite). The result of selecting *vd* as the x-axis variable and *ib* as the left y-axis variable from the *vd_ib_vb0_vg0.6_vs0* dataset can be seen in [Figure 19](#).

Chapter 3: Working With XY Plots

Loading XY Plots

Figure 19 Plotting a single xy curve



Plotting Multiple Curves

To display multiple datasets on the same plot:

1. Hold the Ctrl key and click the required datasets.

The common variables in the datasets selected are displayed in the middle pane and bottom pane.

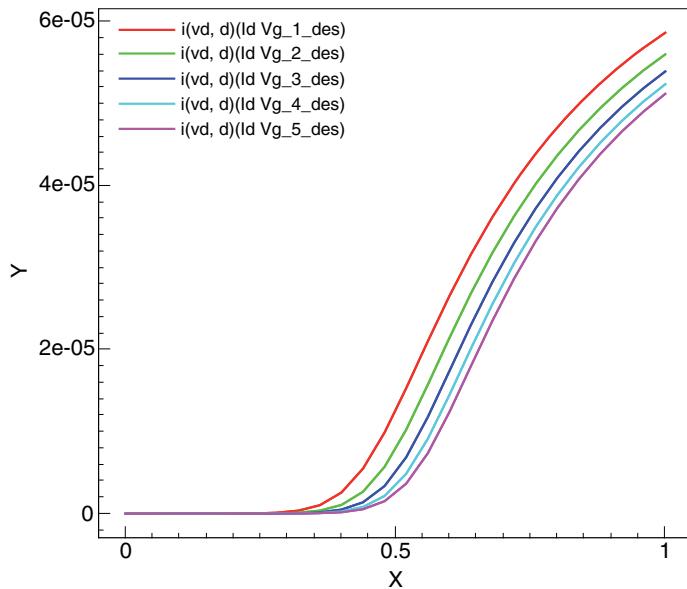
2. Repeat the procedure in the same way as for plotting one curve.

As shown in [Figure 20](#), five datasets were selected; $v(g)$ was selected as the x-axis variable and $id(vd,d)$ was selected as the left y-axis variable.

Chapter 3: Working With XY Plots

Loading XY Plots

Figure 20 Plotting multiple xy curves



When you display multiple curves in an xy plot, the curves are colored according to user-defined rules set in the User Preferences dialog box (expand **1D > Curve Colors**). By default, Sentaurus Visual displays curves using a round-robin logic from a list of colors as shown in [Figure 21 on page 75](#).

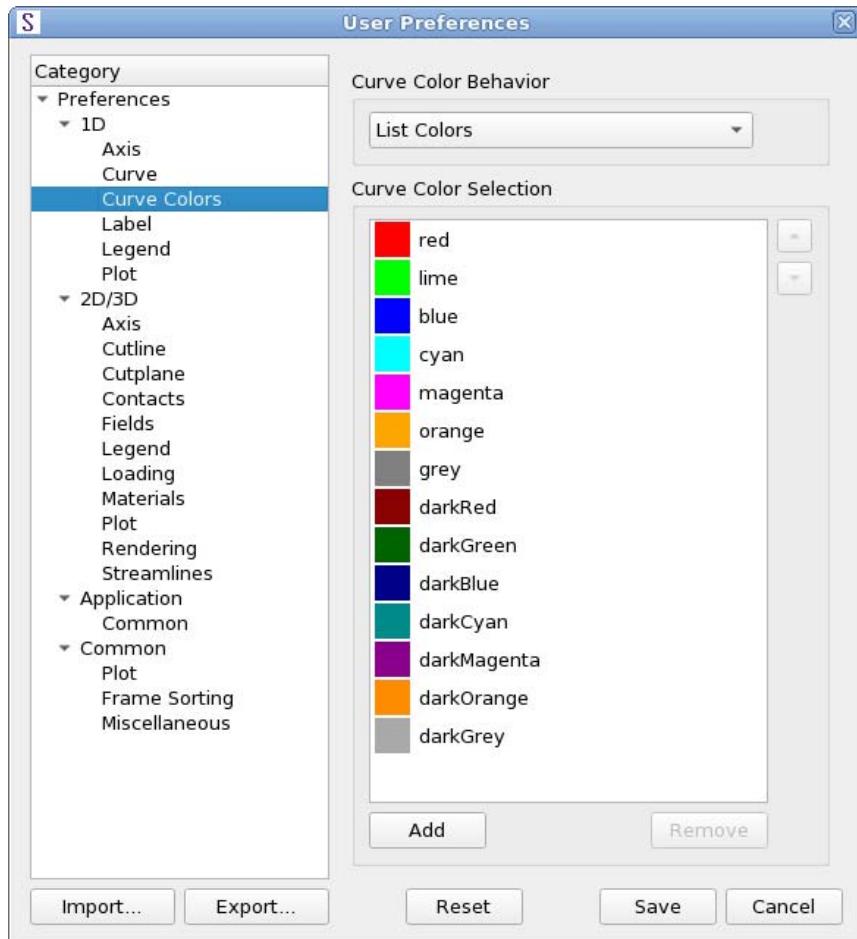
To add a custom color to the list of colors:

1. Under Curve Color Behavior, select **List Colors**.
2. Under Curve Color Selection, click **Add**.
3. In the dialog box that opens, specify a custom color.
4. Click **OK** to close the dialog box.
5. Click **Save**.

Chapter 3: Working With XY Plots

Loading XY Plots

Figure 21 List of default curve colors



To set the same color for all curves in xy plots:

1. Under Curve Color Behavior, select **Constant Color** (see [Figure 22 on page 76](#)).
2. Under Curve Color Selection, select a color from the list.
3. Click **Save**.

You can also map colors to curves using curve labels as well as wildcards. The color-to-curve mapping rules are applied from top to bottom.

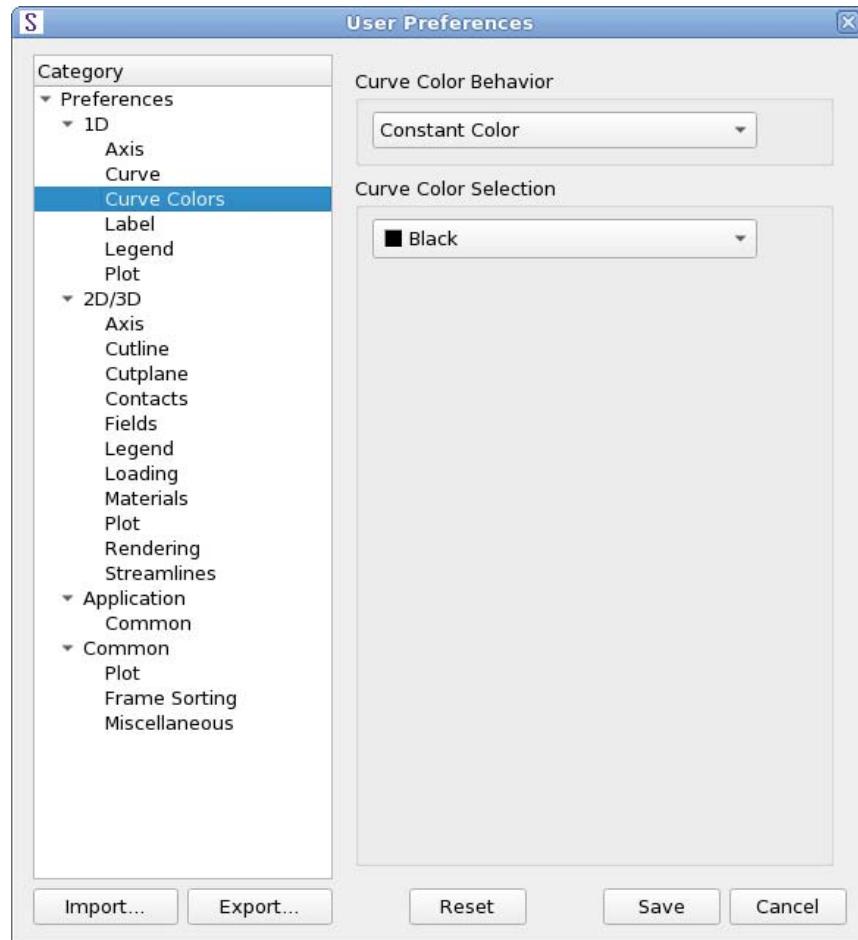
To map colors:

1. Under Curve Color Behavior, select **Map Colors**.
2. Under Curve Color Selection, specify the mapping as required (see [Figure 23 on page 77](#)).
3. Click **Save**.

Chapter 3: Working With XY Plots

Loading XY Plots

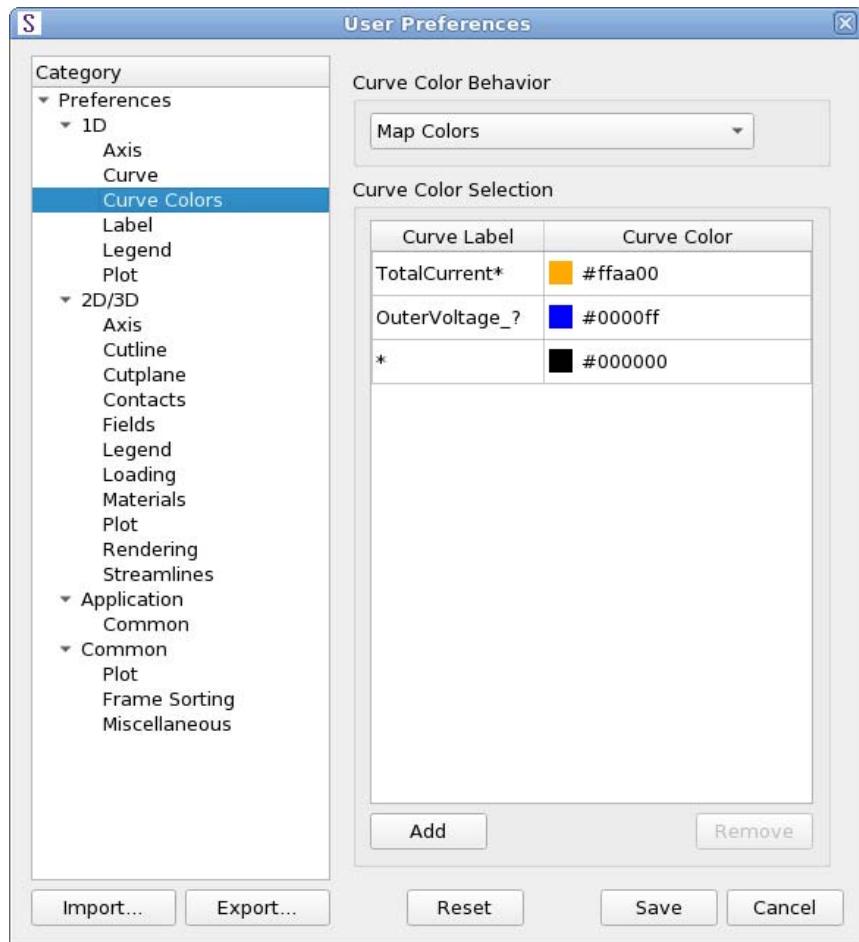
Figure 22 User Preferences dialog box showing a constant curve color has been specified



Chapter 3: Working With XY Plots

Loading XY Plots

Figure 23 Mapping curve colors



Visualizing Multiple TDR States

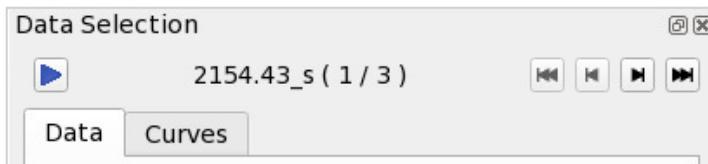
TDR files can contain multiple states of different simulation results. The states are related with regard to geometry. In other words, the main structure data (the number of points) is maintained in all states but their variable data changes. Sentaurus Visual allows you to visualize all the states.

For xy plots, variables can have multiple states. If a curve is created using such variables, a navigation area specific to the existing curve is displayed to allow you to easily navigate through the different states of the curve (see [Figure 24 on page 78](#)).

Chapter 3: Working With XY Plots

Loading XY Plots

Figure 24 Navigation area displaying state name and state index of curve



The navigation area allows you to switch between displayed states quickly with the:

- Next State button ►
- Previous State button ◀
- First State button ▶◀
- Last State button ▶▶

With any change to the state index, the plot title will be updated reflecting the state name.

In addition, the Play button ► allows you to automatically go through all the states, with a 1 second delay between changes, in ascending order. If the last state is reached and the Play button is still active, the sequence will restart.

If displayed curves do not have the same states, the navigation area changes to display the generic state name *state* and the state index of all curves (see [Figure 25](#)).

Figure 25 Navigation area displaying state index of curve



When displaying more than one curve with different state lengths, if you increase the state index to be displayed and one of the curves already reaches its maximum state number, that curve will remain in the maximum valid state number and only the other curves will continue changing accordingly. Switching the state of a curve is a property of a plot, but it cannot be handled in isolation for a single curve being plotted with other curves with different state lengths.

Plotting a multistate curve together with non-multistate (normal) curves will still display the navigation area and the navigation of the multistate curves as previously described. However, normal curves are not affected by any state changes.

Chapter 3: Working With XY Plots

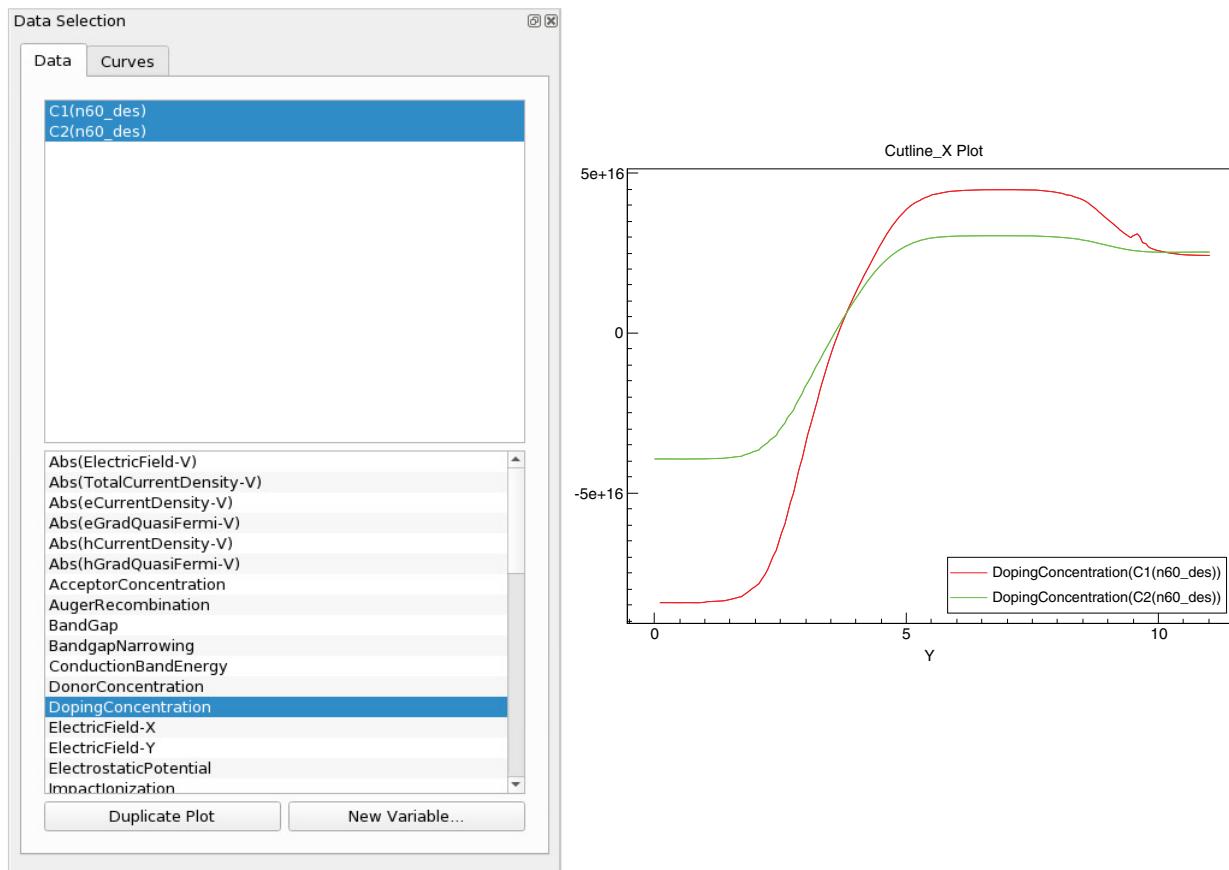
Cutline Plots

Cutline plots have a special interface that allows you to plot new curves by simply selecting one or more variables from a single dataset or a set of datasets. The curve visualization depends only on the datasets and variables selected in the Data Selection panel (see [Figure 26](#) and [Figure 27](#)). You only need to select a new variable (or a set of them) to remove the old curves and to create new ones.

The Data Selection panel does not have the buttons to assign variables to the x-axis or y-axis, but it maintains the **New Variable** button and implements the **Duplicate Plot** button that is used to duplicate the current plot as an xy plot, which enables the features of an xy plot for the currently displayed cutline plot by cloning it.

Cutline plots have a special plot title that follows the format: Cutline_* Plot, where * can be X, Y, Z, or Free, depending on the type of cut. This helps to distinguish cutline plots from xy plots.

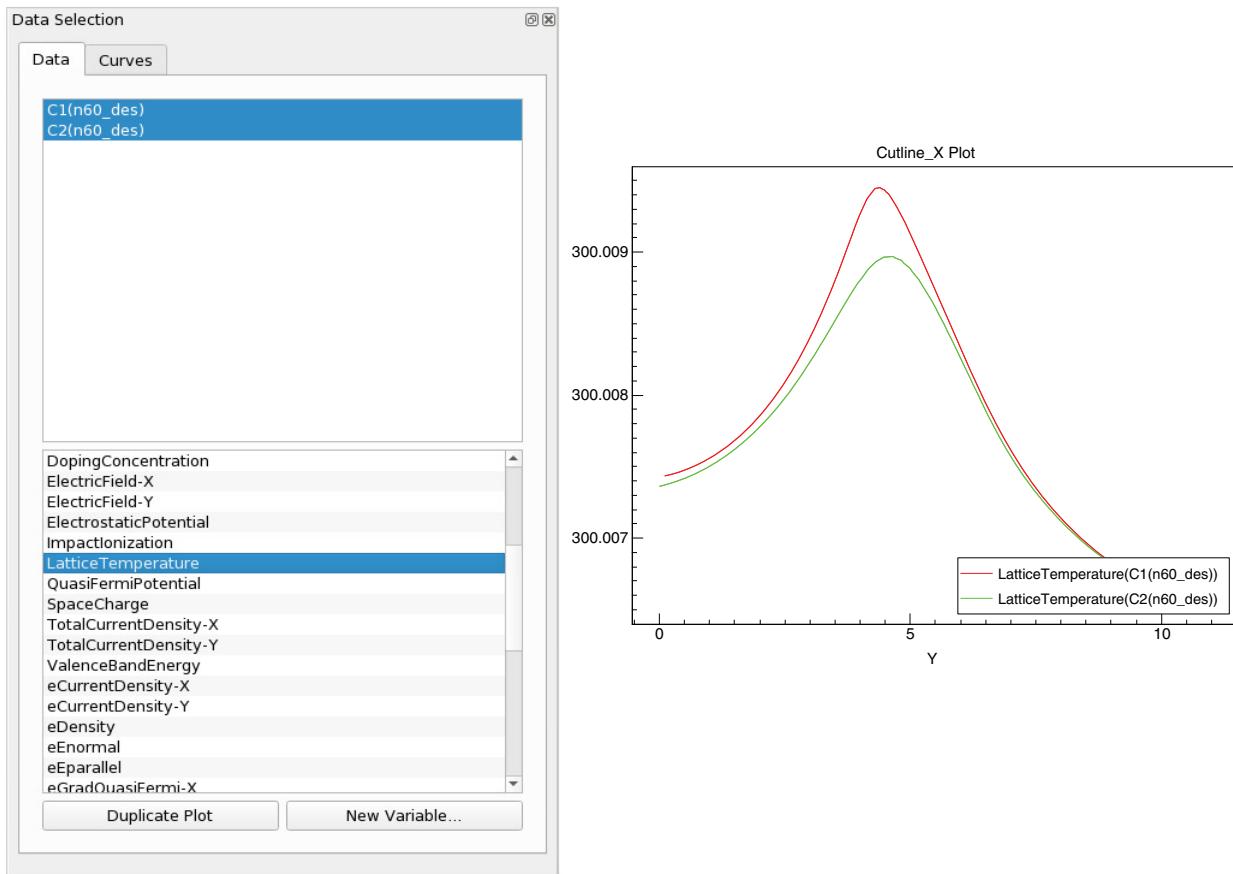
Figure 26 Cutline plot displaying DopingConcentration from two datasets



Chapter 3: Working With XY Plots

Curve Properties

Figure 27 Cutline plot displaying LatticeTemperature from two datasets



Curve Properties

To edit the properties of a curve, select it from the active plot, or you can select the curve from the list in the Data Selection panel. You also can select multiple curves in the Data Selection panel and apply properties to all of them. The Curve Properties panel is displayed (see [Figure 28 on page 81](#)).

In the Curve Properties panel:

- On the **Main** tab, you can change the label of the curve, and select to show or hide the legend and named curve.
- On the **Shape** tab, you can change properties such as curve color, line style, line width, and data pointers.
- On the **Trans.** tab, you can apply curve transformations. It is possible to apply an integration or the first and second derivative to the dataset, or to plot a function using the

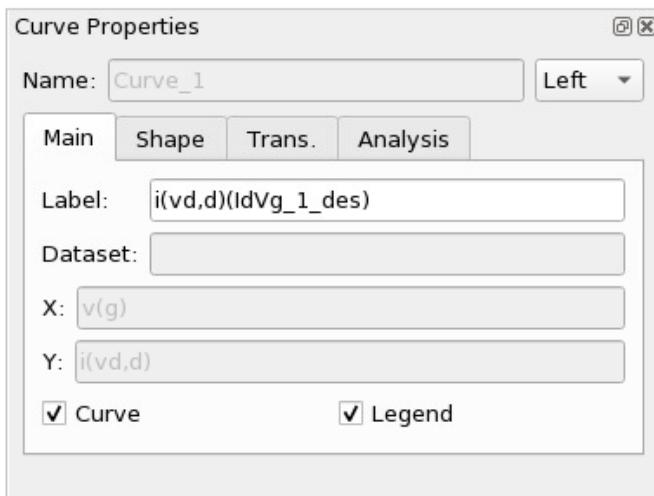
Chapter 3: Working With XY Plots

Curve Properties

dataset values to evaluate the required function. In addition, you can shift and scale the selected curve in the x-axis and y-axis.

- On the **Analysis** tab, you can perform certain analyses on the dataset. For a detailed explanation, see [Computing Electrical Characteristics on page 97](#).

Figure 28 Curve Properties panel



Modifying Properties in Multiple Curves

Sentaurus Visual provides a dialog box where you can modify all curve properties in one view (see [Figure 29](#)). You can modify one property in several curves at the same time.

To modify a property in multiple curves:

1. Choose **Data > Curve Properties**, or click the  toolbar button.
2. Select the required curve rows.
3. Click the column header of the property you want to modify.

A dialog box is displayed where you change the value of the property.

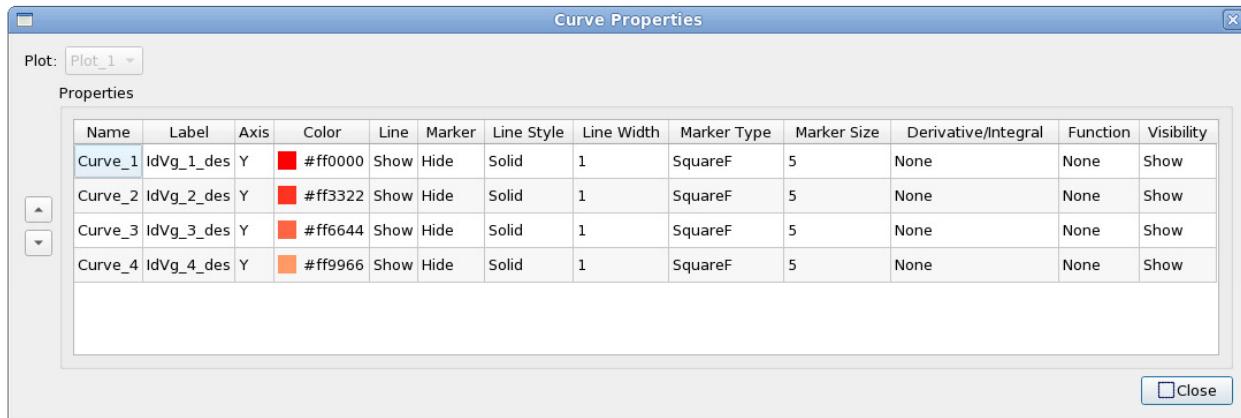
In addition, you can change the order of curves. To do this, select one or more curve rows, and click either the Up arrow button or the Down arrow button at the left of the dialog box.

The order of curves changes immediately and is displayed in the legend as well as the list of curves in the Data Selection panel.

Chapter 3: Working With XY Plots

Plot Area Properties

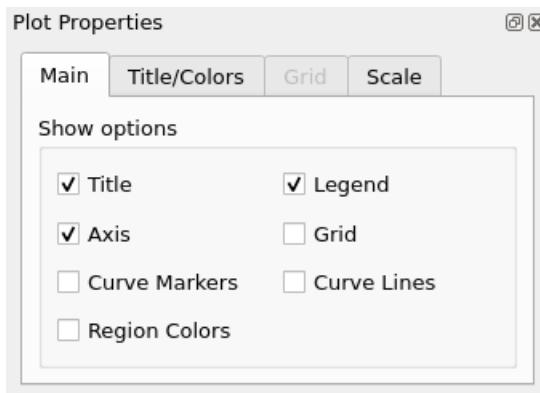
Figure 29 Curve Properties dialog box



Plot Area Properties

The appearance of the plot area can be modified using the Plot Properties panel (see [Figure 30](#)). The Plot Properties panel allows you to change such attributes as the background and foreground colors of the plot, and to show or hide the title, legend, axes, curves, or background color of regions.

Figure 30 Plot Properties panel showing selected options



For example, to hide the legend:

1. Select the plot.
2. On the **Main** tab, deselect **Legend**.

Note:

To show the Plot Properties panel, double-click an empty part of the required plot if another panel is active.

Chapter 3: Working With XY Plots

Legend Properties

See [Quick Access to Tabs of Plot Properties and Axis Properties Panels on page 29](#).

Legend Properties

Legend properties such as position, font attributes, and colors can be changed in the Legend Properties panel (see [Figure 31](#)). To open the panel, double-click the legend of an xy plot.

Figure 31 Legend Properties panel



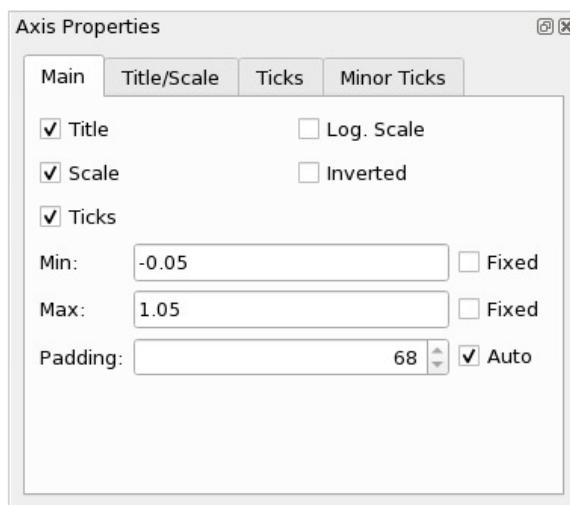
Axis Properties

The appearance of axes can be modified using the Axis Properties panel (see [Figure 32](#)).

To open the Axis Properties panel, double-click any axis in the plot area.

See [Quick Access to Tabs of Plot Properties and Axis Properties Panels on page 29](#).

Figure 32 Axis Properties panel showing selected options



Changing the Axis Padding

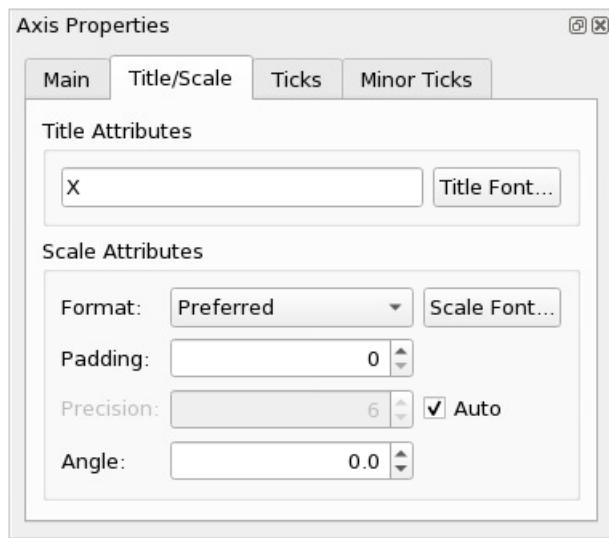
You can change the padding value using the **Padding** field on the **Main** tab of the Axis Properties panel. By default, the **Auto** option is selected for padding, in which case, the padding value is calculated automatically and cannot be edited.

When several xy plots are linked and the **Auto** option is selected, the padding for each axis is the same for all linked plots. The padding value used is the *largest* padding value of each axis from all linked plots. This feature helps to compare curves or plots visually.

Changing the Axis Precision

You can set the precision of the axis (for xy plots and 2D plots) on the **Title/Scale** tab of the Axis Properties panel (see [Figure 33](#)). The precision refers to the number of *relevant* digits after the decimal point.

Figure 33 Axis Properties panel showing the Title/Scale tab where you can change the precision of the axis manually



By default, the precision is chosen automatically based on the dimension of the plot, but this can be manually changed:

1. Deselect **Auto**.
2. In the **Precision** box, select the precision required.

Duplicating XY Plots

You can duplicate an xy plot by choosing **Data > Duplicate Plot**. All properties of the selected plot are replicated in a new plot.

Using Symbols and Scientific Notation in Plots

You can insert Greek symbols, subscripts, superscripts, and math symbols in xy plots by using XML tags in the text box of the plot title, the axis labels, and the legend. The available tags are:

Symbol	Tag	Example	Result
Greek symbol	<greek>	<greek>abcdefghijkl</greek>	αβγδεφη
Math symbol	<math>	$plusminus$	± (See Table 4 .)
Subscript	<sub>	v_d	V _d
Superscript	<sup>	10⁻⁸	10 ⁻⁸
Bold		word	word
Italic	<i>	<i>word</i>	<i>word</i>
Underline	<u>	<u>word</u>	<u>word</u>
Strikethrough	<s>	<s>word</s>	word

[Table 4](#) lists the defined words that are allowed in the `<math>` tag. Only one word is allowed in the `<math>` tag.

Table 4 Defined words that are allowed in `<math>` tag

Word	Result	Word	Result
3root	$\sqrt[3]{}$	laplace	\mathcal{L}
4root	$\sqrt[4]{}$	mho	\mathfrak{U}
contains	\ni	notcontains	\nexists

Chapter 3: Working With XY Plots

Using Symbols and Scientific Notation in Plots

Table 4 Defined words that are allowed in $<\math>$ tag (Continued)

Word	Result	Word	Result
contourintegral	\oint	notelementof	\notin
deriv	∂	notexists	\nexists
doubleintegral	\iint	permille	$\%_0$
e	e	permyriad	$\%_000$
elementof	\in	plusminus	\pm
emptyset	\emptyset	sqroot	$\sqrt{}$
exists	\exists	sum	\sum
forall	\forall	surfaceintegral	$\iint\!\!\!\iint$
fourier	\mathcal{F}	tripleintegral	\iiint
gradient	∇	union	\cup
inf	∞	volumeintegral	$\iiint\!\!\!\iint$
integral	\int		

The text in xy plots also supports HTML symbols without using an XML tag pair, so as to avoid conflicts with reserved characters such as `<` and `>`. To use HTML symbols, you can use either the entity number with the `&#` prefix or the entity name with the `&` prefix, for example, `ΔI [mA/μm]` shows as $\Delta I [mA/\mu m]$. [Table 5](#) lists some of the frequently used HTML symbols.

Chapter 3: Working With XY Plots

Using Symbols and Scientific Notation in Plots

Note:

Not all entities have a name, but all can be referenced by a number.

Table 5 Frequently used HTML symbols

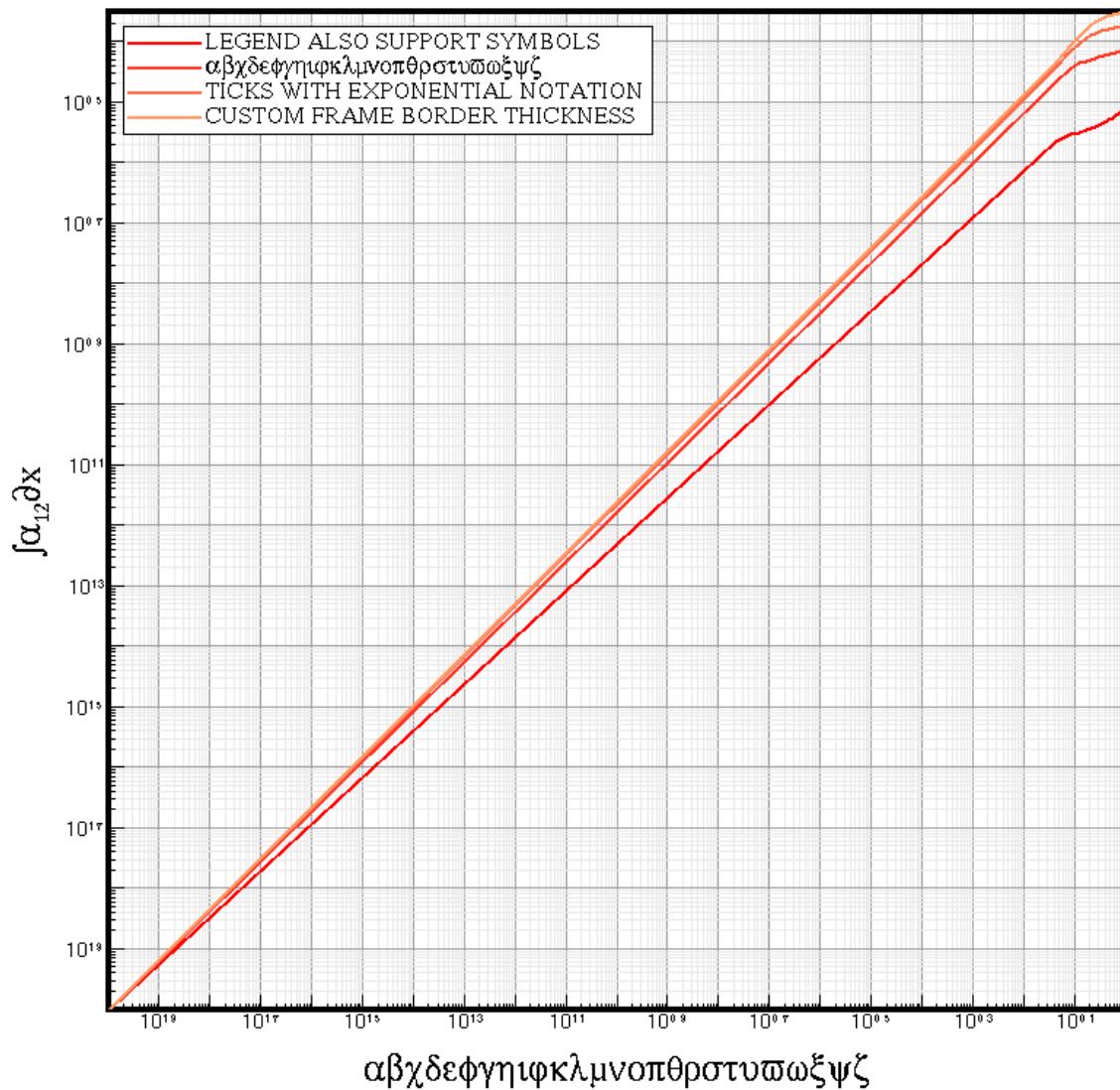
Result	Entity		Result	Entity	
	Name	Number		Name	Number
A	Α	Α	$\sqrt[3]{\cdot}$	—	∛
α	α	α	$\sqrt[4]{\cdot}$	—	∜
Ω	Ω	Ω	\exists	∋	∋
ω	ω	ω	\in	∈	∈
\circledcirc	©	©	\cup	∪	∪
\circledR	®	®	\cap	∩	∩
$^{\text{TM}}$	™	™	\emptyset	—	∮
$\%$	‰	‰	∂	—	∂
$\frac{1}{2}$	½	½	\int	∫	∫
$\frac{1}{4}$	¼	¼	\iint	—	∬
.	—	∙	Σ	∑	∑
$^{\circ}$	°	°	\mathcal{L}	—	ℒ
$\sqrt{\cdot}$	√	√	∞	∞	∞

[Figure 34](#) shows an example of how these symbols are displayed.

Chapter 3: Working With XY Plots

Using Symbols and Scientific Notation in Plots

Figure 34 Plot showing Greek symbols, math symbols, and scientific notation in axis labels and legend

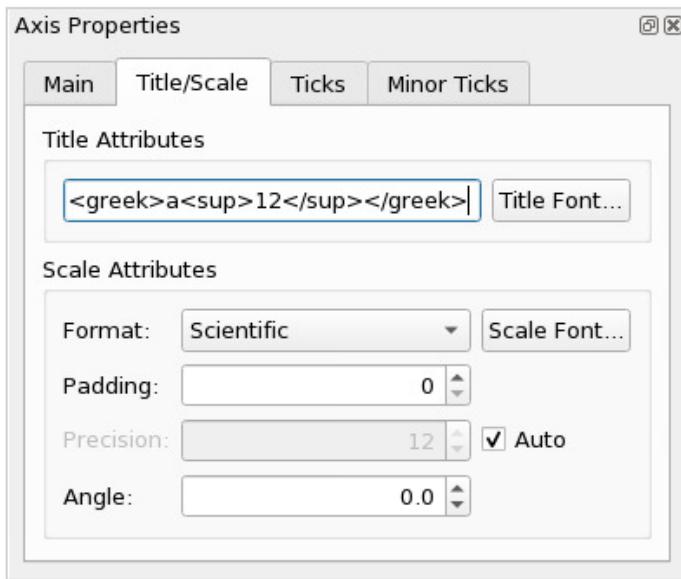


You also can use scientific notation in the axis labels of xy plots by choosing **Scientific** from the **Format** list on the **Title/Scale** tab of the Axis Properties dialog box (see [Figure 35](#)).

Chapter 3: Working With XY Plots

Best Look Option

Figure 35 Axis Properties dialog box showing the selection of scientific notation



Best Look Option

Best look is an option that selects the optimal parameters for the active plot automatically. This option has the following functions:

- Adjusts the zoom level to the optimal position, and changes the x-axis label to the variable used if it is common between curves
- Changes the label of the legend to the variable being plotted (if there is only one curve, it switches off the legend and uses the variable name for the y-axis label)
- Changes the title to the dataset name if all curves share the same dataset

To activate best look, click the  toolbar button.

Plotting Edges of Depletion Regions

You can visualize the edges of depletion regions as vertical lines in xy plots. The x-values of the edges are obtained by using the `get_vertical_lines_prop` command (see [get_vertical_lines_prop on page 293](#)).

To show or hide all edges of depletion regions, click the  toolbar button.

Chapter 3: Working With XY Plots

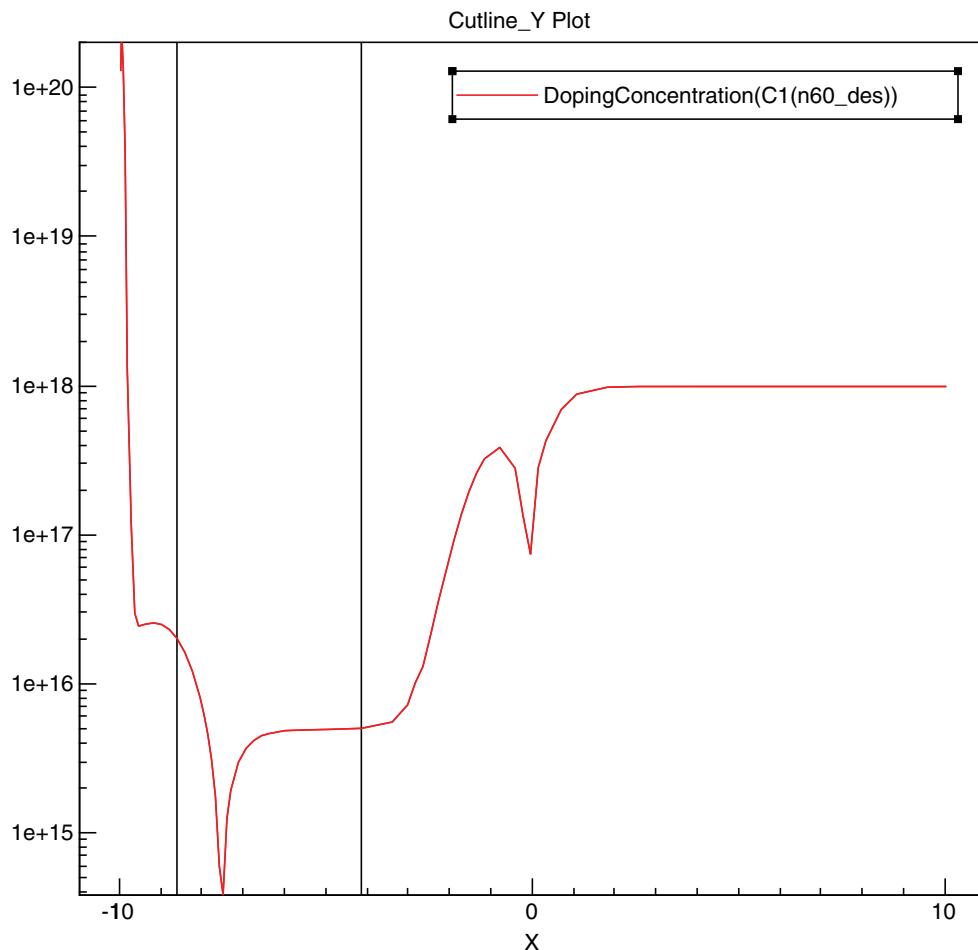
Plotting Edges of Depletion Regions

The dataset must have the following variables defined (variable names are italicized):

- Electron density (*eDensity*)
- Hole density (*hDensity*)
- At least one of the following doping fields:
 - *DopingConcentration*
 - *NetActive*
 - *NetDoping*

If the dataset does not meet these requirements, then the  toolbar button is not available.

Figure 36 Example of visualizing edges of depletion regions; edges are indicated by black lines

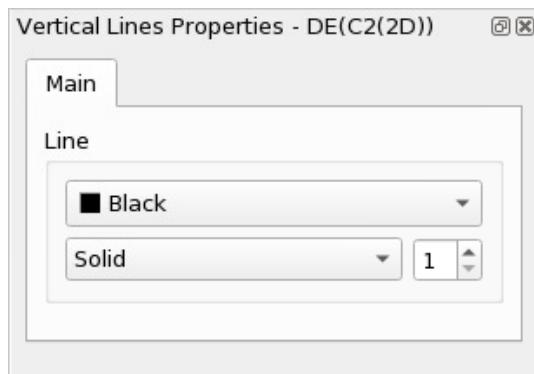


Chapter 3: Working With XY Plots

Plotting Band Diagrams

You can change the color, style, and width of the edges by using either the `set_vertical_lines_prop` command (see [set_vertical_lines_prop on page 379](#)) or the Vertical Lines Properties dialog box (double-click one of the edges to open the dialog box).

Figure 37 *Vertical Lines Properties dialog box showing modifiable properties*

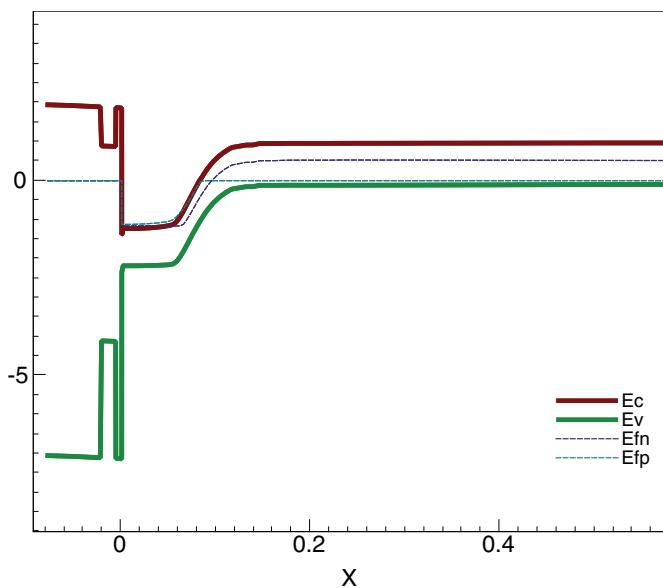


Plotting Band Diagrams

Sentaurus Visual allows you to plot band diagrams, which show the electron energy of the valence band and the conduction band edges versus a spatial dimension.

To create a band diagram for datasets, click the  toolbar button.

Figure 38 *Example of a band diagram*



Chapter 3: Working With XY Plots

Plotting Band Diagrams

The dataset must have the following variables defined (variable names are italicized):

- Conduction band energy (*ConductionBandEnergy*)
- Valence band energy (*ValenceBandEnergy*)
- Electron quasi-Fermi energy (*eQuasiFermiEnergy*) or electron quasi-Fermi potential (*eQuasiFermiPotential*) but not both in the same dataset
- Hole quasi-Fermi energy (*hQuasiFermiEnergy*) or hole quasi-Fermi potential (*hQuasiFermiPotential*) but not both in the same dataset

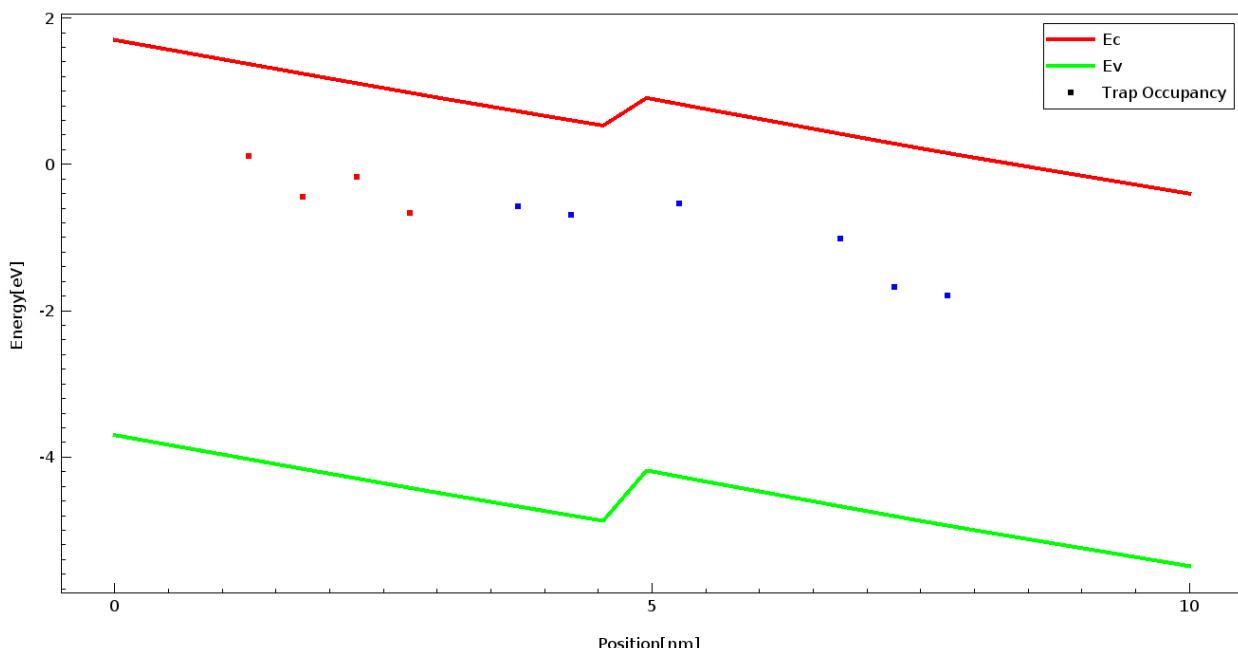
Note:

Typically, band diagrams are created from xy datasets resulting from cuts of 2D or 3D geometries.

Visualizing Discrete Traps in Band Diagrams

A special visualization case is discrete traps in band diagrams. Discrete traps are visualized as rectangular markers by default. However, you can also display traps as circles, diamonds, x-symbols, or plus symbols.

Figure 39 Visualizing discrete traps in a band diagram



Discrete traps have three components: position, energy, and scalar field. The scalar field is used to color the discrete traps from red to blue, with red indicating the maximum value and

Chapter 3: Working With XY Plots

Visualizing Data as a Step-Like Plot

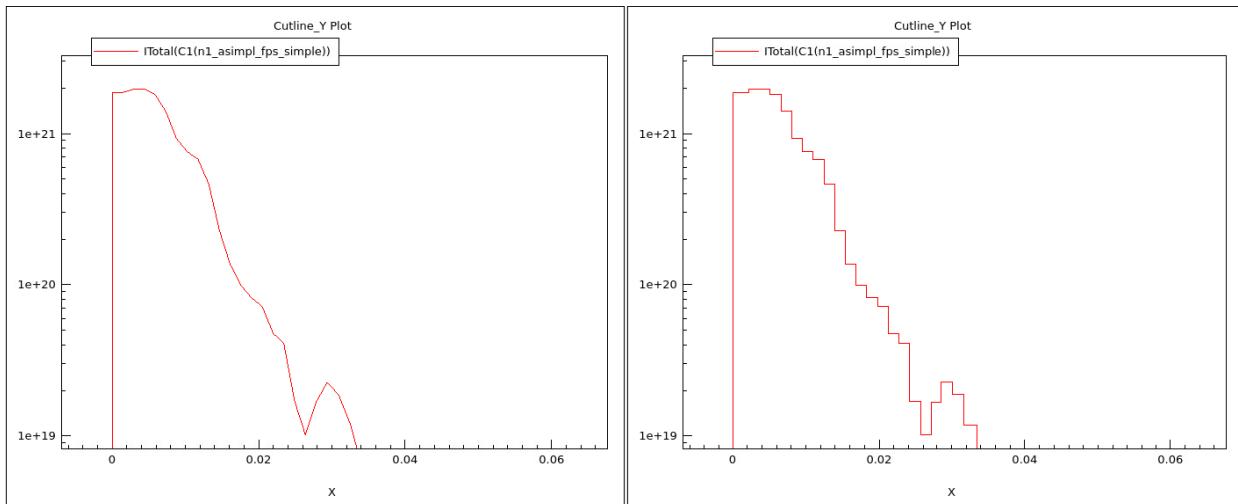
blue indicating the minimum value. You can use the probe tool to obtain the value of each discrete trap (see [Probing on page 94](#)).

Visualizing Data as a Step-Like Plot

A special visualization case is the step-like plot, which is activated when data is elemental (data is on the edges) and the **Convert to Nodal** option on the **Levels** tab of the Data Selection panel is *not* selected.

A step-like plot retains the value of the data until the next value and then jumps to this value, thereby generating a discrete visualization of data. If the **Convert to Nodal** option is selected, then data is interpolated in the center of the cell.

Figure 40 (Left) Typical visualization of data and (right) visualization of data as a step-like plot



Saving the Plot to a Tcl File

Sentaurus Visual can save the current xy plot (including the plot settings, curve data, and displayed curves) to a Tcl file that allows you to recreate the plot easily. You can either choose **Data > Save Plot** or use the `save_plot_to_script` command (see [save_plot_to_script on page 337](#)).

When the file is generated, you can either choose **File > Run Script** or use the `load_script_file` command to reproduce a previously saved plot (see [load_script_file on page 321](#)).

Chapter 3: Working With XY Plots

Probing

The generated file has a particular structure that you can edit for customized loads:

- *Plot Configuration Module*: This module updates the plot with the saved properties, which is performed with the `set_axis_prop`, `set_grid_prop`, `set_legend_prop`, and `set_plot_prop` commands (the `set_axis_prop` command is executed for x-axes, y-axes, and y2-axes).
- *Curves Configuration Module*: This module updates the curves with their saved properties such as color and line width. The update is performed with the `set_curve_prop` command.
- *Drawings Restoration Module*: This module places the drawings in the plot and restores their properties.

Note:

Only text box properties are restored. Other drawings properties must be updated manually.

- *Data Initialization Module*: This module initializes the curve data using a Tcl array structure. The final data is stored as:

```
set datasetList(<curveName>) { {<xAxisData>} {<yAxisData>} }
```
- *Plot and Curves Restoration Module*: After its initialization, the data module creates the xy plot and then creates the curves using the `datasetList` data. In general, this module should never be modified.

Probing

You can sample the intersection value for a horizontal or vertical line depending on whether probing is performed on the x-axis or y-axis. In xy plots, the probe tool uses the interpolation that matches the axis to obtain the value: linear when the axis is in normal mode and log when the axis is in log mode.

To use the probe tool, click the  toolbar button.

Probe Options

In the Probe panel, the following options are available:

- To show the active curve only, select **Only Active Curve**.
- To show guide lines while probing, select **Show Guide Lines**.
- To show and use the closest point on the curve, select **Snap to Point**.

Chapter 3: Working With XY Plots

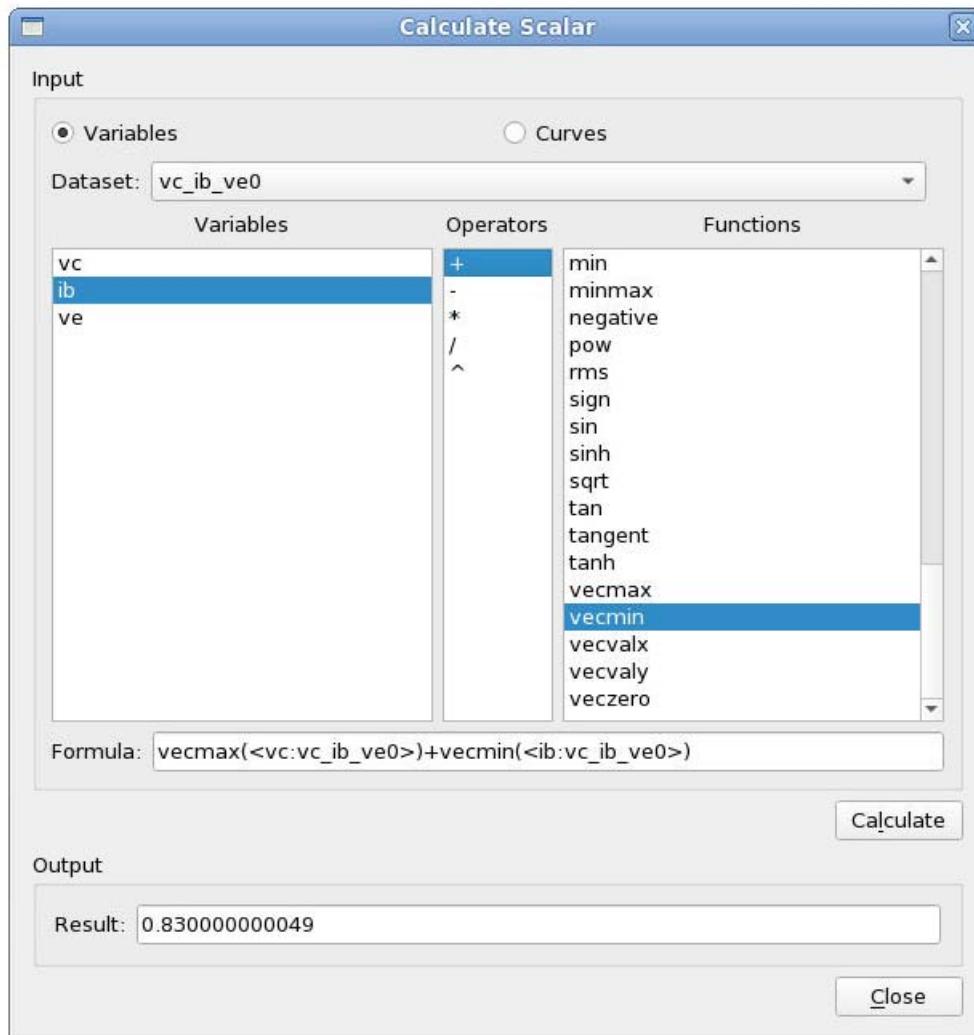
Performing Complex Mathematical Operations on 1D Data

Performing Complex Mathematical Operations on 1D Data

You can perform complex mathematical operations on available 1D data in memory.

To display the Calculate Scalar dialog box, choose **Tools > Calculate Scalar**.

Figure 41 Calculate Scalar dialog box showing the results from the mathematical operations in the Function field



In the dialog box, you create a formula by inserting functions and operators, and using existing 1D data. For the latter, you must select whether the formula will operate on variables (from a dataset) or curves (from a plot).

Chapter 3: Working With XY Plots

Performing Complex Mathematical Operations on Curves

Note:

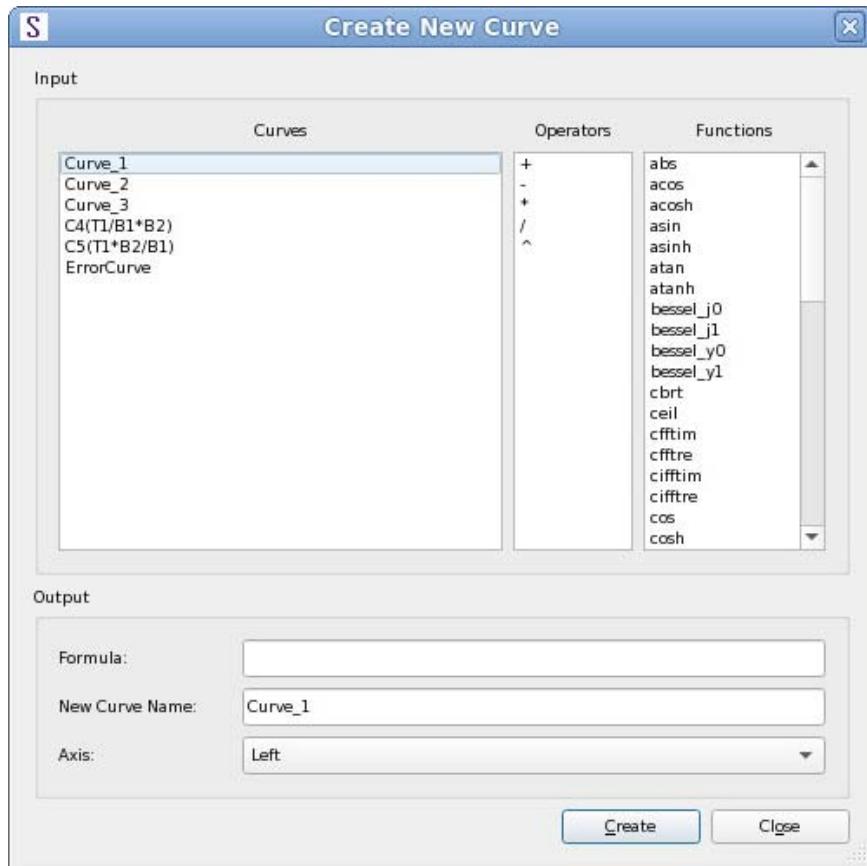
The last operation that encloses the entire set of functions must be a scalar value function. Otherwise, the calculation will fail.

Performing Complex Mathematical Operations on Curves

You can perform complex mathematical operations on available curves.

To display the Create New Curve dialog box, in the Data Selection panel, click the **Curves** tab and then click the **New** button.

Figure 42 Create New Curve dialog box showing available mathematical operations



Curves can have different values of x or length, and all operations are allowed. When this happens, the original curves are interpolated to obtain two curves with the same values of x that allow the operations to be performed point by point.

Computing Electrical Characteristics

You can compute the electrical characteristics of field-effect transistors. Depending on the curve being plotted, you can perform different analyses, such as the threshold voltage, the maximum transconductance value, the drain saturation current, the leakage current, and the output resistances in the linear or saturation region.

To use the analysis tool, click the  toolbar button.

Table 6 Types of curve analysis

Type of analysis	Description
V_{th}	Threshold voltage is defined as the minimum gate electrode bias required to strongly invert the surface under the poly and to form a conducting channel between the source and the drain regions. It can be calculated on I_d-V_g curves.
$G_{M(MAX)}$	Transconductance is a measure of the sensitivity of the drain current to changes in the gate–source bias. It is influenced by gate width, which increases in proportion to the active area as cell density increases. It can be calculated on I_d-V_g curves.
$I_{D(SAT)}$	For a constant gate voltage (V_g), this computes the drain saturation current on I_d-V_d curves.
$I_{D(OFF)}$	For a constant drain voltage (V_d) and a gate voltage (V_g) equal to zero, this computes the leakage drain current on I_d-V_g curves.
R_{out}	R_{out} is the value of the output resistance in the saturation region when $V_g > V_{th}$. This value can be calculated on I_d-V_d curves.
R_{on}	R_{on} is the value of the on-state resistance. It is calculated when the transistor is in the linear region. This value can be calculated on I_d-V_d curves.

For more information about the extraction formulas used to obtain results in the analysis tool, see *Inspect User Guide*, Chapter 8.

Exporting Data From Variables and Curves

You can export data from variables to a .csv file and data from curves to a .csv or .plx file, to allow the use of other tools for further analysis and plotting.

Chapter 3: Working With XY Plots

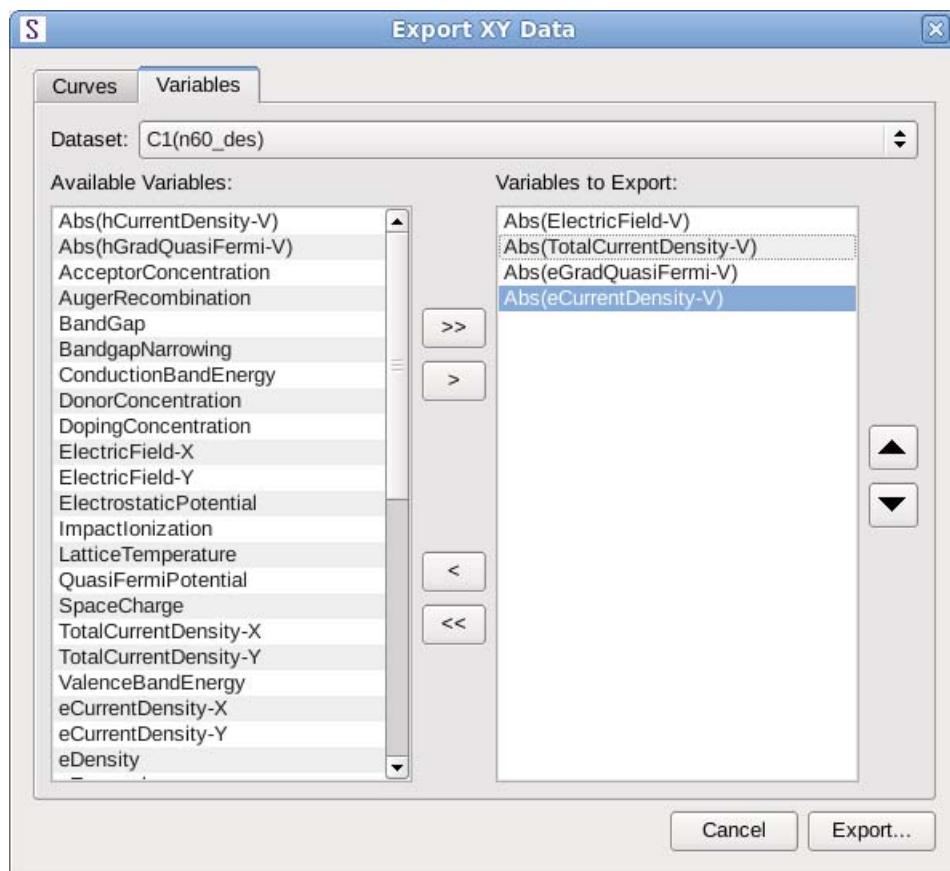
Exporting Data From Variables and Curves

To export data from a variable or curve:

1. Select an xy plot.
2. Choose **Data > Export XY Data**, or click the  toolbar button.

The Export XY Data dialog box is displayed.

3. Select the variables or curves to export by clicking the relevant tab.



4. Export all variables (click the **>>** button), or export only the variables you need (click the **>** button) to the Variables to Export pane.
5. Define the order of variables or curves in the export list using the **Move Up** or **Move Down** buttons to the right of the Variables to Export pane.
6. Click **Export**.
7. In the dialog box that opens, select the file format in which to export the data.

Chapter 3: Working With XY Plots

Exporting Data From Variables and Curves

Note:

The precision of the data exported can be changed in the User Preferences dialog box (expand **Application > Common** and, under Export, specify the precision).

When you export 1D plot variables to a .csv file, Sentaurus Visual might add a row to the file. The data in this row is only used internally. This additional row is always the second row and contains only `none` and `SELECTED` values. You can delete this row.

4

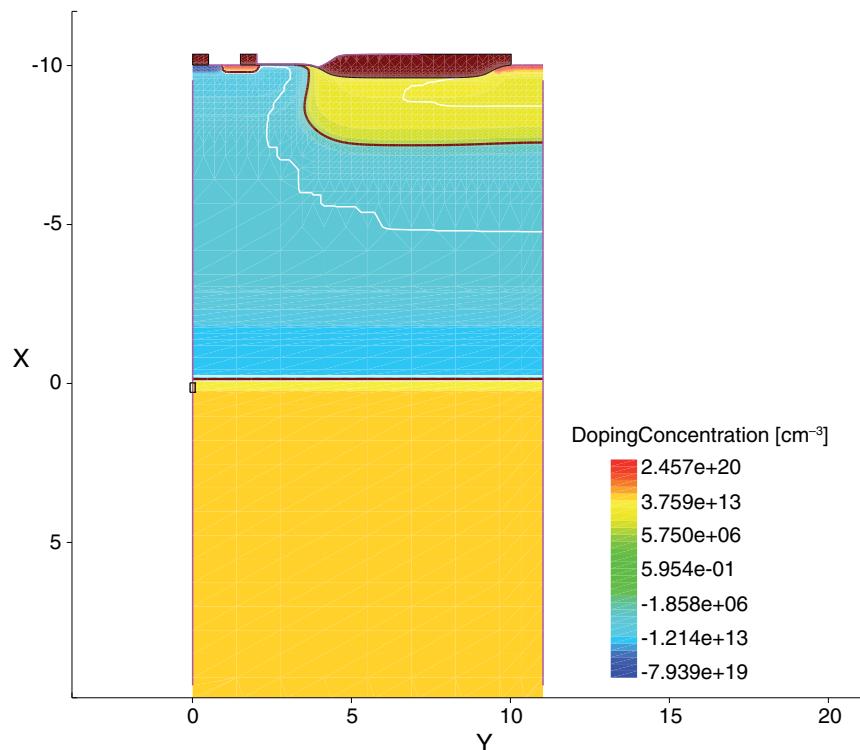
Working With 2D and 3D Plots

This chapter presents specific topics about working with 2D and 3D plots in Sentaurus Visual.

Visualizing 2D and 3D Plots

Sentaurus Visual can visualize simulation results for 2D and 3D plots. When a 2D or 3D file is loaded, Sentaurus Visual automatically generates a plot with the edge, field, and bulk layers activated by default as shown in [Figure 43](#).

Figure 43 Example of 2D plot



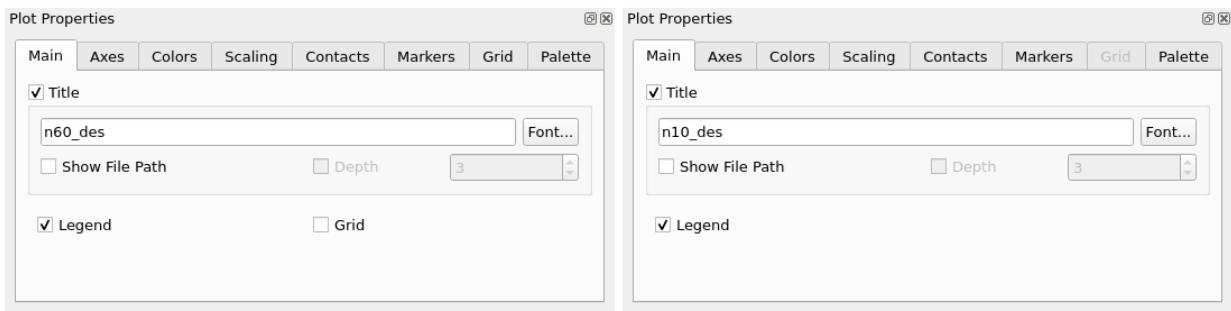
Chapter 4: Working With 2D and 3D Plots

Visualizing 2D and 3D Plots

Changing Plot Properties

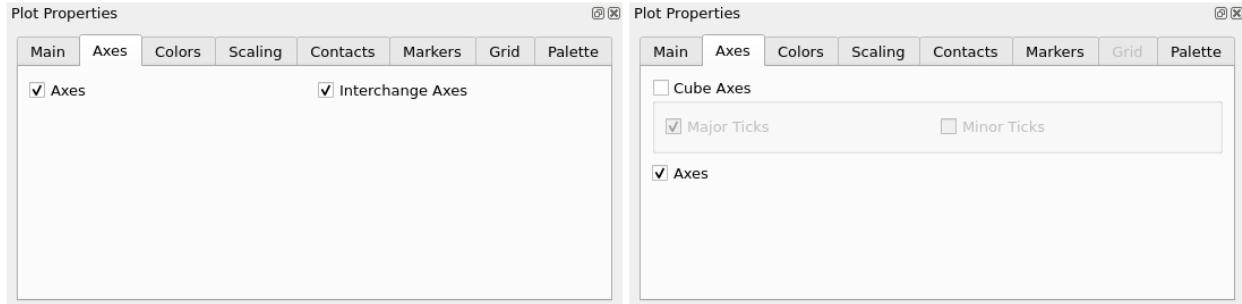
You can easily change the properties of the current active plot using the Plot Properties panel. The **Main** tab allows you to customize the plot title including text, font, and whether you want to include the file path. You can also show or hide the legend and grid (only for 2D plots).

Figure 44 Main tab for (left) 2D plots and (right) 3D plots



The **Axes** tab allows you to show or hide the axes, to interchange the axes (only for 2D plots), and to show or hide the cube axes (only for 3D plots).

Figure 45 Axes tab for (left) 2D plots and (right) 3D plots

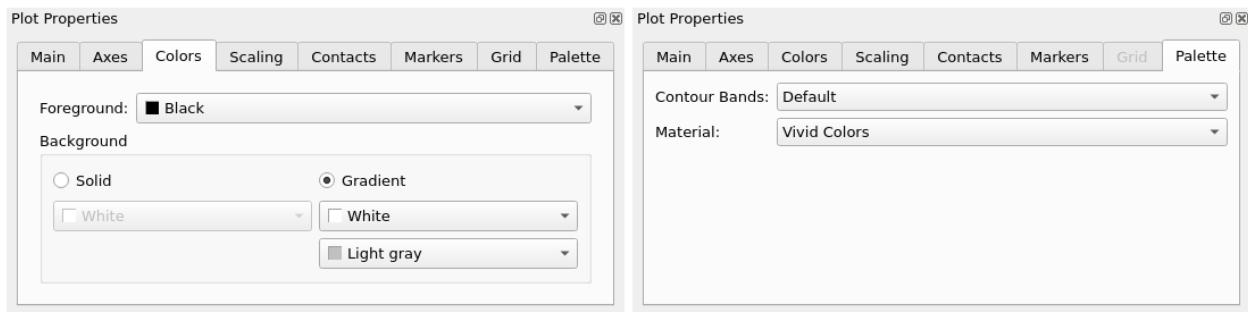


The **Colors** tab allows you to change the background and foreground colors of plots. The **Palette** tab allows you to select the color scheme to use for contouring and materials.

Chapter 4: Working With 2D and 3D Plots

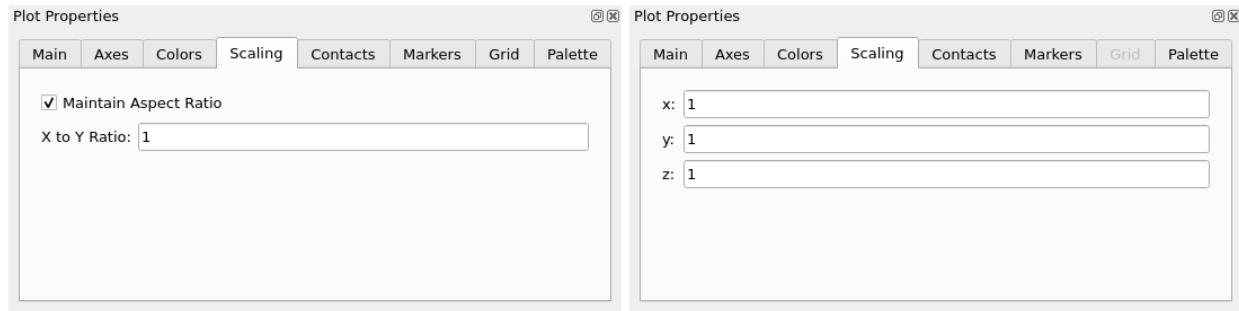
Visualizing 2D and 3D Plots

Figure 46 (Left) Colors tab and (right) Palette tab for 2D and 3D plots



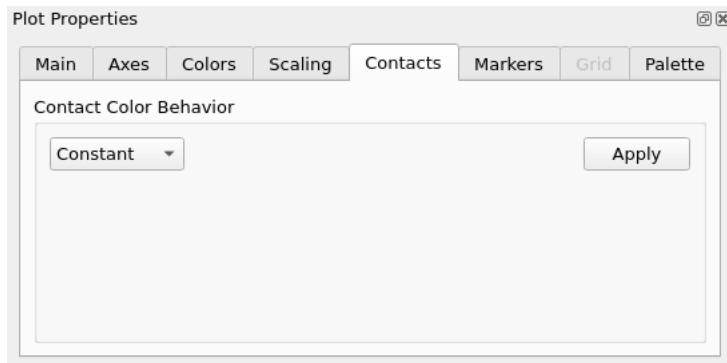
For 2D plots, the **Scaling** tab lets you change the x-to-y ratio. For 3D plots, this tab lets you scale plot axes.

Figure 47 Scaling tab for (left) 2D plots and (right) 3D plots



The **Contacts** tab lets you customize the behavior of the contact color by selecting the behavior from the list.

Figure 48 Contacts tab for 2D and 3D plots

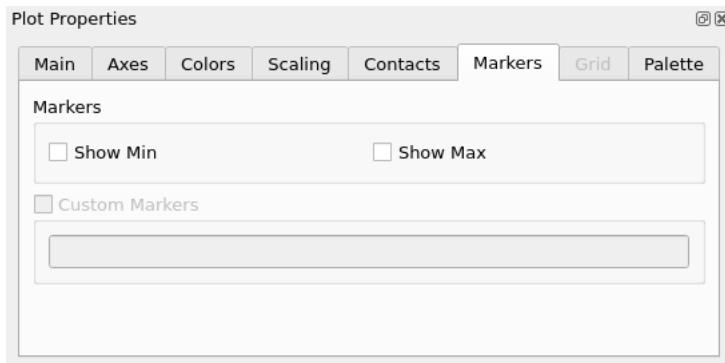


Chapter 4: Working With 2D and 3D Plots

Visualizing 2D and 3D Plots

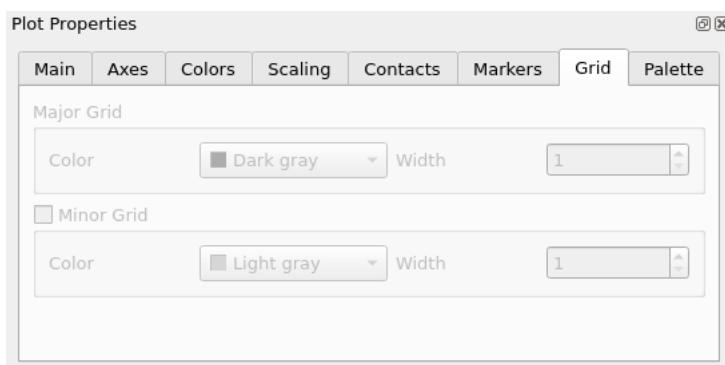
The **Markers** tab lets you display a location marker for minimum and maximum values.

Figure 49 *Markers tab for 2D and 3D plots*



For 2D plots only, the **Grid** tab allows you to customize the color and width of grid lines.

Figure 50 *Grid tab for 2D plots*



Changing the Color Scheme for Materials

You can change the color scheme for materials by using the User Preferences dialog box. This selection is used as the initial color scheme for materials shown in all plots. However, you can change the color scheme independently in each plot using the Plot Properties panel.

Note:

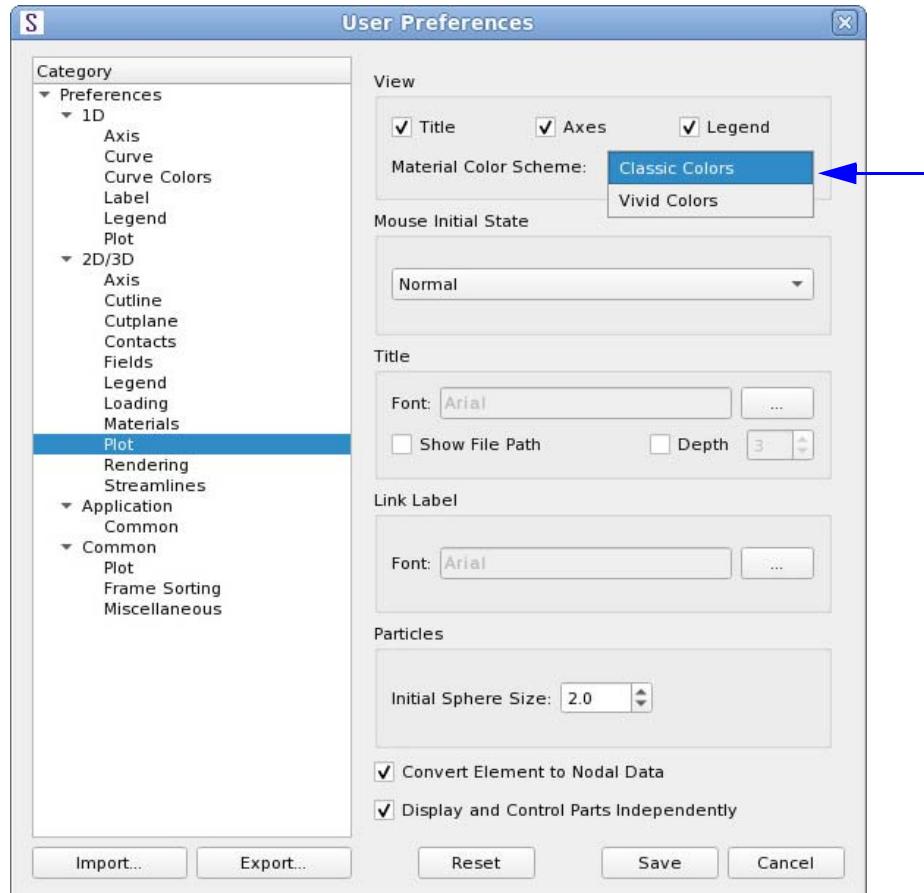
When you change the color scheme, you overwrite any custom colors used for the materials.

Chapter 4: Working With 2D and 3D Plots

Visualizing 2D and 3D Plots

To change the color scheme:

1. Choose **Edit > Preferences**.
2. In the User Preferences dialog box, expand **2D/3D > Plot**.



3. From the **Material Color Scheme** list, select **Classic Colors** or **Vivid Colors**.

The Classic color scheme is the one that has been traditionally used.

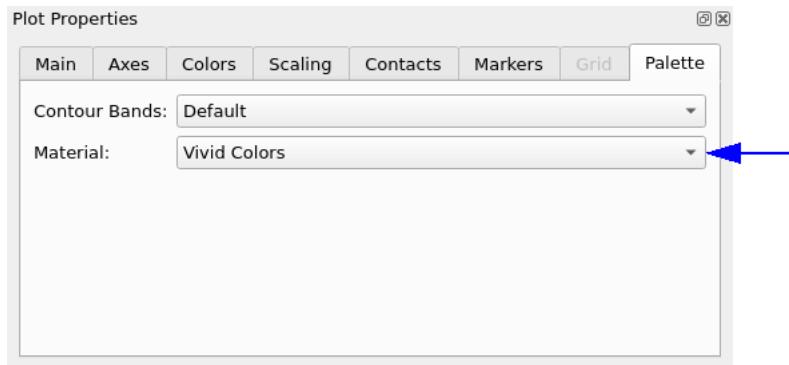
4. Click **Save**.

Chapter 4: Working With 2D and 3D Plots

Visualizing 2D and 3D Plots

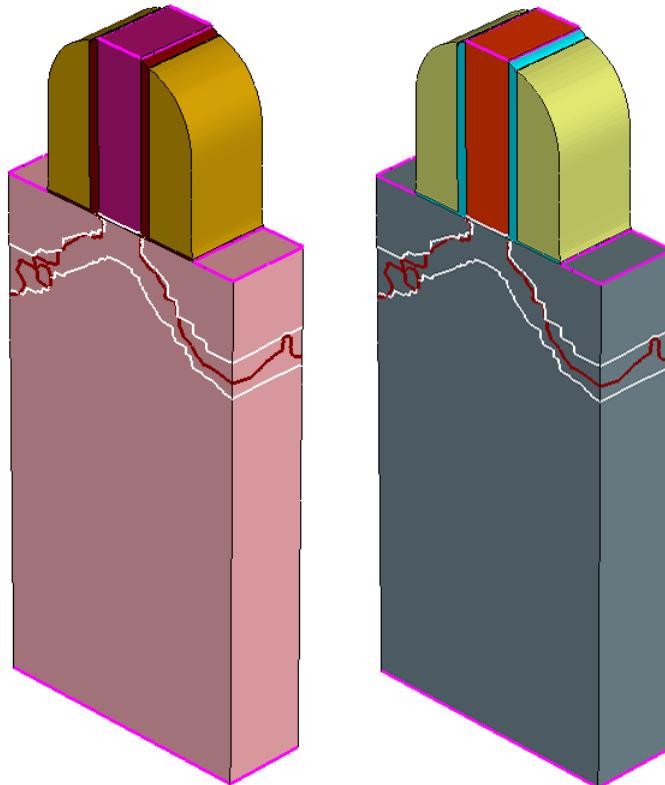
To change the color scheme in a plot:

1. Select the plot.
2. Open the Plot Properties panel.



3. On the **Palette** tab, from the **Material** list, select **Classic Colors** or **Vivid Colors**.

Figure 51 Comparison of (left) Classic color scheme and (right) Vivid color scheme



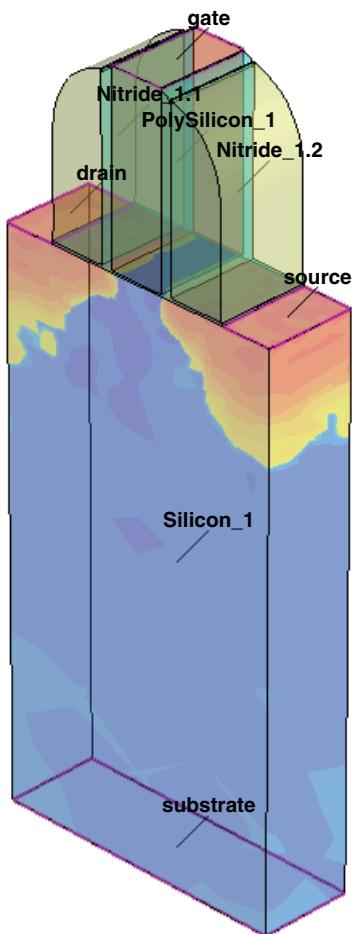
Displaying Region or Material Names

You can display the names of regions or materials in 3D plots as floating captions next to each region. The exact location of each caption corresponds to the center of the bounding box surrounding each region, so the caption might lie outside the region for concave ones.

To show or hide region or material names, specify `-show_caption` or `-hide_caption` in the `set_region_prop` command (see [set_region_prop on page 367](#)).

To adjust the font size and leader visibility of the caption as well as whether region or material names are shown, use the caption-related arguments of the `set_plot_prop` command (see [set_plot_prop on page 362](#)).

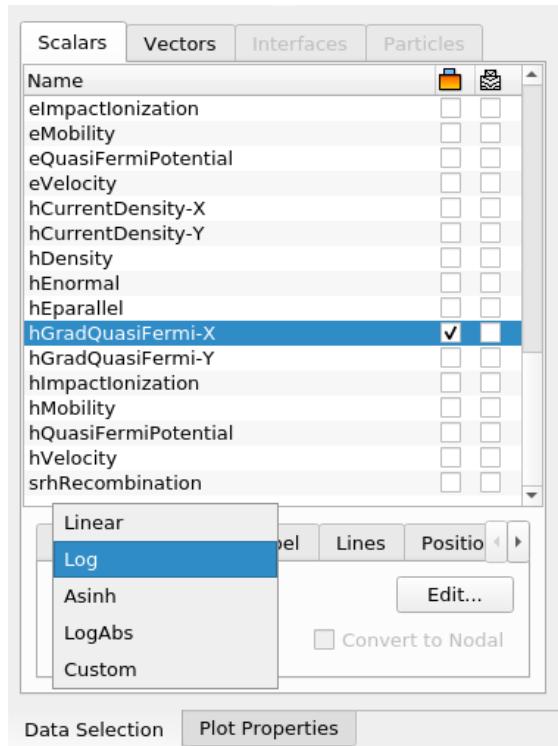
Figure 52 Region captions over a translucent structure



Visualizing Fields

The active field to be visualized in a plot can be chosen in the Data Selection panel (see [Figure 53](#)). The fields can be either scalar or vector. For scalar fields, you can choose the number of colors in which the visualization will be divided as well as the scale, which can be linear, logarithmic, hyperbolic arcsine (Asinh), logarithmic of the absolute (LogAbs), or some custom list of points that you define. Interface and particle fields are on separate tabs from scalar fields.

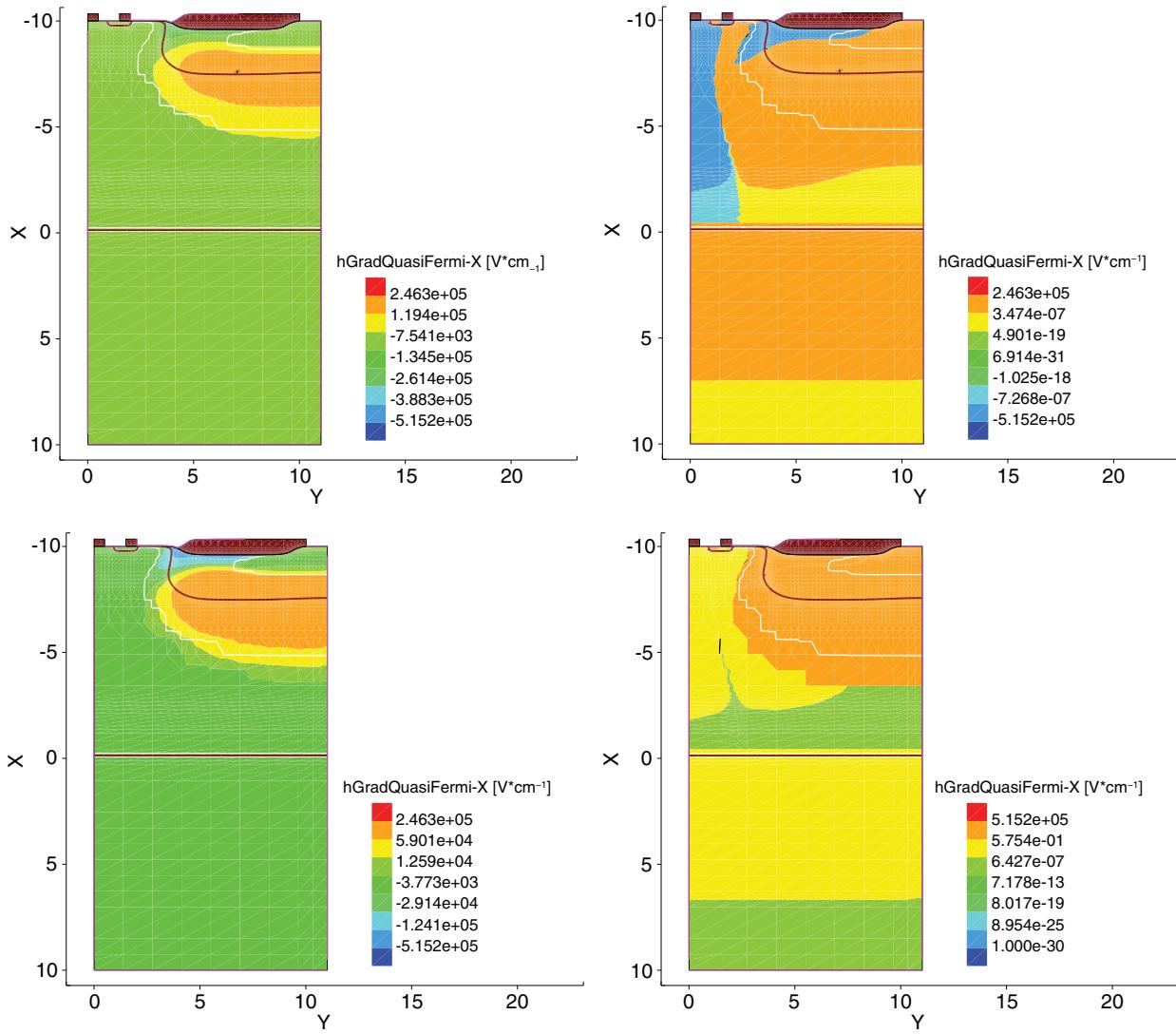
Figure 53 Data Selection panel showing options for visualizing fields



Chapter 4: Working With 2D and 3D Plots

Visualizing 2D and 3D Plots

Figure 54 Scale options for field visualization: (upper left) linear scale, (upper right) logarithmic scale, (lower left) Asinh, and (lower right) LogAbs



Visualizing Fields Defined on Interface Regions

Structures can have fields defined on interface regions. These fields are distinguished from regular region fields by the prefix `Int(field name)`. For example, the name of the field `DopingConcentration` would change to `Int(DopingConcentration)`. The prefix allows you to easily identify such fields on the **Scalars** tab of the Data Selection panel.

For 2D plots, the width of the interface increases automatically to improve visualization of the field data (see [Figure 55 \(right\)](#)).

Chapter 4: Working With 2D and 3D Plots

Visualizing 2D and 3D Plots

Figure 55 Interface data displayed in 2D plots for: (left) regular region field and (right) field defined on interface region

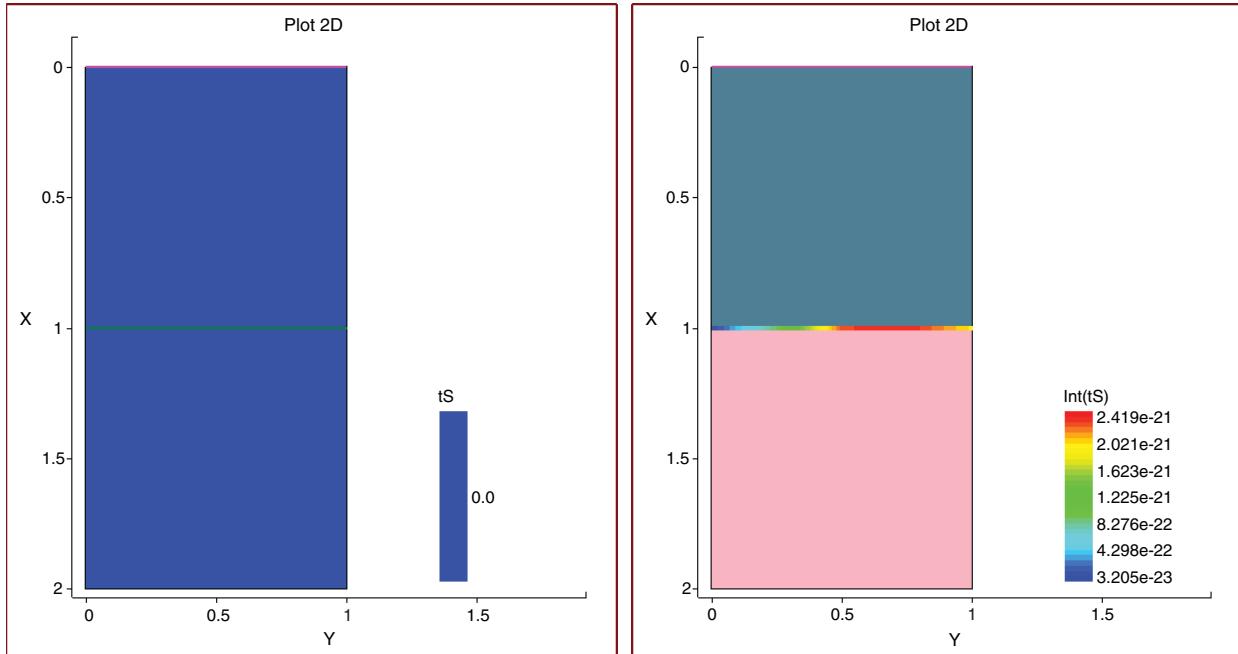
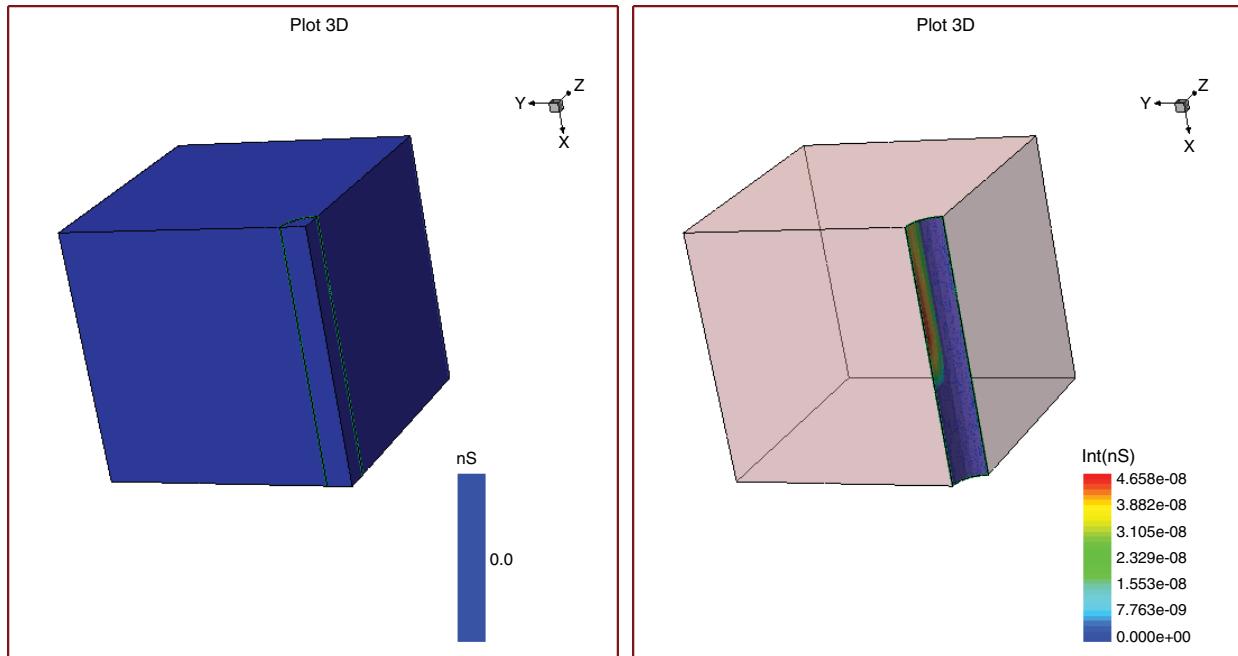


Figure 56 Interface data displayed in 3D plots for: (left) regular region field and (right) field defined on interface region



Chapter 4: Working With 2D and 3D Plots

Visualizing 2D and 3D Plots

Note:

When rendering 3D plots, you can observe the *stitching* phenomenon. It occurs when interface regions share their points with other regions. Both these types of region consist of coplanar polygons where two faces occupy essentially the same space, but neither is in front of the other. The result is a visible *flickering* as affected pixels are rendered from one polygon and then another polygon randomly.

For this reason, when working with 3D plots, you must switch on translucency of other regions to minimize the flickering effect.

Visualizing Automatically Generated Regions

You can visual junction lines and depletion regions.

Junction Lines

Sentaurus Visual calculates automatically the junction line in semiconductor regions where the Doping field is present.

The junction line is visualized as a dark-red contour line and is defined where Doping (DopingConcentration or NetActive) is equal to zero (Doping = 0).

Depletion Regions

Sentaurus Visual calculates automatically the depletion region in semiconductor regions where the Doping field (DopingConcentration or NetActive) and the electron and hole density fields (eDensity and hDensity, respectively) are present.

The edge of the depletion region is visualized as a white contour line. The depletion region is defined by:

$$n \cdot \frac{\text{eDensity}}{\text{Doping}} - p \cdot \frac{\text{hDensity}}{\text{Doping}} = \text{DepletionEdgeValue}$$

where:

$$n = \max\left(\frac{\text{Doping}}{\text{abs}(\text{Doping}) + 1}, 0\right)$$

$$p = \max\left(\frac{-\text{Doping}}{\text{abs}(\text{Doping}) + 1}, 0\right)$$

The DepletionEdgeValue is equal to 0.05 by default. You can modify this value by changing the Sentaurus Visual configuration file (`~/.config/Synopsys/SVvisual.conf`).

Chapter 4: Working With 2D and 3D Plots

Visualizing 2D and 3D Plots

The parameter that defines the DepletionEdgeValue is called `depletion\edgeValue` and belongs to the `PlotHD` group:

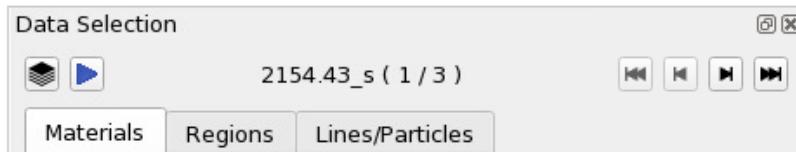
```
...
[PlotHD]
...
depletion\edgeValue=0.05
...
```

Visualizing Multiple TDR States

TDR files can contain multiple states of different simulation results from Sentaurus Process Kinetic Monte Carlo simulations or Sentaurus Device simulations. The states are related with regard to geometry. In other words, the main structure data and regions are maintained in all states but their field data changes. Sentaurus Visual allows you to visualize all the states.

For 2D and 3D plots that are generated from TDR files with multiple states, a navigation area specific to such plots is displayed to allow you to easily navigate through them (see [Figure 57](#)).

Figure 57 Navigation area for 2D and 3D plots generated from TDR file containing multiple states



The navigation area allows you to switch between displayed states quickly with the:

- Next State button ►
- Previous State button ◀
- First State button ▲◀
- Last State button ▶▶

With any change to the state index, the plot title will be updated to show the current state name.

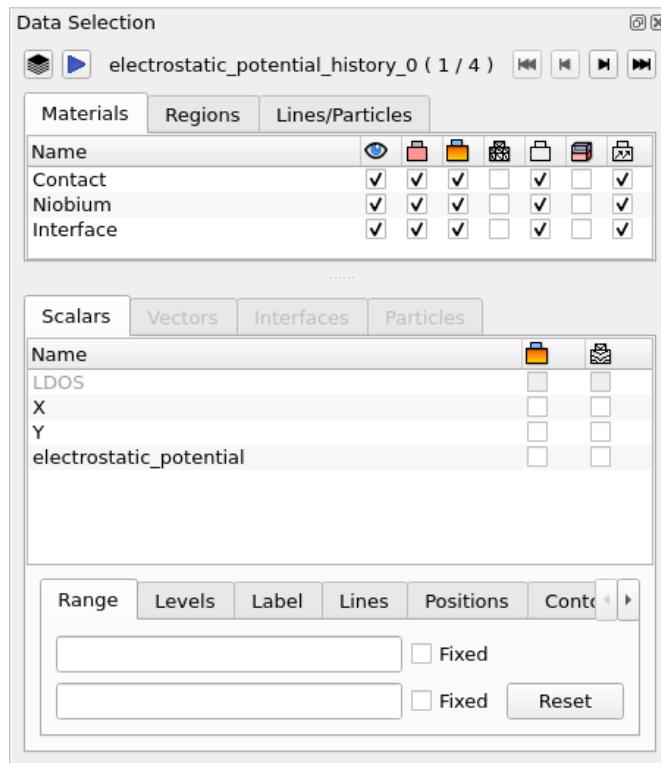
In addition, the Play button ► allows you to automatically go through all the states, with a 1 second delay between changes, in ascending order. If the last state is reached and the Play button is still active, the sequence will restart.

Chapter 4: Working With 2D and 3D Plots

Visualizing 2D and 3D Plots

Fields that are not present on the current selected state are not available in the Data Selection panel and cannot be selected. The plot renders only the corresponding selected fields for that state.

Figure 58 Data Selection panel showing unavailable field on Scalars tab



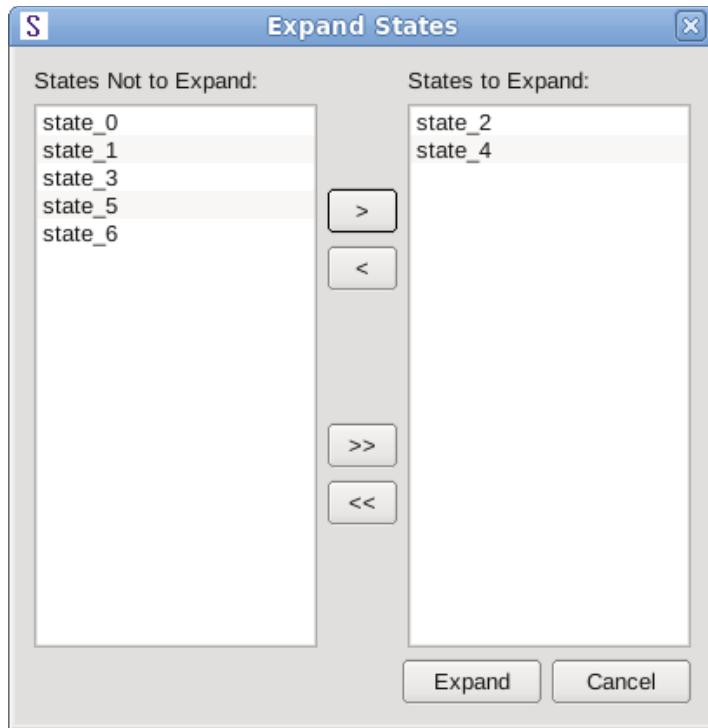
The navigation area for 2D and 3D plots also has the Expand/Collapse States button  that opens the Expand States dialog box where you can select the states to expand (see Figure 59):

- Click the > button to add one state only.
- Click the < button to remove one state only.
- Click the >> button to add all states.
- Click the << button to remove all states.

Chapter 4: Working With 2D and 3D Plots

Visualizing 2D and 3D Plots

Figure 59 Expand States dialog box



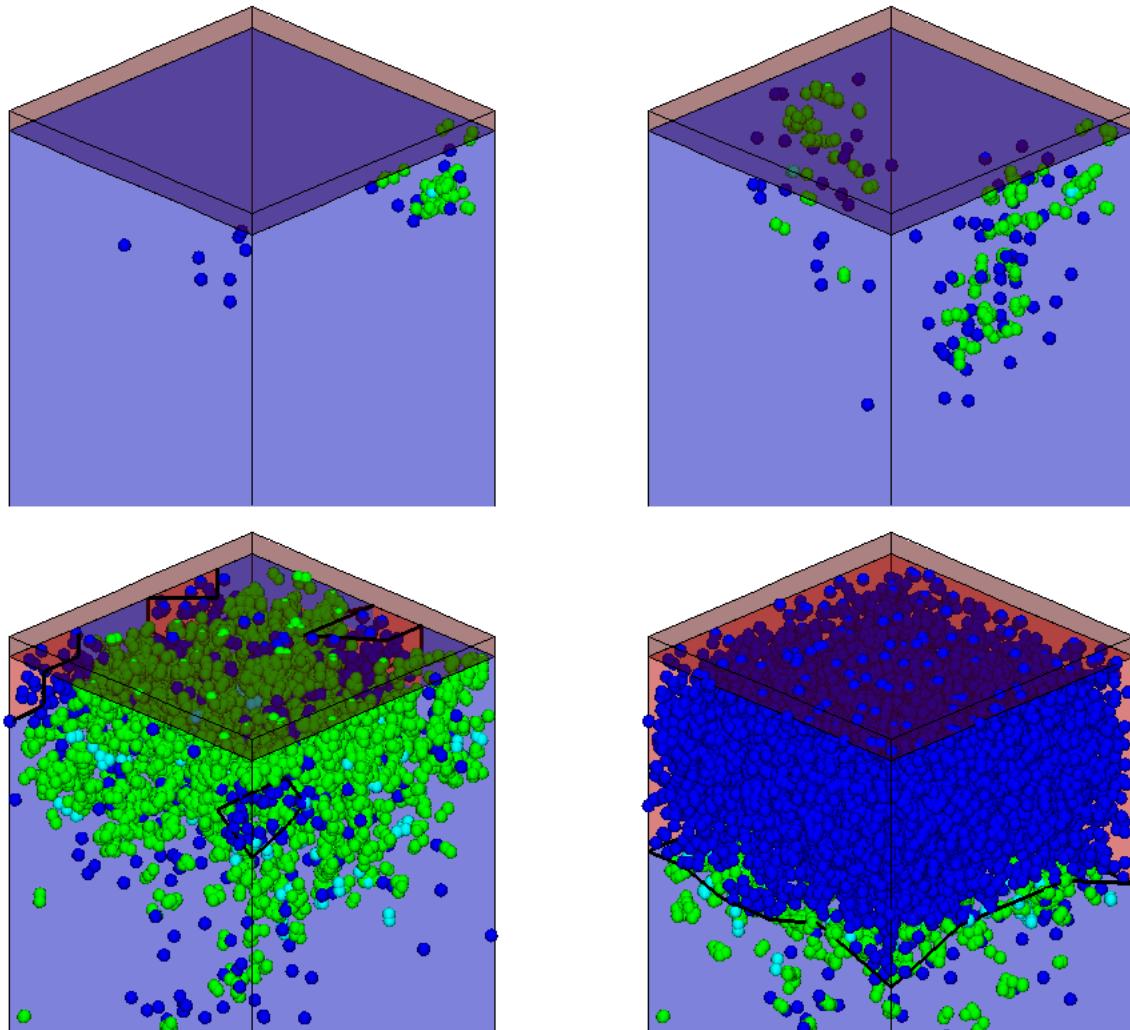
When you click the **Expand** button in the dialog box, all the selected states are expanded to separate plots, so that you can analyze each state one by one (see [Figure 60](#)).

If you click the Expand/Collapse States button again in any expanded plot or the parent plot, all of the expanded plots will collapse, but not the parent plot.

Chapter 4: Working With 2D and 3D Plots

Visualizing 2D and 3D Plots

Figure 60 Example of 3D kinetic Monte Carlo TDR file expanded to show the same structure with changes to the state over time

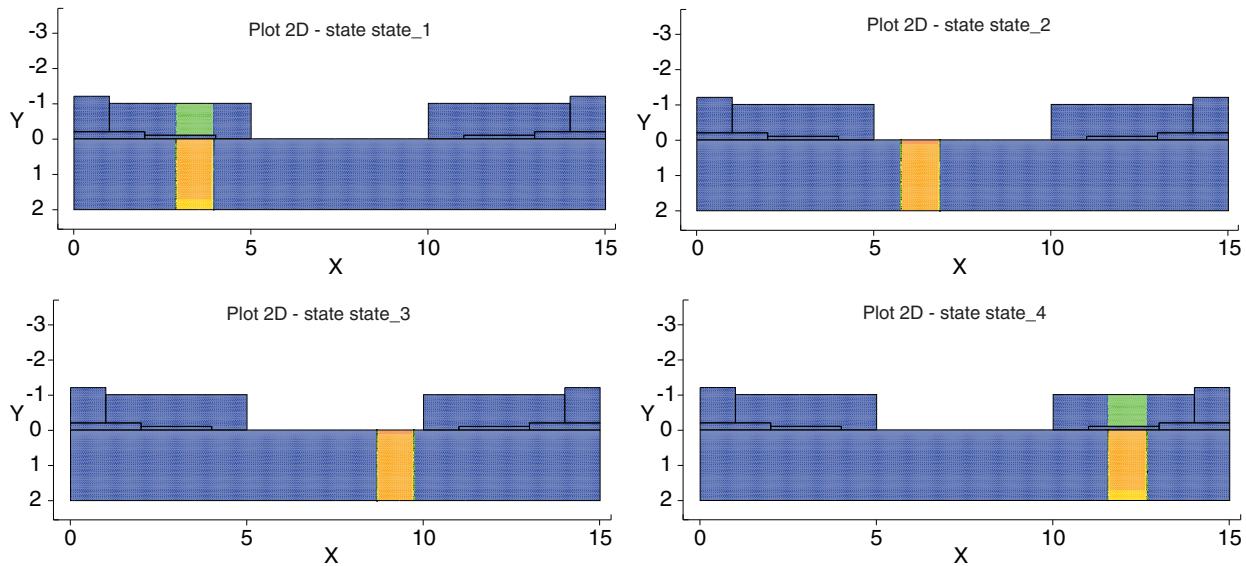


Such plots can also have different field data for each state. Switching states or expanding plots can help you to visualize data (see [Figure 61](#)).

Chapter 4: Working With 2D and 3D Plots

Visualizing 2D and 3D Plots

Figure 61 Example of multistate TDR file that has been expanded to show field data as separate plots



Visualizing Regions With Multiple Parts

Some boundary structures, mainly originating from Sentaurus Topography 3D simulations, can contain regions with multiple parts. Sentaurus Visual can visualize and manipulate these parts independently.

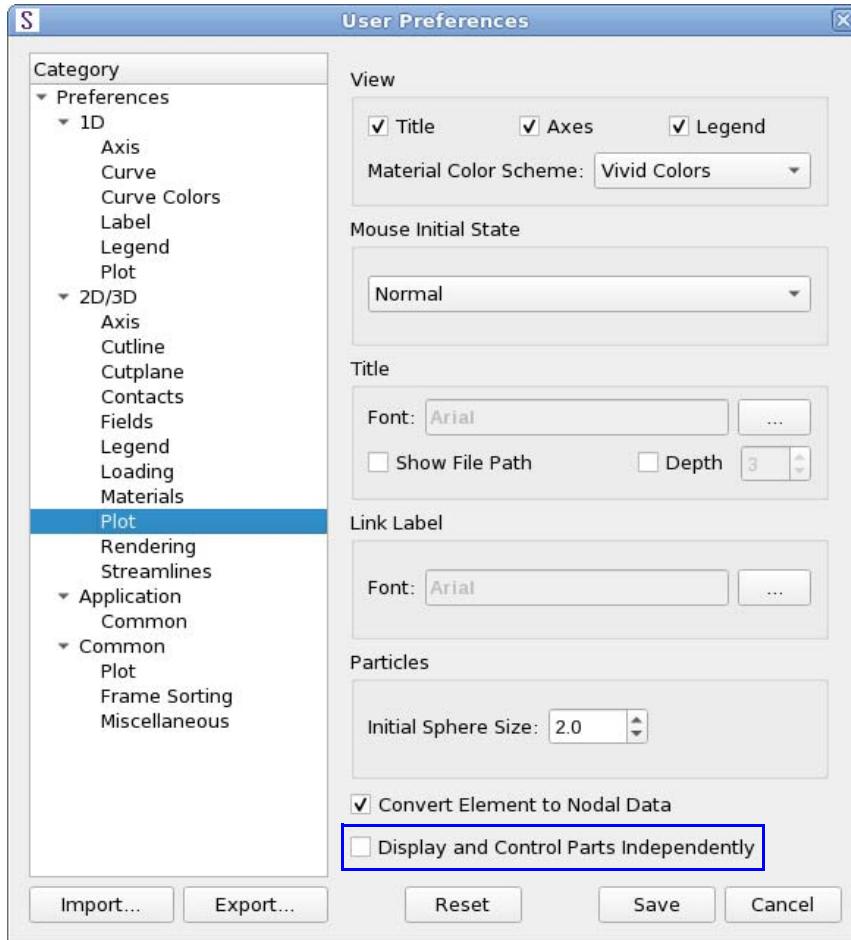
When you load a TDR file from the command line, Sentaurus Workbench, or the user interface using the File Open dialog box, you can select an option to display parts individually. However, if you load a TDR file using Sentaurus Visual Tcl commands, you can display parts individually using the `-parts` option (see [load_file on page 318](#) and [load_file_datasets on page 319](#)).

To visualize regions with multiple parts:

1. Choose **Edit > Preferences**.
2. In the User Preferences dialog box, expand **2D/3D > Plot**.
3. Select **Display and Control Parts Independently**.

Chapter 4: Working With 2D and 3D Plots

Visualizing 2D and 3D Plots



4. Click **Save**.

Note:

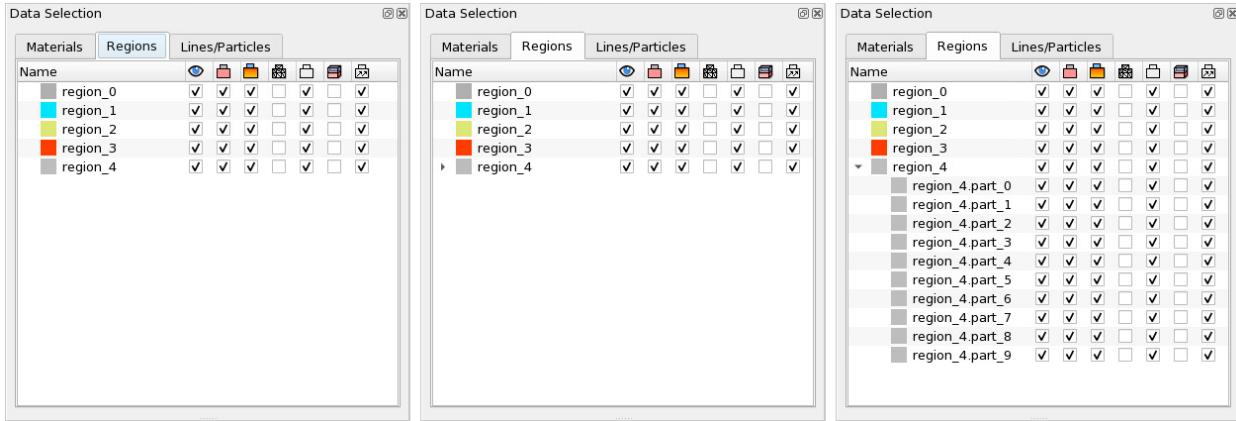
If a TDR file is loaded with this option selected, field data is not loaded for regions with multiple parts.

The **Regions** tab and the **Lines/Particles** tab of the Data Selection panel show the hierarchy of regions and parts. All parts are shown under the parent region. Therefore, if you modify a property of the parent region (such as visibility), all its parts are also modified (see [Figure 62 on page 117](#)).

Chapter 4: Working With 2D and 3D Plots

Visualizing 2D and 3D Plots

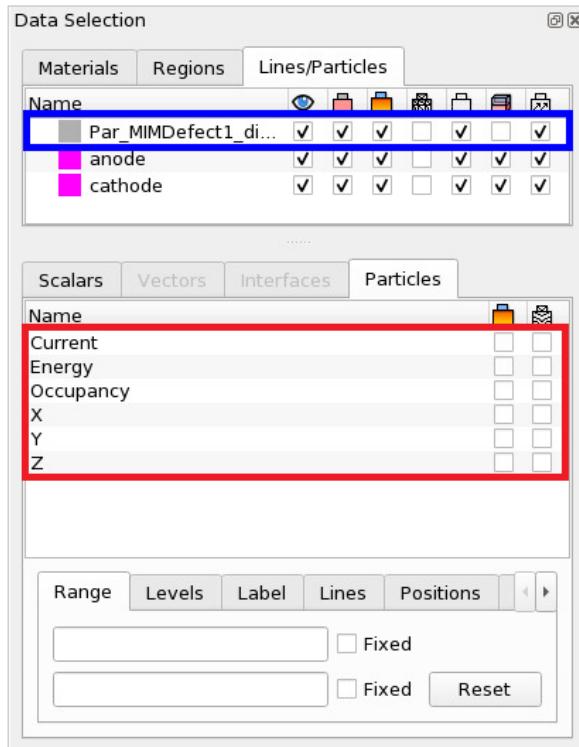
Figure 62 Regions tab showing (left) no regions with parts, (middle) one region (region_4) with parts, and (right) region with parts expanded



Visualizing Discrete Traps

You can visualize discrete traps as particles and visualize scalar fields as the color of the particles. Scalar fields and particle fields are on separate tabs (see Figure 63).

Figure 63 Regions of particles (blue box) and fields of particles (red box)



Chapter 4: Working With 2D and 3D Plots

Visualizing 2D and 3D Plots

Discrete traps are displayed as spheres of an initial size defined in the User Preferences dialog box. However, you can change the size by using the Region Properties dialog box.

To change the initial size of spheres:

1. Choose **Edit > Preferences**.
2. In the User Preferences dialog box, expand **2D/3D > Plot**.
3. Under Particles, from the **Initial Sphere Size** list, select the value.
4. Click **Save**.

To change the size of spheres in a plot:

1. Select the plot.
2. Choose **Data > Region Properties**, or click the  toolbar button.

The Region Properties dialog box opens.

3. Select **Lines/Particles**.



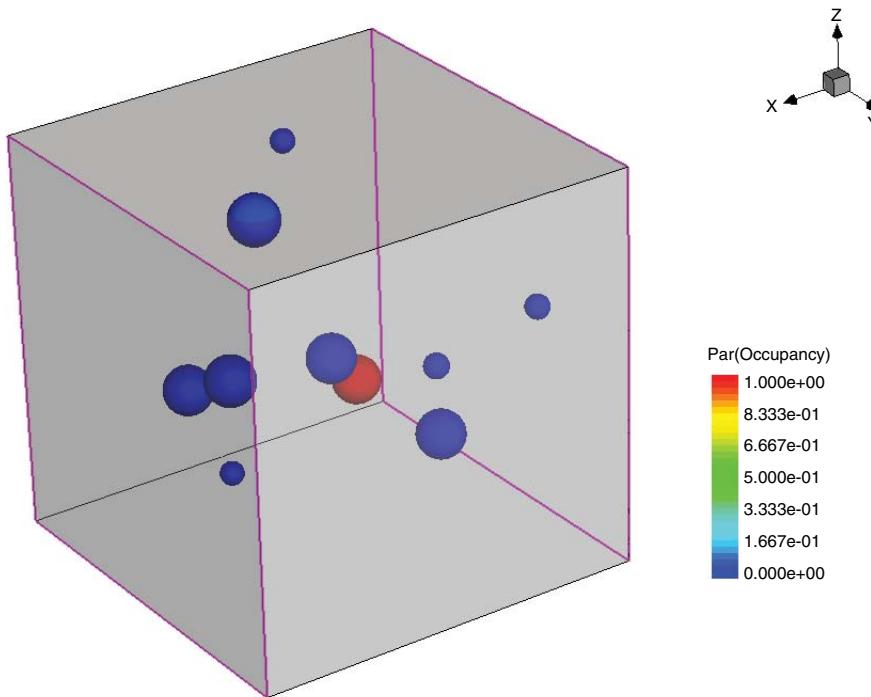
4. Under the Particle Size column, change the values as required.
5. Click **Close**.

[Figure 64](#) illustrates changing the sphere size of discrete traps.

Chapter 4: Working With 2D and 3D Plots

Visualizing 2D and 3D Plots

Figure 64 Visualizing the scalar field Occupancy



3D View

The 3D view is described by the position and the orientation of the camera in the world coordinate system.

The orientation of the camera is described by the vector formed between its position and focal point. This vector is called the *direction of propagation*. Initially, the center of the structure is located at the focal point.

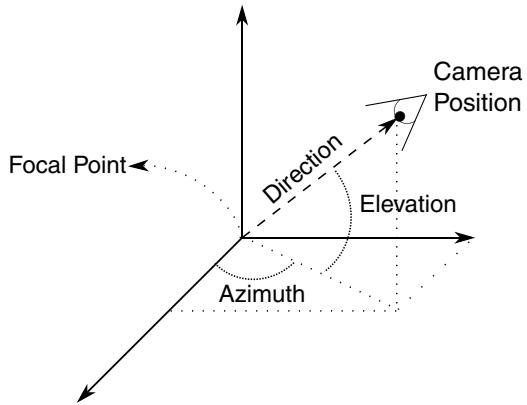
The position can be described in a spherical coordinate system by the distance between the center of the camera and the focal point, also known as the *depth distance*, and two angles (azimuth and elevation).

The camera has its own coordinate system that is defined by three vectors: the view up, the direction of propagation, and the horizontal vector.

Chapter 4: Working With 2D and 3D Plots

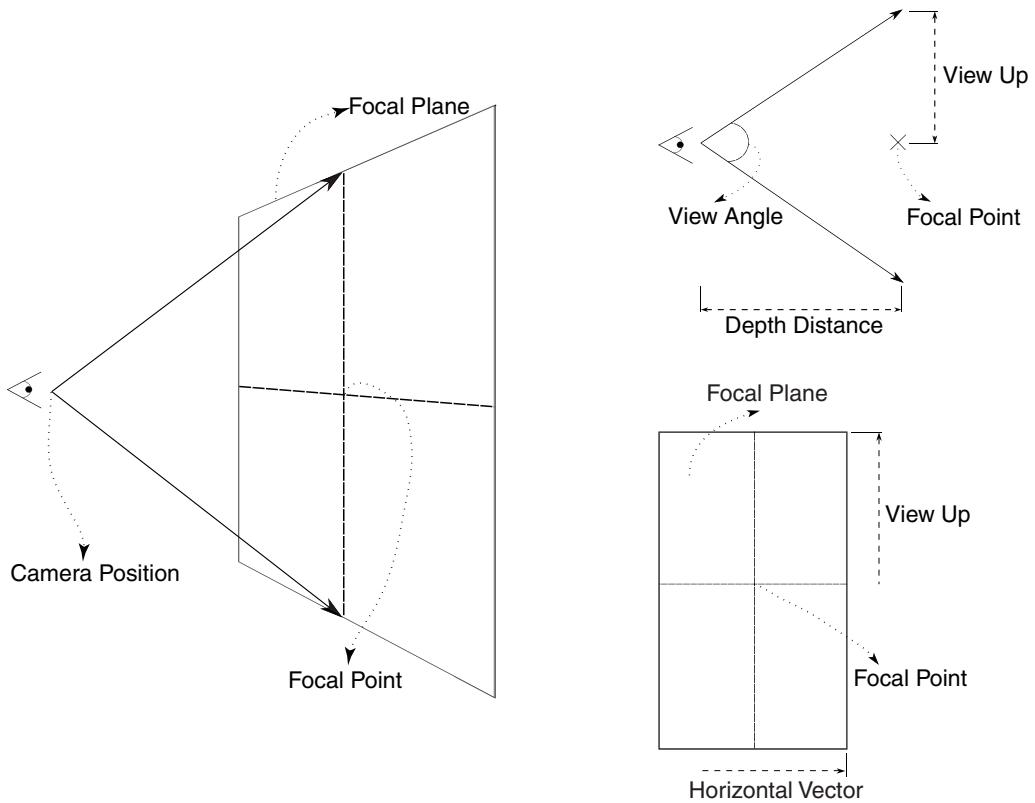
Visualizing 2D and 3D Plots

Figure 65 Description of parameters used to define camera position



The view is defined by the view angle of the camera and the location of the focal plane, which is the plane defined by the focal point and the view up vector. The projection of this plane in the screen will be the view observed. [Figure 66](#) shows the variables that describe the camera and the final view given by these variables.

Figure 66 Camera properties: (left) perspective view, (upper right) horizontal view, and (lower right) vertical view



Interacting With 3D Plots

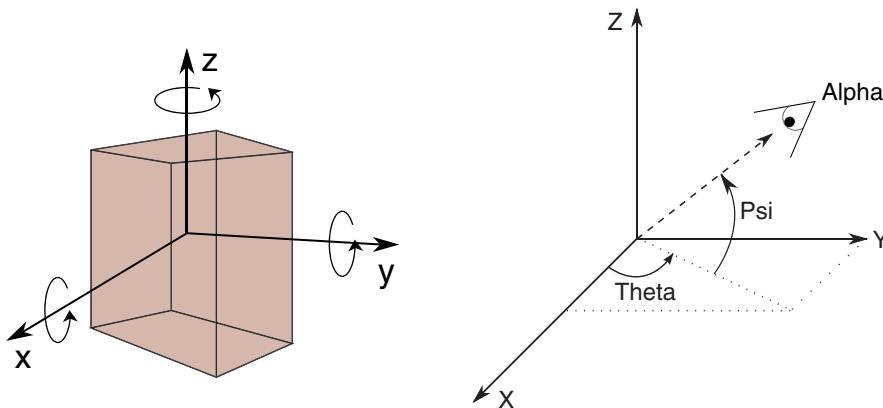
While a 3D plot is active, you can interact directly with the camera.

Using the Select/Rotate tool  and dragging, you can rotate the view in relation to the rotation center using the default rotation mode.

Using the Spherical Rotation tool  and dragging, you can perform a spherical rotation of the view. Originally, the rotation center is the same as the focal center and can be changed by pressing the O key (lowercase character). The rotation transformation changes the azimuth and the elevation angles (taking the rotation center as reference), thereby maintaining the distance from the origin constant. In addition, you can rotate the plot more accurately in each angle by choosing **View > Rotate** or using the `rotate_plot` command. You can specify the use of either the x-, y-, and z-coordinates, or the psi, theta, and alpha spherical coordinates (see [rotate_plot on page 336](#)).

In the same way, by right-clicking, you can change the position of the camera and its focal point in the plane perpendicular to the direction of propagation, keeping the direction of propagation and the depth distance constant.

Figure 67 (Left) Rotation of axes in relation to rotation center and (right) spherical rotation



You can also zoom in to the plot, changing the depth distance using the mouse wheel.

Furthermore, with the `zoom_plot` command, you can specify a factor to change the view angle of the camera (see [zoom_plot on page 386](#)). The factor given to the command is multiplied by the *view up* vector (the reflection of the view angle in the focal plane), resulting in zooming in when the factor is greater than 1 and zooming out when it is less than 1.

In addition, you can change the position of the camera and its focal point, in the world coordinate system, in the Camera Properties panel (choose **View > Camera Configuration**). See [Editing Camera Properties on page 122](#).

Chapter 4: Working With 2D and 3D Plots

Visualizing 2D and 3D Plots

You can change the projection mode to a parallel projection, rather than a perspective projection, by any of the following methods:

- Set the `-parallel` option of the `set_camera_prop` command (see [set_camera_prop on page 344](#)).
- Choose **View > Camera Configuration**. In the Camera Properties panel, click the **More** tab. Under Projection, select **Parallel** (see [Figure 68 on page 123](#)).
- Choose **Edit > Preferences**. In the User Preferences dialog box, expand **2D/3D > Rendering**. Under Projection, select **Parallel**. Click **Save**.

The term *parallel projection* means that the lines of sight from an object to the projection plane are parallel to each other. Lines that are parallel in 3D space remain parallel in the 2D projected image. In addition, parallel projection corresponds to a perspective projection with an infinite focal length, that is, the distance from the lens of a camera to the focal point.

Editing Camera Properties

When visualizing 3D structures, you can query and edit camera properties in the Camera Properties panel. To open the panel, choose **View > Camera Configuration**.

The tabs of the Camera Properties panel are as follows:

- The **Position** tab allows you to modify the position and the focal point of the camera.
- The **Rotation Point** tab allows you to set the properties of the rotation point (see [Rotation Point on page 146](#)).
- The **More** tab allows you to configure the view up and the view angle. You can also select the type of projection as well as whether to use the dolly zoom.

The **Perspective** projection is the default and helps to visualize objects as they are seen in real life: near objects look bigger than distant objects. When using **Parallel** projection, the sizes of objects are proportional to their real sizes, so distance does not affect the final visualization.

The **Dolly Zoom** option is available only for perspective projection. By default, this option is not selected, which means zoom operations are performed by modifying the view angle. When you select this option, the camera moves towards and away from the focal point of an object.

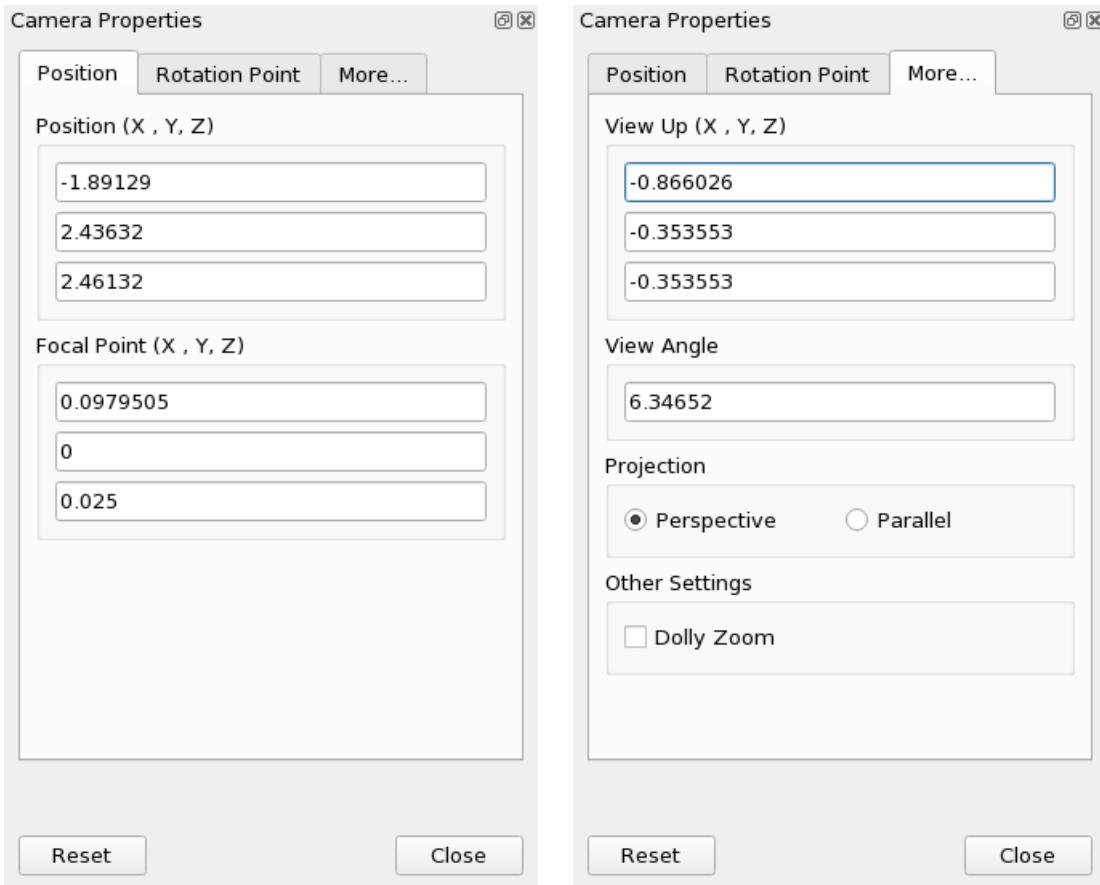
Note:

When you select the **Dolly Zoom** option, the camera might move inside visualized objects.

Chapter 4: Working With 2D and 3D Plots

Visualizing 2D and 3D Plots

Figure 68 Camera Properties panel: (left) Position tab and (right) More tab



2D View

The visualization of a 2D plot is described by the same camera values as the 3D visualization, but the difference is that, in a 2D view, the direction of propagation never changes, thereby maintaining constant azimuth and elevation angles.

Interacting With 2D Plots

Dragging across a plot changes the position of the camera and its focal point in the plane perpendicular to the direction of propagation, which is the same plane for 2D plots.

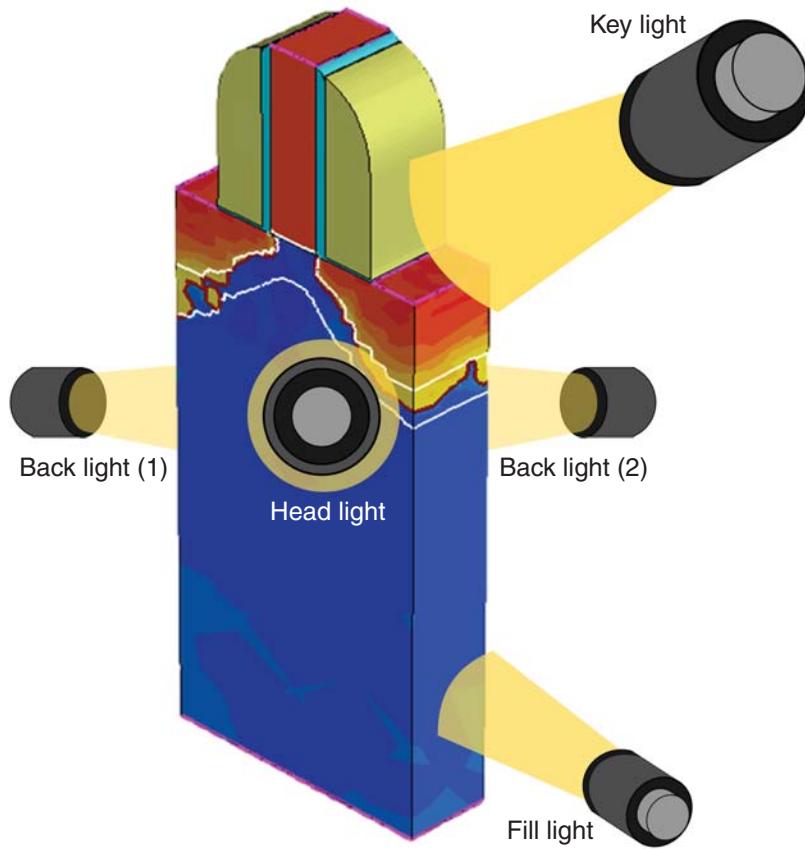
Using the mouse wheel changes the depth distance from the focal point, zooming in to and zooming out of the plot.

Light Kit

To provide clear visualization of 3D structures, Sentaurus Visual uses a light kit instead of a single light source. The light kit consists of the following different light sources (see [Figure 69](#)):

- A *key light*, which is the main light source, represents the sun or a ceiling light. It is located in front of the displayed structure, above the camera, and displaced slightly to the right.
- A *fill light*, which represents ambient light, is located in front of the displayed structure and under the key light.
- A *head light* is located in the same place as the camera.
- Two *back lights*, located behind the displayed structure, are evenly spaced from the vertical center.

Figure 69 Light kit illustrating different light sources



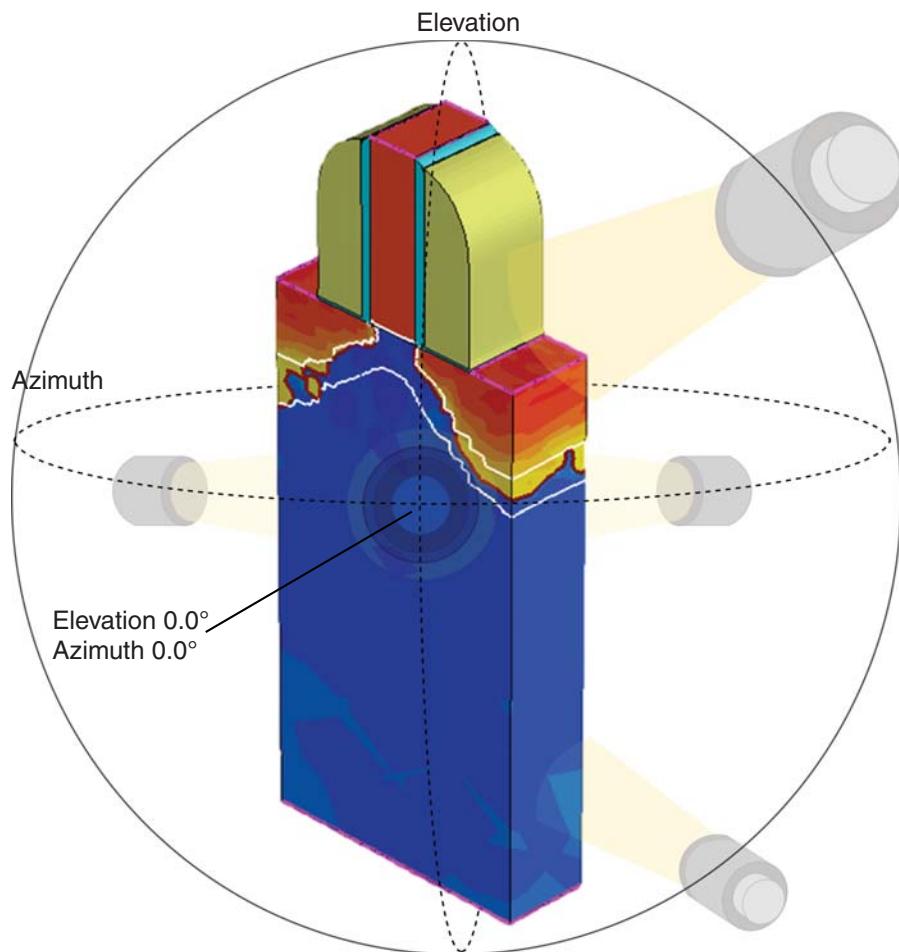
Chapter 4: Working With 2D and 3D Plots

Visualizing 2D and 3D Plots

The light kit is designed so that the key light is the brightest reference for all the other light sources, whose intensity can be adjusted directly. For the other light sources (head, fill, and back lights), the **Key Ratio** parameter can be adjusted, which represents the relative intensity of the key light to each of the other lights (default ratio is 1:3).

The position of the key, fill, and back light sources can be adjusted by changing the elevation and azimuth (latitude and longitude) with respect to the origin of the coordinate system. Due to its nature, the head light is always located at zero elevation and azimuth.

Figure 70 Elevation and azimuth around the structure

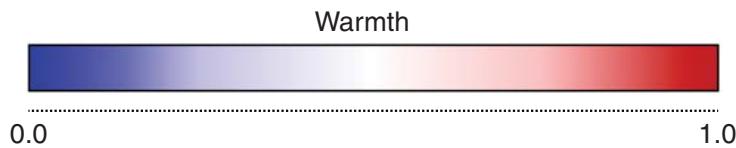


Besides position, the warmth of the light sources can be adjusted from 0.0 to 1.0, where 0.0 represents a *cold* blue-tinted light and 1.0 represents a *warm* red-tinted light. Values around 0.5 provide white light.

Chapter 4: Working With 2D and 3D Plots

Visualizing 2D and 3D Plots

Figure 71 Warmth range



Editing the Properties of Lights

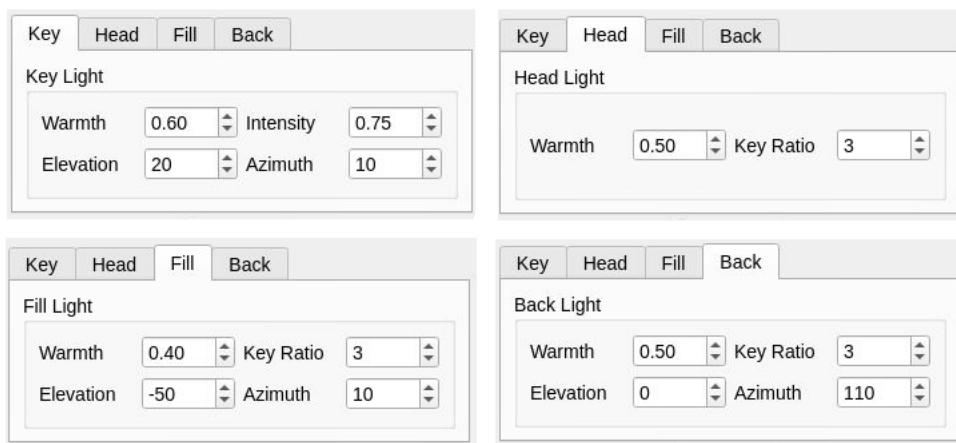
You can edit the properties of the different light sources in the Lights Properties panel. To open the panel, choose **View > Lights Configuration**.

The tabs of the Lights Properties panel (see Figure 72) are each linked to one of the light sources of the light kit. You can adjust the following parameters:

- **Warmth [0.0, 1.0]**: Color temperature
- **Intensity [0.0, 1.0]**: Intensity of the key light
- **Key Ratio [0, 99]**: Relative intensity of the key light to the head, fill, and back lights (default ratio is 1:3)
- **Elevation [-180°, 180°]**: Latitude of the light source
- **Azimuth [-180°, 180°]**: Longitude of the light source

The **Reset** button is used to reset the parameters of all lights to their default values.

Figure 72 Tabs of Lights Properties panel



Loading Options for Data

Sentaurus Visual provides options that allow you to omit some data from loading (only for 2D and 3D TDR files), to reduce the loading time and memory consumption:

- You can clear the **Load Interfaces** option to not load interfaces. Some TDR files can have a large amount of data about interfaces, and not loading this data reduces the loading time. By default, Sentaurus Visual loads interfaces. If you clear this option, the data fields located on interfaces are also not read.
- The **Load Selected Fields** option allows you to load only certain fields. Some simulations generate a long list of fields, but you might want to view or analyze only a few fields. This option reduces the loading time and memory consumption. You can load only a selected list of field names. The filtering works with wildcards (such as * and ?), making selection of the list simpler. For example, by adding the element **Net***, Sentaurus Visual will load the NetActive and NetDoping fields, if they exist. By default, Sentaurus Visual loads all fields.
- The **On Demand** option allows you to load fields only when needed. This option greatly reduces loading time and memory consumption. Fields that are not loaded are shown in italics in the Data Selection panel. The field is loaded when you change the rendering option of that field (see [Rendering Options on page 129](#)).

The **In Background** option allows fields to be loaded automatically in the background after the TDR file has been opened. You can select this option only if the **On Demand** option is selected.

Note:

These loading options work only with TDR files version 17. This version implements a new optimized data and file format, which improves the Sentaurus Visual loading time significantly. In addition to loading on demand, data is read using multiple threads.

- In the Sentaurus Visual configuration file (~/.config/Synopsys/SVisual.conf), you can prevent some fields from being loaded and shown at interfaces, while still allowing them in bulk regions. To configure this option, edit the `interfaces\blockSpecificFields` entry in the `PlotHD` group and list the fields to be blocked. For example:

```
interfaces\blockSpecificFields=Arsenic, As3, Boron, BTot
```

The loading options work as set up in the User Preferences dialog box, unless you execute Sentaurus Visual with the `-alldata` command-line option to overwrite the settings. When you use `-alldata`, the `-alldata` option is added to the Tcl commands related to loading and reloading files, so that you can set the condition explicitly if needed. However, the `-alldata` option of these commands always overwrites the user preference settings,

Chapter 4: Working With 2D and 3D Plots

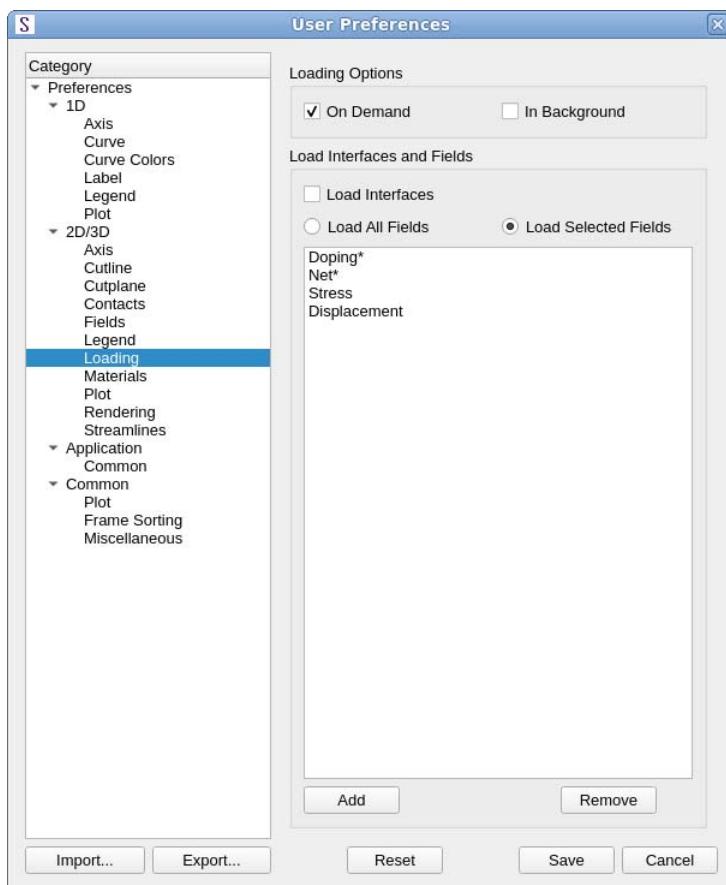
Loading Options for Data

meaning all data is loaded. See [load_file](#) on page 318, [load_file_datasets](#) on page 319, [reload_datasets](#) on page 326, and [reload_files](#) on page 326.

To change loading options for data:

1. Choose **Edit > Preferences**.

The User Preferences dialog box opens.



2. Expand **2D/3D > Loading**.

The **Load Interfaces** option is selected by default.

3. Select **Load Selected Fields** if required.

4. Perform any of the following operations:

- To remove a field, select a field and click **Remove**.

The list is updated without the field.

- To modify a field, double-click a field, change its name, and press the Enter key.

Chapter 4: Working With 2D and 3D Plots

Rendering Options

- To add a field, click **Add**.

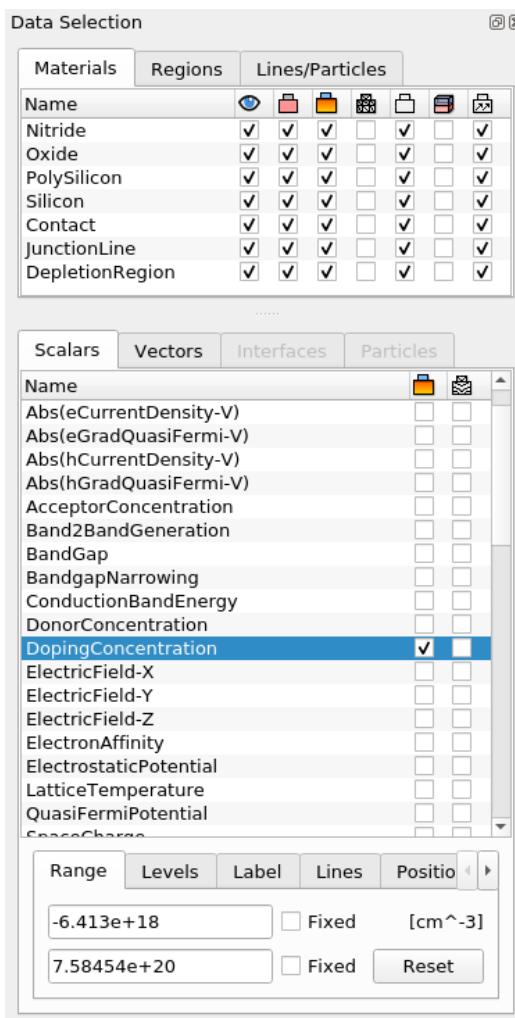
A new field name is added to the end of the list. Double-click the field, change its name, and press the Enter key.

5. Click **Save**.

Rendering Options

Two-dimensional or 3D plots are composed of materials that are distributed in regions with properties defined in contour maps (scalars, interfaces, and particles) or flux lines (vectors). All these properties can be found on the Data Selection panel (see [Figure 73](#)).

Figure 73 Data Selection panel showing materials and scalar properties



Materials and Regions

The materials and regions of which a plot is composed are shown in the upper part of [Figure 73](#).

Note:

Clicking the **Name** column heading of the Data Selection panel sorts the elements, by cycling through unsorted, then alphabetical ascending order, and then alphabetical descending order.

Double-clicking a cell of a structure in the plot area highlights the region or material to which that cell belongs in the Data Selection panel.

Table 7 Icons relevant for materials and regions

Icon	Description
	Shows or hides the material or region completely. If deactivated, it hides the bulk, contour fields, mesh, borders, and vector fields independent of their state.
	Shows or hides the bulk.
	Shows or hides the contour fields.
	Shows or hides the mesh.
	Shows or hides the borders.
	Switches translucency on or off.
	Shows or hides the vector fields.

Showing or Hiding Properties for Multiple Materials and Regions

Clicking a check box next to a material or region shows or hides that specific property only for that region or material.

Note:

You can select multiple rows of materials or regions by dragging the cursor, or holding the Ctrl key or the Shift key while selecting rows in the Data Selection panel.

Chapter 4: Working With 2D and 3D Plots

Rendering Options

If you select multiple rows of materials or regions, when you click an icon itself, it shows or hides that specific property only for all the selected materials or regions.

If no materials or regions are selected, clicking an icon affects all materials or regions. These operations are immediately shown in the plot area.

Modifying Properties in Multiple Materials and Regions

Sentaurus Visual provides a dialog box where you can modify all properties in several regions, particles, or materials at the same time (see [Figure 74](#)).

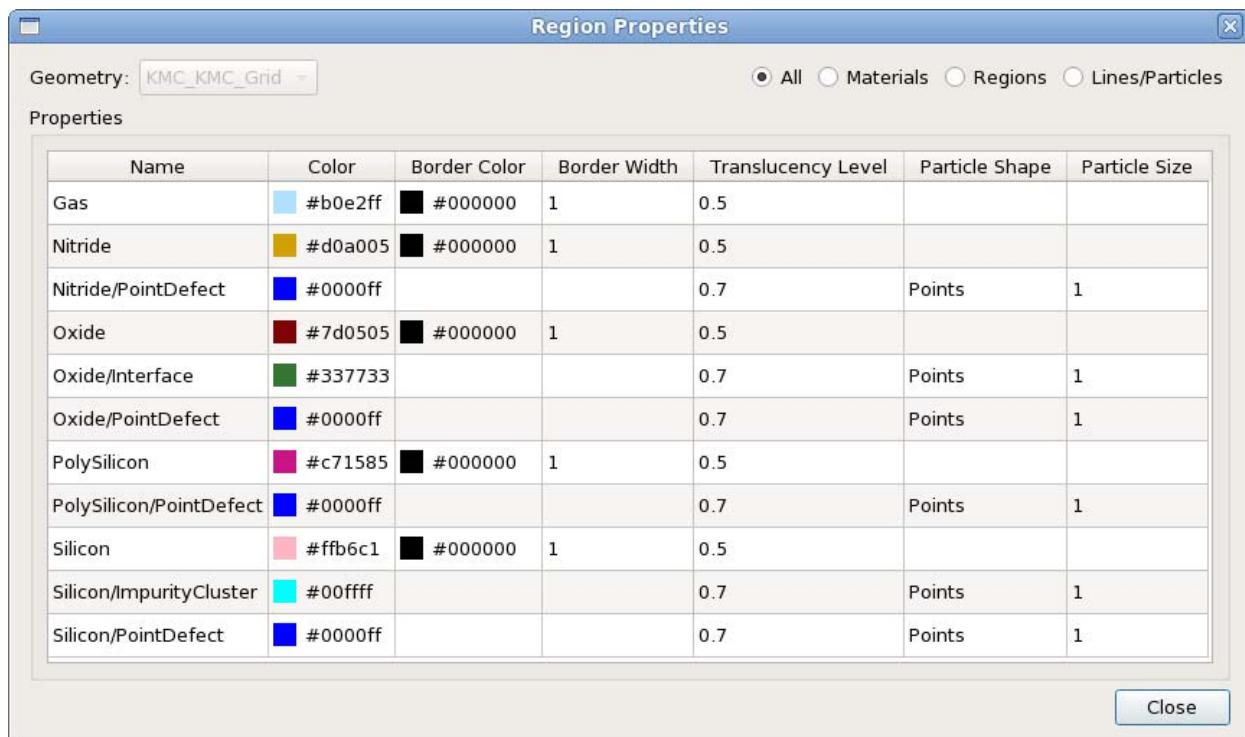
To modify multiple regions:

1. Choose **Data > Region Properties**, or click the  toolbar button.
2. Select the required rows.
3. Click the column header of the property you want to change.

A dialog box is displayed where you enter the value of the property.

However, if only one region needs to be modified, double-click the entry and type the new value to change the property.

Figure 74 Region Properties dialog box



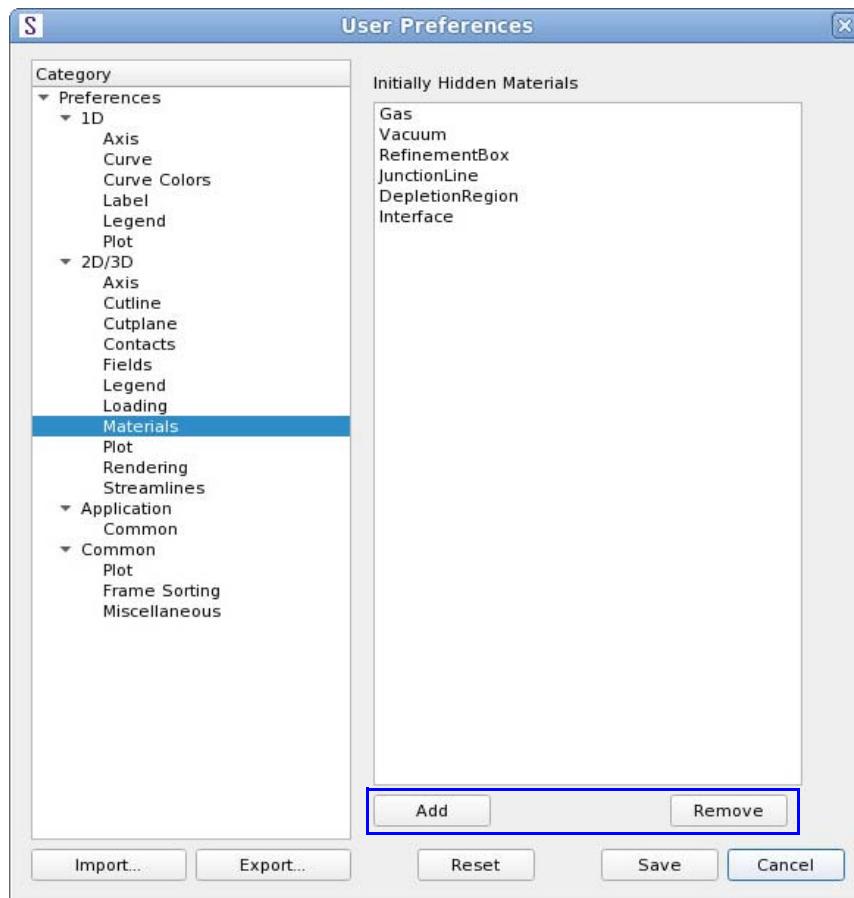
Modifying the List of Initially Hidden Materials

Sentaurus Visual manages a list of materials that are not rendered when you open a new file. The following materials are not rendered by default:

- Gas
- Vacuum
- RefinementBox
- JunctionLine
- DepletionRegion
- Interface

To modify the list of initially hidden materials:

1. Choose **Edit > Preferences**.
2. In the User Preferences dialog box, expand **2D/3D > Materials**.



Chapter 4: Working With 2D and 3D Plots

Rendering Options

3. Perform any of the following operations:
 - To remove a material, select a material and click **Remove**.
 - The list is updated without the material.
 - To modify a material, double-click a material, change its name, and press the Enter key.
 - To add a material, click **Add**. A new material name is added to the end of the list. Double-click the material, change its name, and press the Enter key.
4. Click **Save**.

Contact Regions

The contact material and its regions can be colored in a special way defined in the User Preferences dialog box. This feature allows you to differentiate the contacts for better understanding of the types of region. Any change to the user preferences is applied to the next created plots. Nevertheless, the changes can be applied to the current plot using the **Contacts** tab of the Plot Properties panel.

In the User Preferences dialog box, expand **2D/3D > Contacts**. In the **Contact Color Behavior** box, select from one of the options:

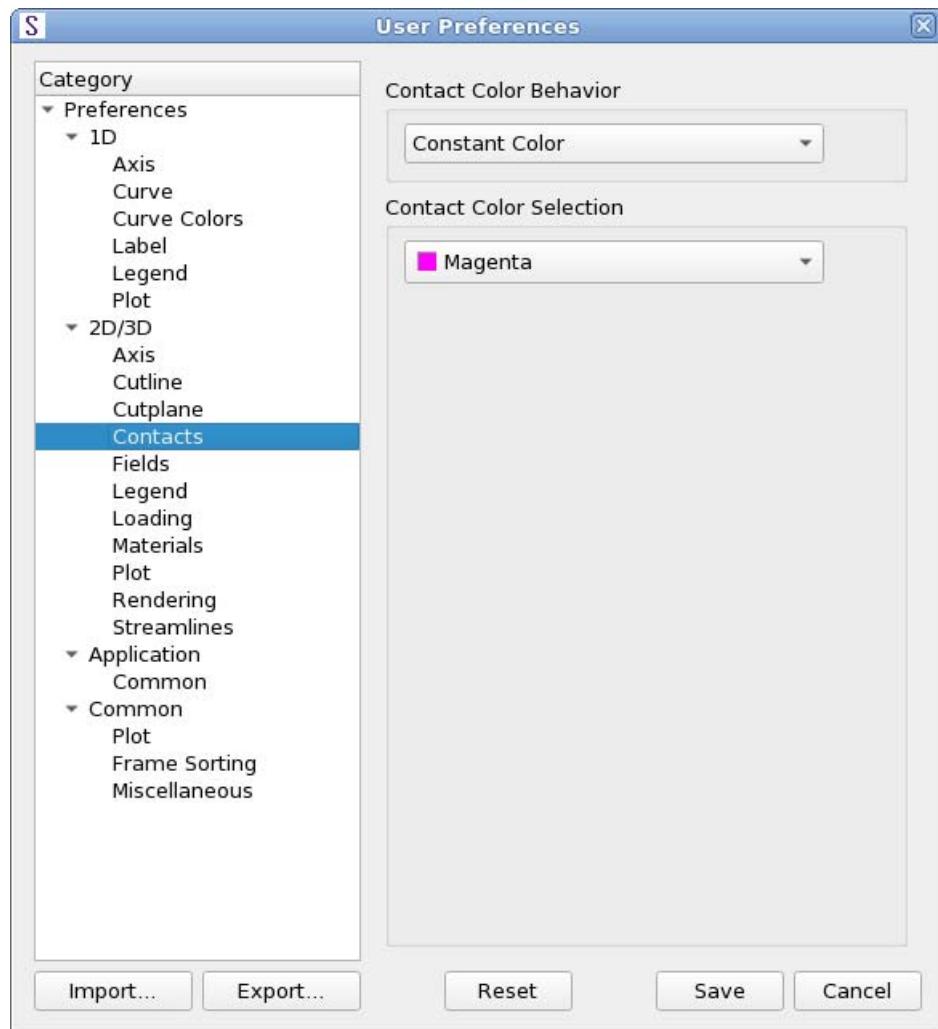
- Constant Color
- List Colors
- Map Colors

The **Constant Color** option loads all the contacts with the same color as a configurable default value. Magenta is the default.

Chapter 4: Working With 2D and 3D Plots

Rendering Options

Figure 75 User Preferences dialog box showing selection of Constant Color

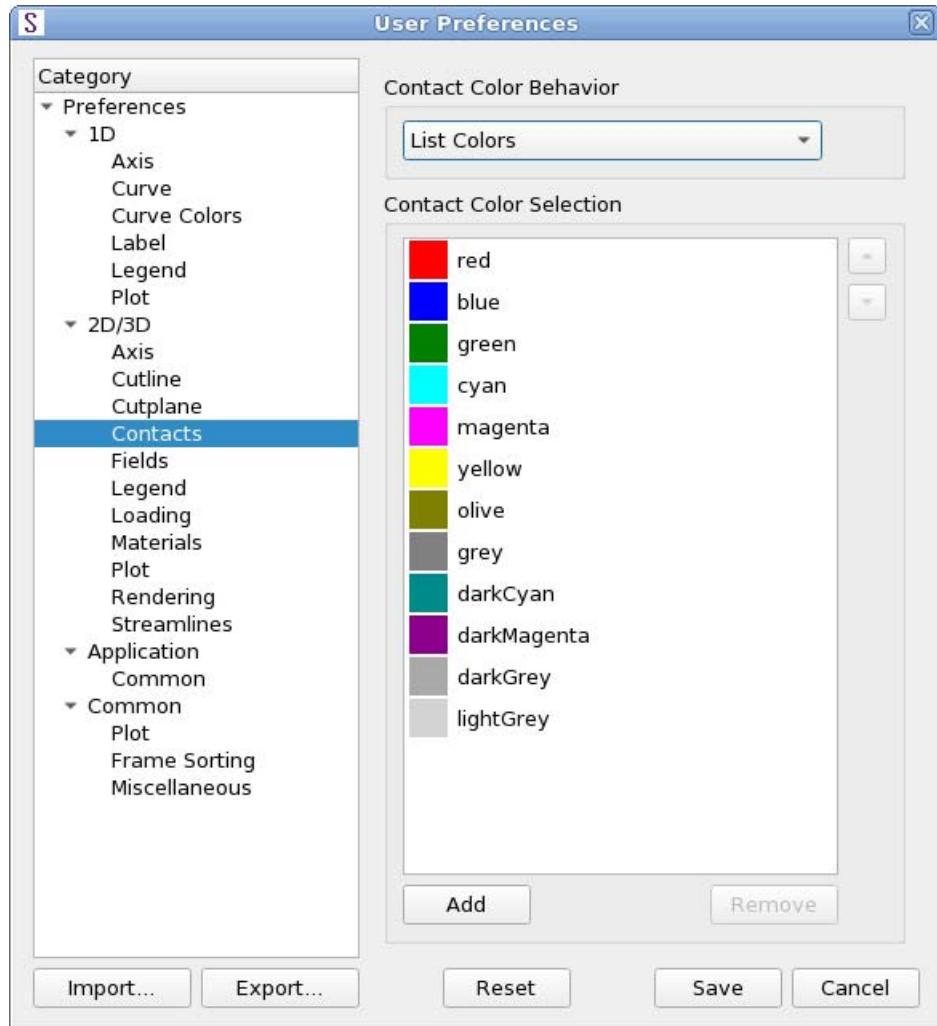


Chapter 4: Working With 2D and 3D Plots

Rendering Options

The **List Colors** option loads the contacts with a set of colors using round robin logic.

Figure 76 User Preferences dialog box showing selection of List Colors



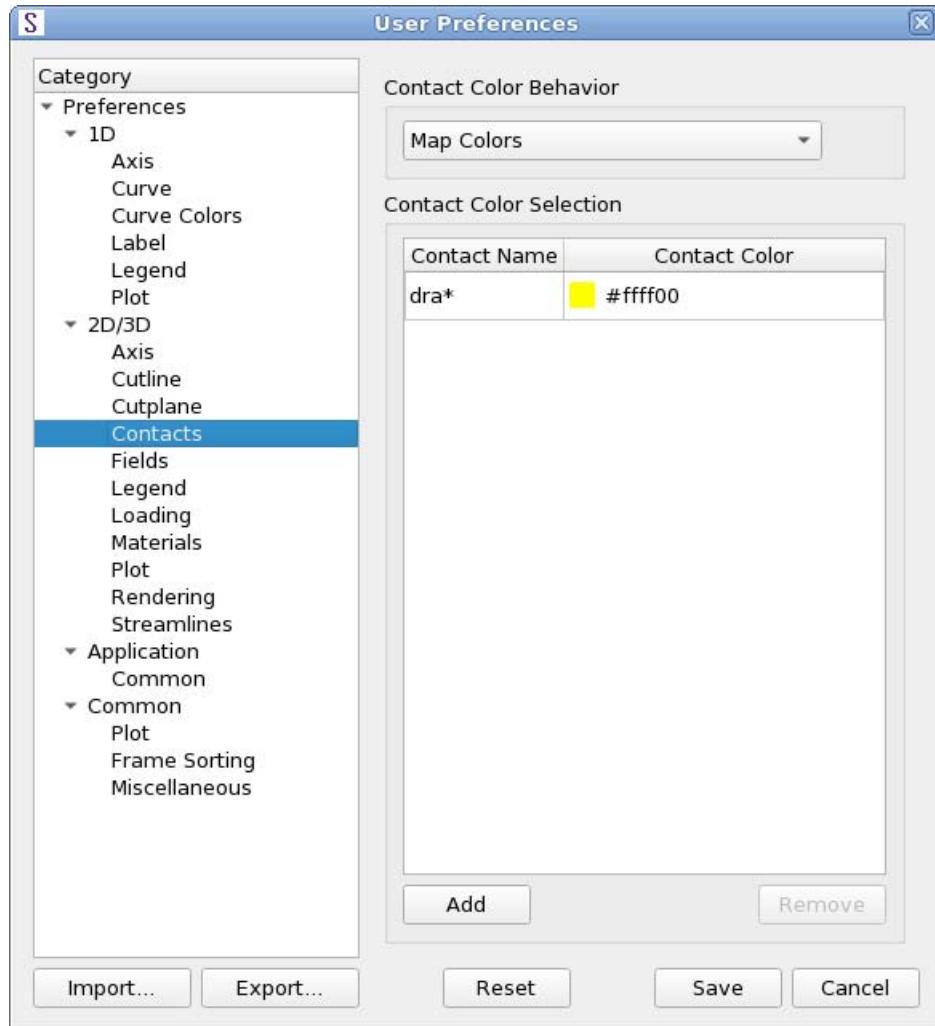
The **Map Colors** option loads the specified contacts with the specified colors; otherwise, they are displayed with a constant color. The Contact Name column supports wildcards for reference contacts with common patterns in their names. This list is empty by default.

To apply changes to the user preferences without reloading a plot, use the **Contacts** tab of the Plot Properties panel to select the color behavior and to apply changes (see [Figure 48 on page 102](#)).

Chapter 4: Working With 2D and 3D Plots

Rendering Options

Figure 77 User Preferences dialog box showing selection of Map Colors



Contour Plots

Scalar fields are used to generate contour plots. Usually, the contour levels are calculated automatically, so that they are distributed evenly within the value range of the active field.

Contour Legend Settings

The properties of the legend of contour plots can be changed by double-clicking the legend.

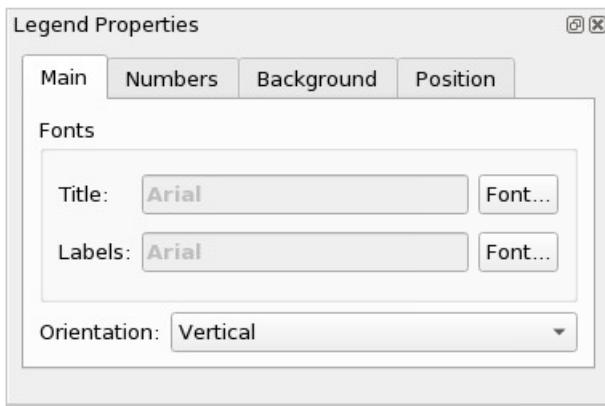
Chapter 4: Working With 2D and 3D Plots

Rendering Options

The Legend Properties panel opens (see [Figure 78](#)) where you can:

- Customize the number, precision, and notation of the labels.
- Enable a background for the legend.
- Change the background color and the frame color.
- Customize the font for the title and the labels.
- Set the orientation of the legend.

Figure 78 Legend Properties panel



The font size of the legend is related to the diagonal of the plot, and Sentaurus Visual internally sets a value so that the legend is visible in plots that are 600×600 pixels or larger.

To change this value, you must apply a font scale diagonal factor to the font base every time that you rescale a plot. You can do this with the `-title_font_factor` and `-label_font_factor` arguments of the `set_legend_prop` command to change the title and labels of the legend, respectively (see [set_legend_prop on page 356](#)).

In addition, you can set this factor in the User Preferences dialog box.

To set a new font scale diagonal factor:

1. Choose **Edit > Preferences**.
2. In the User Preferences dialog box, expand **2D/3D > Legend**.
3. Under Fonts, click the button next to the **Title Font** or **Label Font** field.
4. In the Font dialog box, set a font scale diagonal factor.

Values greater than 1.0 will increase the font size, and values less than 1.0 but greater than zero will decrease it.

Chapter 4: Working With 2D and 3D Plots

Rendering Options

5. Click **OK** to close the Font dialog box.
6. Click **Save**.

Displaying Contour Plots

To create a contour plot, select the required property to be plotted on the **Scalars** tab of the Data Selection panel (see [Figure 73 on page 129](#)). The range and levels are set automatically, but they can be customized using the **Range** and **Levels** tabs (see [Figure 79](#)), where you can manually define a range and the number of levels displayed.

Figure 79 Contour plot options showing the Range tab where the first field is the minimum value and the second field is the maximum value of the range



On the **Lines** tab, you can change the properties of the contour lines of the selected field, such as color and width, and then show several contour lines at the same time.

When a file is not a TDR file, all the default values for field scaling and field units are defined in the `datexcodes.txt` file for each field (see *Utilities User Guide*, Variables).

In the case of a TDR file, all field units shown are those contained in the TDR file. However, for field scaling, the values are obtained from the `datexcodes.txt` file.

Despite the input file format, if a field is defined (in the `datexcodes.txt` file) to be present in only one type of material (for example, semiconductor), no data is loaded or displayed for regions of a different material type.

Although only one field can be displayed using color-filled contour levels, you can display multiple contour lines from other fields by clicking the second column of the field list of the Data Selection panel (see [Figure 73](#)).

If a field unit has a question mark (?), it means the unit is undefined and, most likely, comes from an old file that you must update.

Converting Data to Nodal

For element-type data, there is an option that interpolates element data to nodal data.

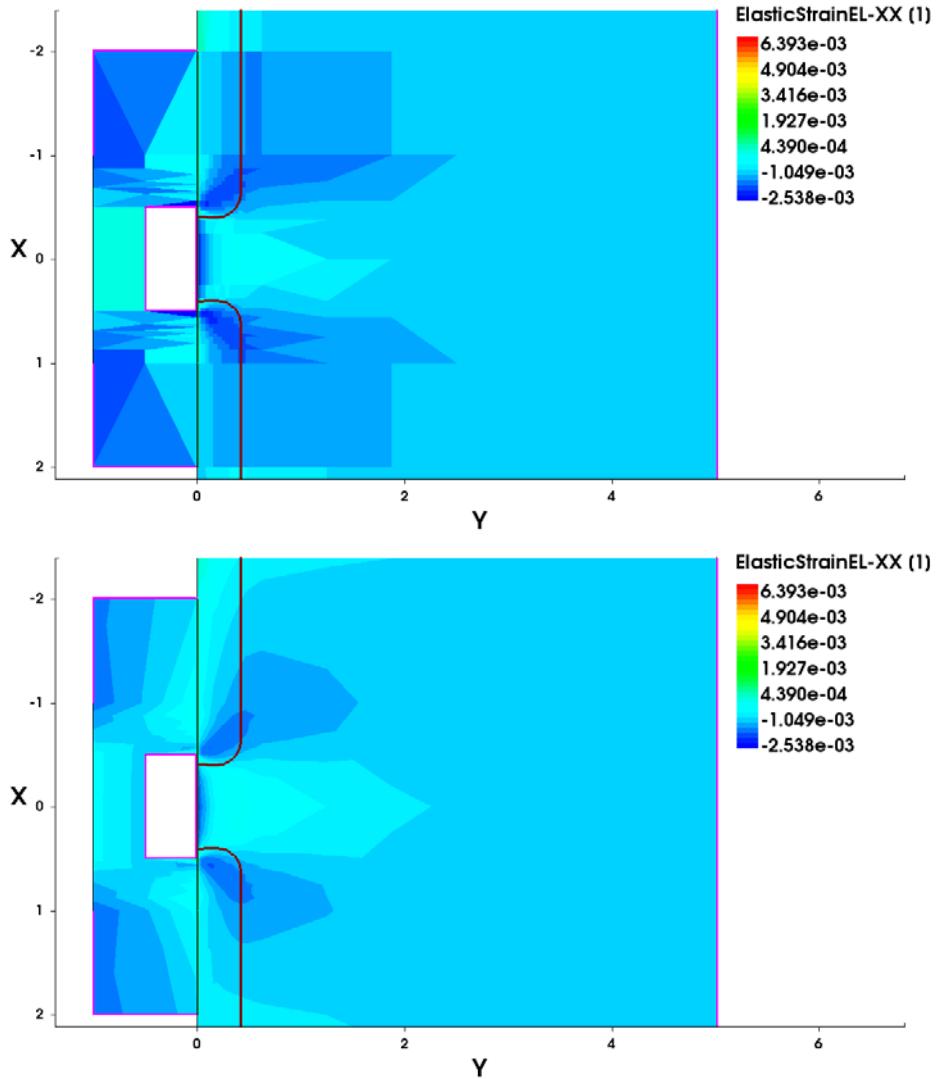
To convert data, select the **Convert to Nodal** option on the **Levels** tab of the Data Selection panel. [Figure 80](#) shows the results of converting data to nodal. The converted data looks smoother.

Chapter 4: Working With 2D and 3D Plots

Rendering Options

In addition, in the User Preferences dialog box (expand **2D/3D > Plot**), the **Convert Element To Nodal Data** option is selected by default.

Figure 80 Comparison of (top) element-type data and (bottom) nodal-type data



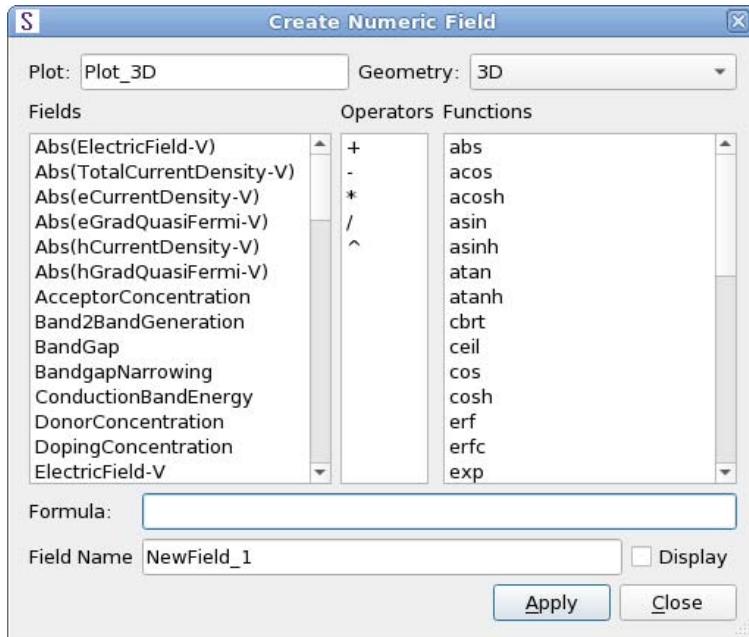
Creating New Scalar Fields

Custom scalar fields can be created on the **More** tab (see [Figure 79 on page 138](#)). When you click the **Add Field** button on this tab, a dialog box is displayed where you can create a custom field (see [Figure 81](#)). It allows insertion of functions and operators, and the use of existing fields.

Chapter 4: Working With 2D and 3D Plots

Rendering Options

Figure 81 Create Numeric Field dialog box displaying fields, operators, and functions



Vector Plots

To add a vector field to a plot, click the **Vectors** tab of the Data Selection panel. Select a check box next to a field to display it on the plot. Vector lines can be displayed uniformly or with a size proportional to the magnitude of the field.

Uniform scaling means that all vectors have the same size. The length of the arrow is the value of the scaling.

With the grid scaling option, the length of the arrow is given by the vector field magnitude (or the absolute value if required) multiplied by the scaling factor.

The default uniform value can be set in the User Preferences dialog box (expand **2D/3D > Fields**). In the Vector group box, if no value is set, Sentaurus Visual uses 0.1 as the default.

Figure 82 (Left) Scaling options for vectors and (right) Head tab showing property options for arrowheads



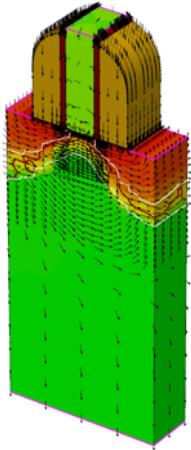
Chapter 4: Working With 2D and 3D Plots

Importing an Image as a Background Field

The **Head** tab lists the properties of the arrowhead such as shape, size, angle, and color (only available when the shape selected is **Arrow Solid** or **Head Solid**).

The **Constant** option maintains the size of the arrowhead regardless of the zoom level or the vector length.

Figure 83 Example of plotting a vector field



Importing an Image as a Background Field

You can load an image file into Sentaurus Visual and use it as a background field.

To load an image:

1. Choose **File > Import Image**.

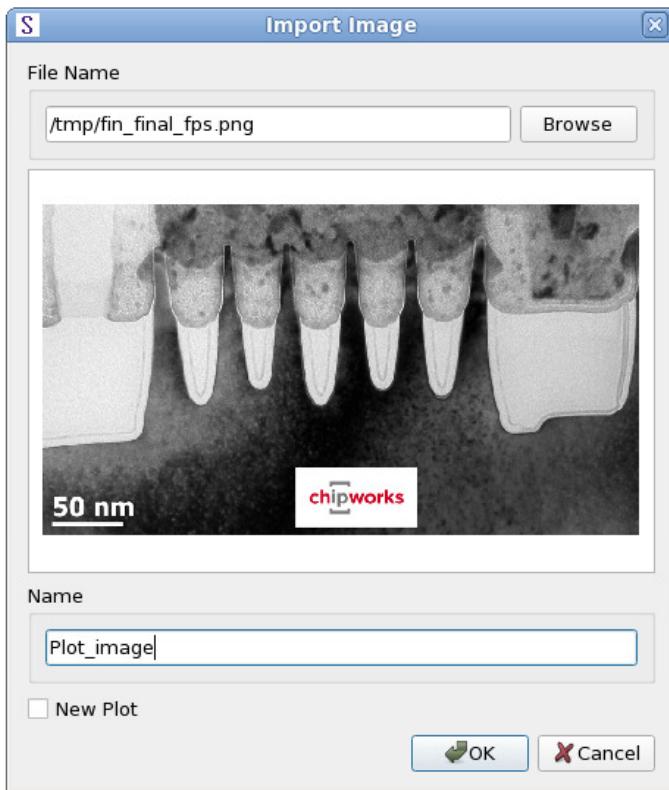
The Import Image dialog box opens.

2. Click **Browse** and select the image, or enter the path to the image in the **File Name** field.

Chapter 4: Working With 2D and 3D Plots

Importing an Image as a Background Field

3. In the **Name** field, enter a name for the new dataset containing the image data.



4. Select **New Plot** to create a new plot containing the dataset.

If this option is not selected, then the dataset will be overlaid onto the active plot.

5. Click **OK**.

Alternatively, you can load an image using the `load_file` and `overlay_plots` Tcl commands. For example:

```
% load_file "pic.png"  
% overlay_plots {Plot_2D} -datasets {pic}
```

See [load_file on page 318](#) and [overlay_plots on page 323](#).

Adjusting Magnification of an Image

After an image is loaded and overlaid onto a plot, you can adjust the magnification to a specific area in the image.

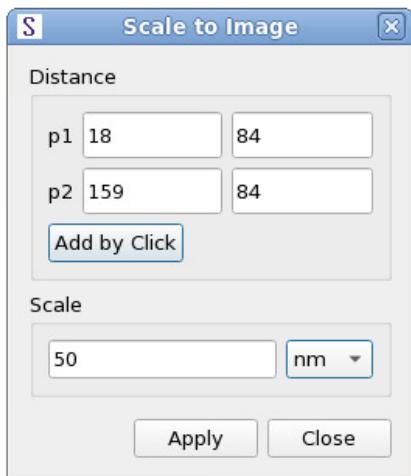
Chapter 4: Working With 2D and 3D Plots

Importing an Image as a Background Field

To adjust the magnification of an image:

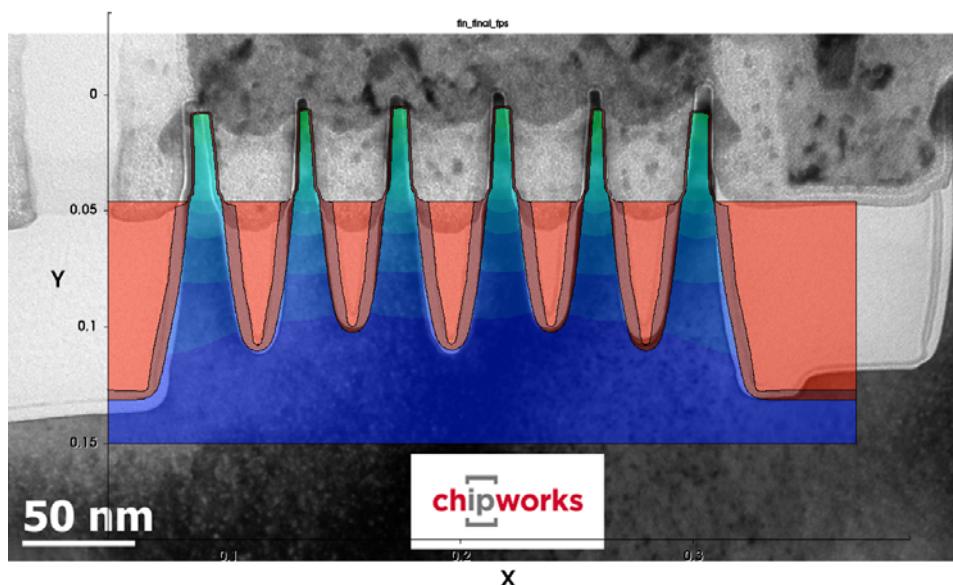
1. Choose **View > Scale to Image**.

The Scale to Image dialog box opens.



2. Specify two points of the screen: the first is the x-coordinate and the second is the y-coordinate.
3. In the **Scale** field, specify the length and unit that will be the scale used for the distance between the two points.
4. Click **Apply**.

Figure 84 Example of a plot with an overlaid image

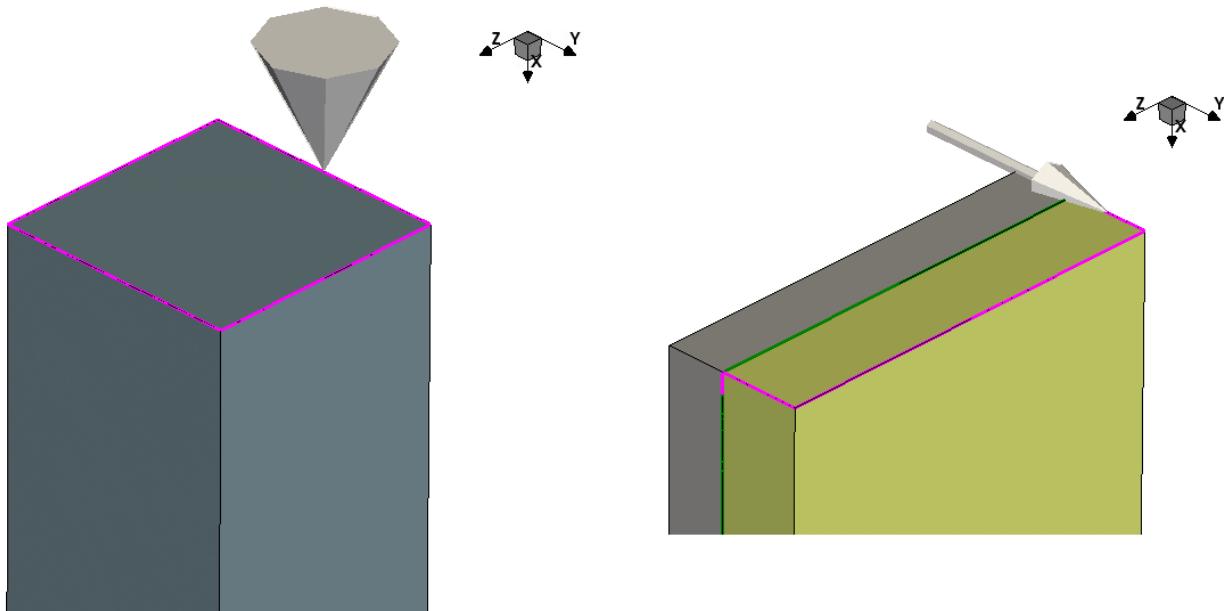


Visualizing Point Boundary Conditions

You can visualize point boundary conditions from Sentaurus Interconnect. Such visualization is switched off by default and is activated by setting `main\loadPointBoundaries=true` in the `PlotHD` section of the `sVisual.conf` file.

Zero point displacement rates are displayed as gray cones pointing to the specific location, while point force conditions are displayed as gray arrows.

Figure 85 (Left) Zero point displacement rate and (right) point force condition



Scaling and Shifting 2D and 3D Geometries

Sentaurus Visual can transform the geometry of 2D and 3D geometries, and change how they are visualized in a plot. Two geometric transformations are available: scale and shift.

Note:

These transformations affect only the visualization of data, not the original geometry data.

To scale or shift a geometry:

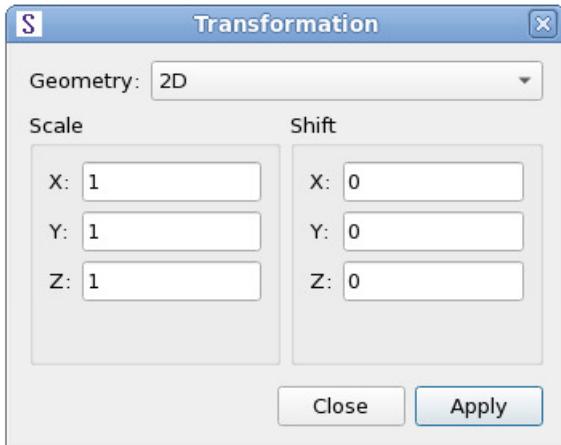
1. Choose **Tools > Transformation**.

The Transformation dialog box opens.

Chapter 4: Working With 2D and 3D Plots

Rotating Structures (3D Plots Only)

2. Change the scale and shift values for each axis for a certain geometry.



3. Click **Apply**.

Rotating Structures (3D Plots Only)

Three-dimensional plots can be rotated freely over a rotation point or fixed to an axis.

To rotate a plot, you must be in selection mode (click the toolbar button). Drag to rotate the plot. When you release the mouse button, the rotation stops.

To set the origin of a rotation point, use the camera configuration by either choosing **View > Camera Configuration** or placing the cursor on the required point and pressing the O key.

Table 8 Rotation modes

Toolbar button	Shortcut keys	Description
	Press the N key while dragging the cursor.	Enables standard rotation until you release the N key.
	Press the S key while dragging the cursor.	Enables spherical rotation until you release the S key.
	Press the X key while dragging the cursor.	Fixes the rotation around the x-axis until you release the X key.
	Press the Y key while dragging the cursor.	Fixes the rotation around the y-axis until you release the Y key.
	Press the Z key while dragging the cursor.	Fixes the rotation around the z-axis until you release the Z key.

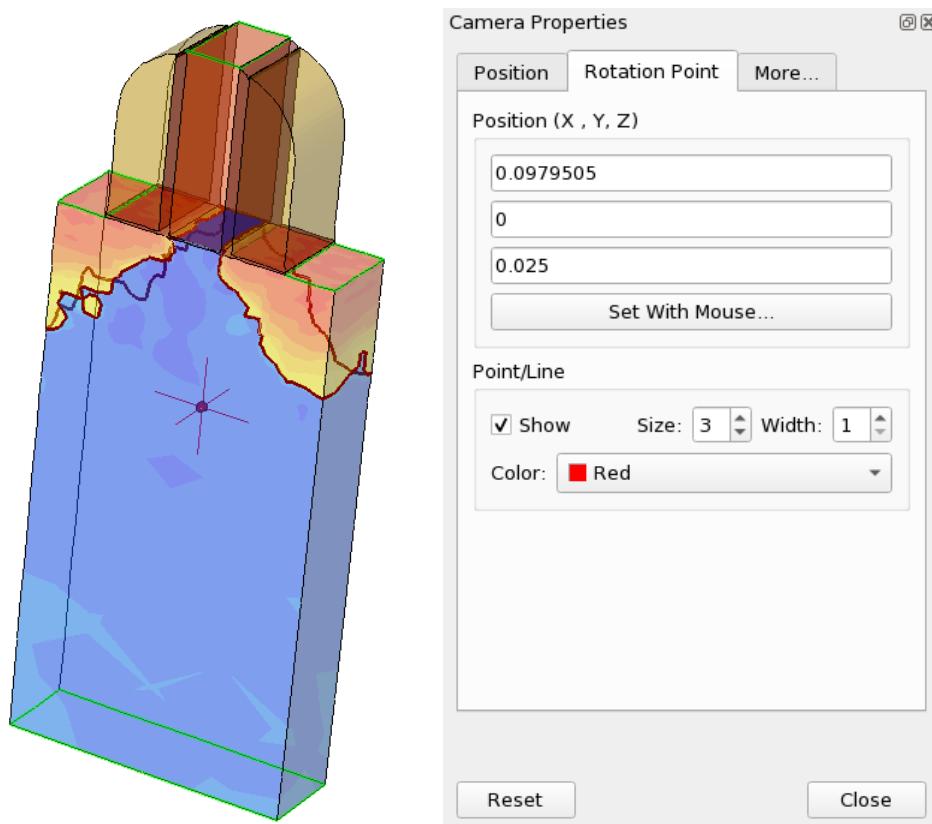
Rotation Point

The rotation point is the reference location for free rotation, that is, when not rotating around an axis. Default coordinates are calculated for every plot, locating the rotation point in the center of the initial visible structure. The same rotation point also is used in Spherical Rotation mode.

The rotation-point coordinates can be changed along with the properties of the rotation point by using either:

- The menu bar (choose **View > Camera Configuration** and, in the Camera Properties panel, click the **Rotation Point** tab)
- The `set_camera_prop` command (see [set_camera_prop on page 344](#))

Figure 86 (Left) Three-dimensional structure showing the default rotation point (translucency activated) and (right) the Rotation Point tab in the Camera Properties panel



Customizing the Rotation Point

The rotation point is visualized as a tiny sphere with axes crossing it over three dimensions. Moreover, the properties of the rotation point can be customized. [Table 9](#) summarizes these properties.

Table 9 Properties of the rotation point that can be customized

Property	Description	Tcl command example
Color	Determines the color of the rotation point when it is visible in <#rrggbb> format.	<code>set_camera_prop -rot_color #FF00FF</code>
Size	Sets the length of the rotation point axes. The size is an integer.	<code>set_camera_prop -rot_size 5</code>
Width	Sets the thickness of the rotation point axes. The width is an integer.	<code>set_camera_prop -rot_width 3</code>
Visibility	Determines whether to show or hide the rotation point.	<code>set_camera_prop -show_rotation_point</code> <code>set_camera_prop -hide_rotation_point</code>
Position	Sets the location of the rotation point. The position is defined by three floating-point values.	<code>set_camera_prop -rotation_point {0.0 1.0 -0.8}</code>

Using the Rotation Point as a Reference Point

The rotation point can be used as a reference point when inspecting 3D structures. As previously mentioned, you can move the rotation point to a specific position using either the user interface or the `set_camera_prop` command. Using the Tcl command, it is only possible to set the position by exact coordinates. For example:

```
set_camera_prop -rotation_point {0.0 1.0 -0.8}
```

Using the user interface, the position of the rotation point can be set by either exact coordinates or the cursor. In both cases, you must show the **Rotation Point** tab by choosing **View > Camera Configuration** to display the Camera Properties panel. On the **Rotation Point** tab, the **Position** fields allow you to introduce the values of the x-axis, y-axis, and z-axis of the position. The position is updated every time you leave any of the fields or if you press the Return key.

Chapter 4: Working With 2D and 3D Plots

Rotating Structures (3D Plots Only)

Another option is to use the **Set With Mouse** button. To set the position of the rotation point with this feature:

1. Click **Set With Mouse**.

The button remains selected. Note that the cursor changes appearance to a cross when it hovers over the plot.

2. Click the structure at the required location to set the position of the rotation point. Note that the point will be set at the surface of the structure.

After this, the **Set With Mouse** button is released, and the cursor returns to its previous state.

The rotation point is visible while the **Set With Mouse** button remains selected. Since the rotation point is in the inner part of the structure by default, it is not visible unless translucency is enabled or any region obstructing its view is hidden.

To make the rotation point permanently visible, select **Show** (see [Figure 86 on page 146 \(right\)](#)). Deselect the option to hide the rotation point.

Note:

A shortcut exists to set the rotation point when using the **Set With Mouse** button:
Hover the cursor in the required location over the structure and press the O key.
This will set the position of the rotation point in the same way as the **Set With Mouse** button.

To set the rotation point inside the structure, you can either:

- Set the rotation point on the surface and, then modify it by using the `set_camera_prop` command or by setting the values in the **Position** fields.
- Hide regions or materials before setting the rotation point with the cursor.

Note:

The position of the rotation point is constant with regard to deformation, value blanking, and other Sentaurus Visual features that alter the structure.

Rotating Plots Using Exact Values

You can rotate 3D plots precisely using the Rotate dialog box (choose **View > Rotate**) or the corresponding `rotate_plot` command (see [rotate_plot on page 336](#)).

The dialog box has the following tabs (see [Figure 87 on page 149](#)):

- On the **XYZ Axis** tab, you can rotate the structure around the x-, y-, or z-axis using relative angles with a defined number of steps, or you can set absolute angles.

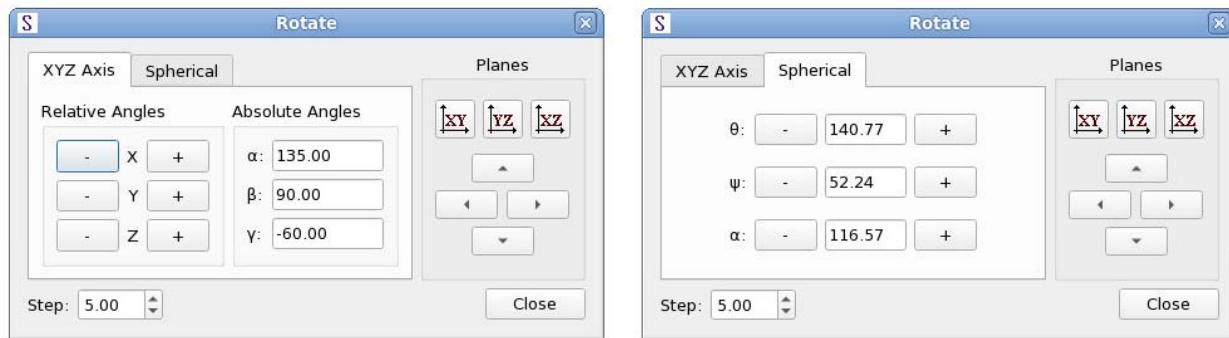
Chapter 4: Working With 2D and 3D Plots

Rotating Structures (3D Plots Only)

- On the **Spherical** tab, you can rotate the structure using spherical coordinates (see [3D View on page 119](#) and [Figure 67](#)) with a defined number of steps, or you can change the angles directly.

The **Step** field applies to both tabs and is used to define the number of steps.

Figure 87 Rotate dialog box showing (left) XYZ Axis tab and (right) Spherical tab



The **Planes** group area is independent of the rotation mode. You can change the view of the structure to a specific plane using the **View Plane XY**, **View Plane YZ**, and **View Plane XZ** buttons, or you can rotate the structure 90° in different directions using the arrow buttons.

The rotations performed by the arrow buttons are equivalent to the rotations performed by mouse operations when the rotation mode is in its default state (see [Figure 88](#)):

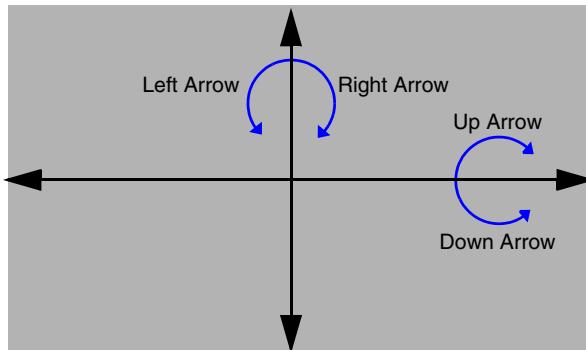
- The **Left Arrow** button and the **Right Arrow** button rotate the plot around an imaginary, completely vertical vector that is located at the rotation point of the plot.
- The **Up Arrow** button and the **Down Arrow** button rotate the plot around an imaginary, completely horizontal vector (perpendicular to the vertical vector) that is located at the rotation point of the plot.

You can use the arrow keys of the keyboard as shortcut keys to rotate the plot in the same way as the arrow buttons of the Rotate dialog box. This functionality is available when a 3D plot is selected or when the Rotate dialog box is open.

Chapter 4: Working With 2D and 3D Plots

Overlaying Plots

Figure 88 Directions of rotation performed when using the arrow buttons



Overlaying Plots

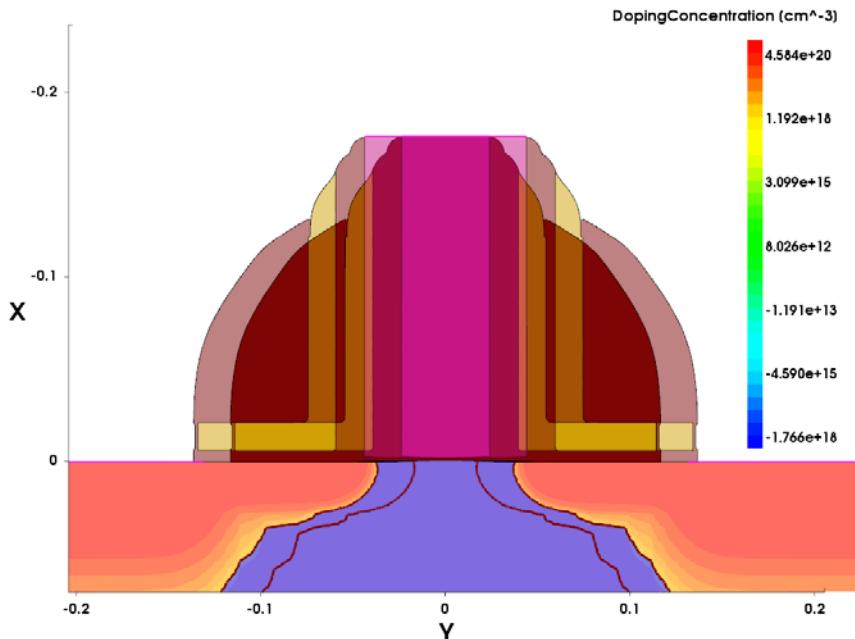
Overlaying 2D or 3D plots allows you to examine the differences between two similar plots.

To overlay plots:

1. Select two or more plots to be overlaid.
2. Click the  toolbar button.

A new plot is generated with the selected plot structures overlaid.

Figure 89 Example of overlaying plots

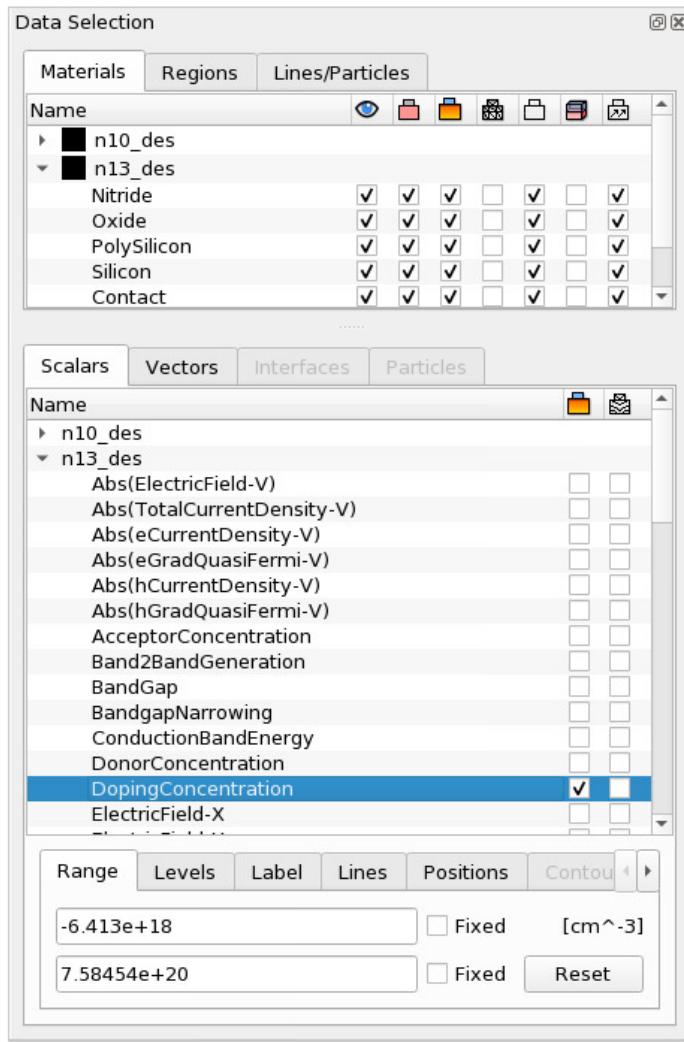


Chapter 4: Working With 2D and 3D Plots

Overlaying Plots

When plots are overlaid, the Data Selection panel changes to a tree view to allow for the visualization of different geometries as shown in [Figure 90](#).

Figure 90 Data Selection panel showing tree view when plots are overlaid



The geometries can be easily distinguished by selecting different boundary or contour line colors.

To select a specific boundary line color:

1. On the **Materials** tab, double-click the filled rectangle preceding the geometry name.
2. Choose a color from the list, and press the Enter key.

Chapter 4: Working With 2D and 3D Plots

Showing Differences Between Plots

To select a specific contour line color:

1. On the **Scalars** tab, **Interfaces** tab, and **Particles** tab, double-click the filled rectangle preceding the geometry name.
2. Choose a color from the list, and press the Enter key.

If you want to change the colors of specific materials or regions, you must use the Region Properties dialog box (press Ctrl+Shift+E). See [Modifying Properties in Multiple Materials and Regions on page 131](#).

Showing Differences Between Plots

Differentiating 2D or 3D plots allows you to determine differences in the common fields of two different plots. A new plot is generated showing the field differences between the two plots.

To differentiate plots:

1. Select two plots with common fields.
2. Click the  toolbar button.

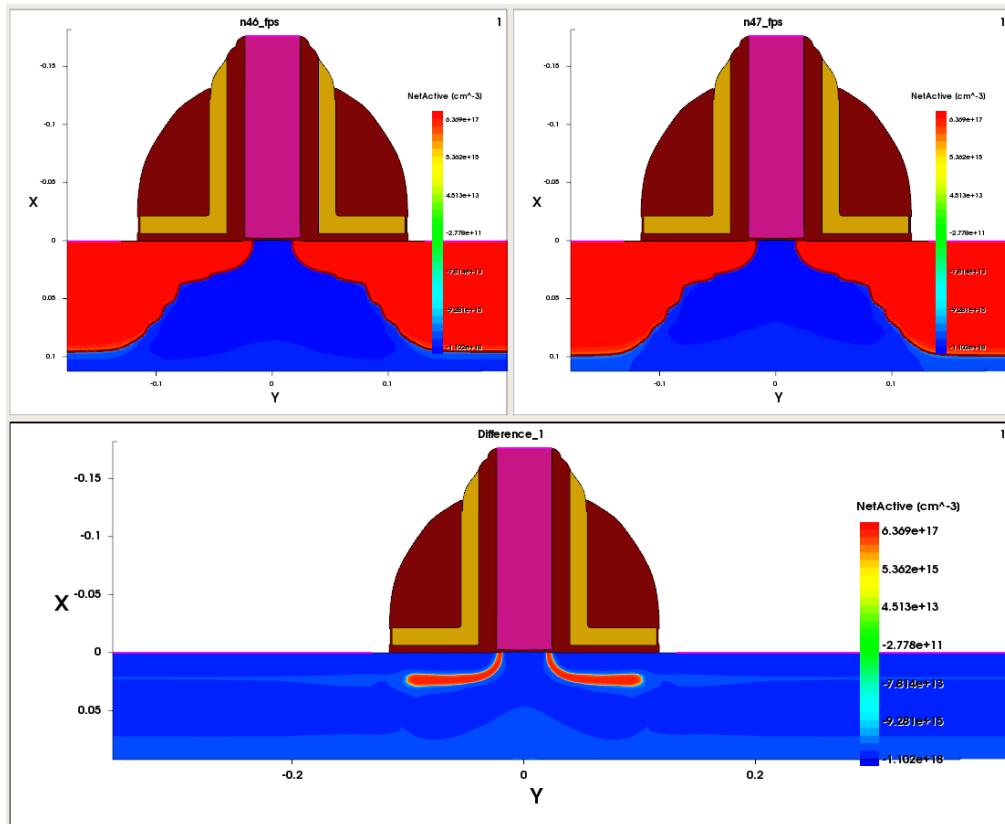
Note:

The objective of this functionality is to compare two similar plots. Therefore, a correct result is not guaranteed if the plots do not have the same number of regions with the same names. A warning message is displayed if there is a mismatch.

Chapter 4: Working With 2D and 3D Plots

Measuring Distances

Figure 91 Example of a difference plot resulting from comparison of two plots



Measuring Distances

You can measure the distance between two points in a plane or in space for 2D and 3D plots.

To measure a distance in a plot:

1. Click the **Ruler**  toolbar button.
2. Drag from the starting point of the measurement.
3. Release the mouse at the end point of the measurement.

Note:

You also can create a new ruler by using the `create_ruler` command (see [create_ruler on page 236](#).

Chapter 4: Working With 2D and 3D Plots

Measuring Distances

To modify the ruler:

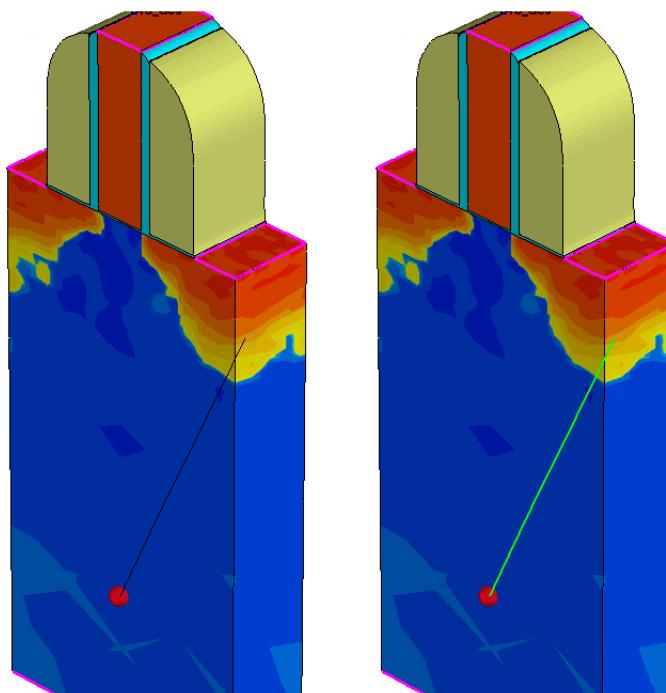
1. Move the cursor over the start or end point.
2. Click the point to select it.
3. Drag the point.

Note:

Holding the Alt key while dragging limits the movement horizontally or vertically only.

You also can modify a ruler by using `set_ruler_prop` command (see [set_ruler_prop on page 369](#)).

Figure 92 (Left) Selected point is highlighted and (right) dragging point



You can keep a selected ruler on screen and create a new one by clicking the **+** tab on the Ruler panel. Each ruler has its own numbered tab and you can click tabs to move between rulers.

The **Data** tab of the Ruler panel shows the coordinates and distances calculated. Clicking the **Remove** button deletes the ruler from the plot and the Ruler panel (see [Figure 93](#)).

The **Properties** tab shows the ruler properties and provides a snap-to-nearest point function. When you select **Snap to Mesh**, Sentaurus Visual automatically selects the

Chapter 4: Working With 2D and 3D Plots

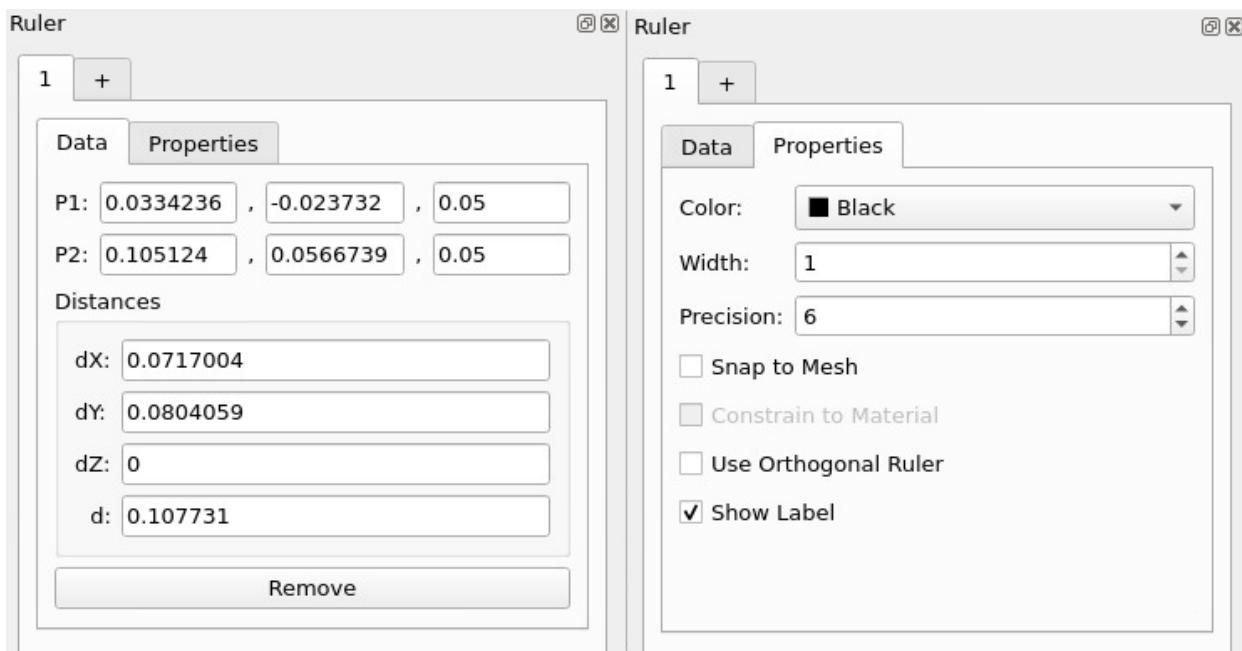
Integration Tool

nearest point in the grid to the one clicked when measuring distances. You can also change ruler properties by using [set_ruler_prop on page 369](#).

While measuring distances, you can rotate a structure using any of the shortcut keys (see [Rotating Structures \(3D Plots Only\) on page 145](#)).

You can also choose the selection mode and rotate the structure (the ruler is permanent). In this mode, you cannot select the ruler until the ruler mode is reactivated.

Figure 93 Ruler panel showing (left) Data tab and (right) Properties tab



Integration Tool

You can integrate the active field on all the materials of the current 2D or 3D plot.

To enable the integration tool, click the $\int dr$ toolbar button.

The Field Integration dialog box is displayed (see [Figure 94](#)) with the results of the integration for each material and a total value calculated over the active field. Integration can be performed on other fields without changing the active field displayed on the structure.

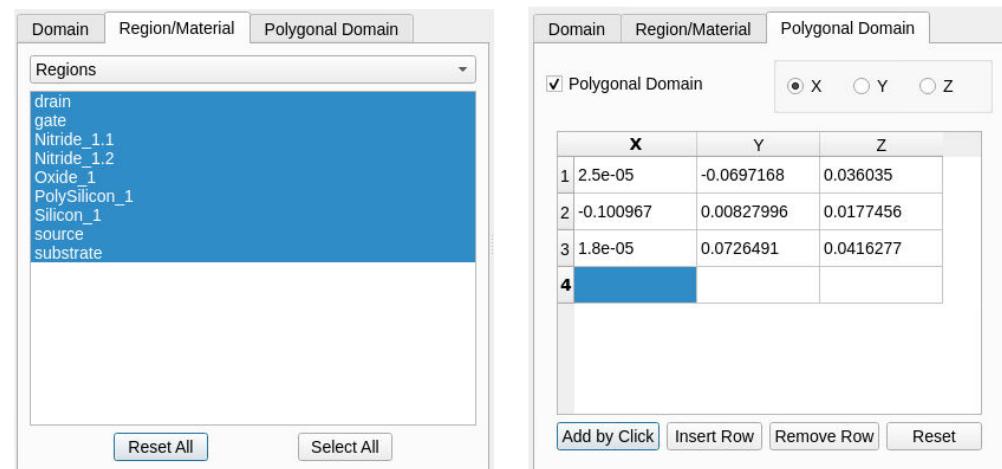
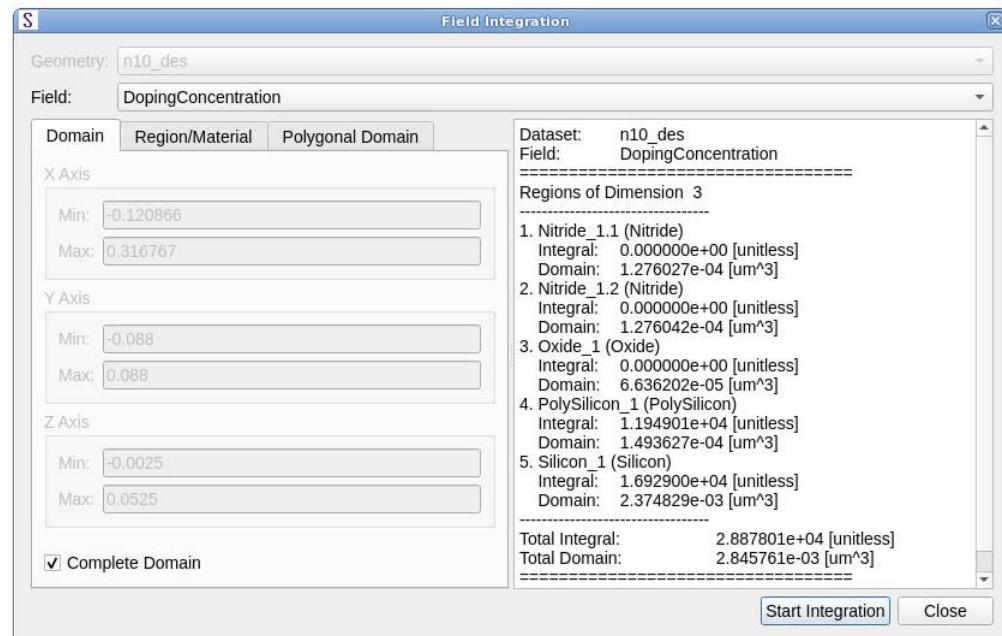
Integration over the active field commences immediately, but it can be stopped by clicking the **Cancel Integration** button of the Field Integration dialog box.

Integration on large structures can take some time. To see the progress of the integration, a progress bar is visible in the lower-right corner of the user interface.

Chapter 4: Working With 2D and 3D Plots

Integration Tool

Figure 94 Field Integration dialog box showing (top) Domain tab, (lower left) Region/Material tab, and (lower right) Polygonal Domain tab



Using a Custom Integration Domain

Integration can be performed over the complete structure, inside a defined bounding box, or inside a defined polyhedron domain.

To use a bounding box:

1. In the Field Integration dialog box, on the **Domain** tab, deselect **Complete Domain**.
2. Enter the custom ranges.
3. Click **Start Integration** to obtain the updated value.

To use a polyhedron domain:

1. In the Field Integration dialog box, on the **Polygonal Domain** tab, select **Polygonal Domain**.
2. Select **X** (default), **Y**, or **Z** as the extrusion axis.
3. Define the polygon to be extruded.

You can add polygon points interactively using the **Add by Click** button or manually in the table. Click the **Insert Row**, **Remove Row**, or **Reset** buttons to adjust the table if necessary.

4. Click **Start Integration** to obtain the updated value.
-

Integrating Only a Defined Set of Regions or Materials

By default, integration is performed over all regions or materials of a structure. This can be changed on the **Region/Material** tab of the Field Integration dialog box by selecting only the regions or materials that you want to integrate, and then clicking **Start Integration**.

Probing

For 2D and 3D plots, you can display information about a selected point on a structure.

To probe a point:

1. Click the  toolbar button.
2. Click the point to be evaluated.

The Probe panel opens, which shows various information about the point such as the values of all fields and information about the cell (see [Figure 95 on page 158](#)).

Chapter 4: Working With 2D and 3D Plots

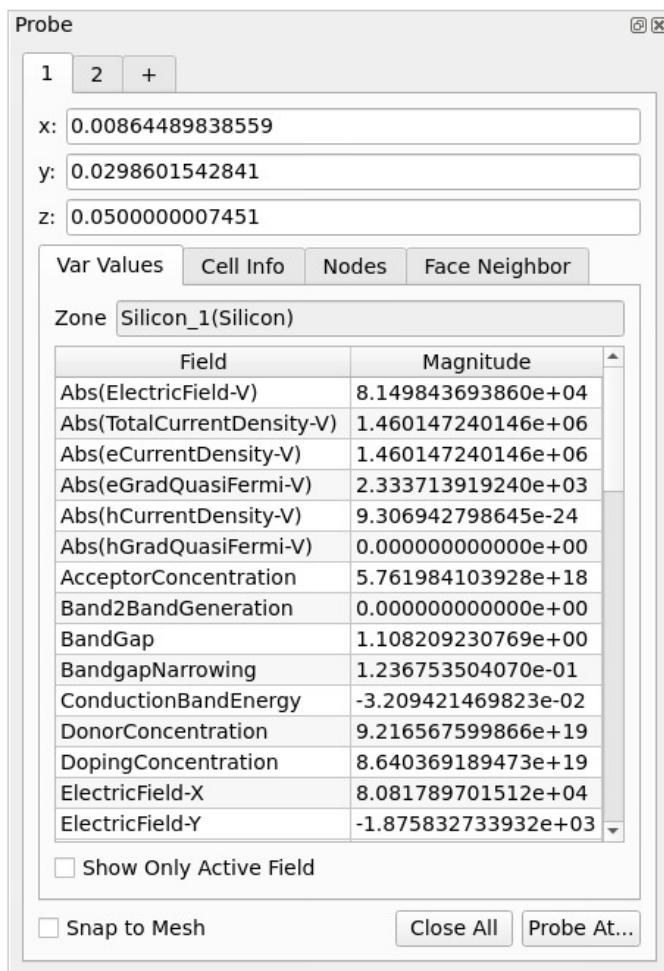
Probing

Note:

If you hold the Ctrl key when you click the point to be evaluated, the cursor (crosshairs) snaps to the closest mesh point. The same is achieved by selecting **Snap to Mesh** on the Probe panel, which provides information about the closest edge to the probed point of the structure.

You can keep a selected point on screen and probe other points. Each point has its own numbered tab and you can click tabs to switch the active point and highlight the cursor (crosshairs) in the plot. In the Probe panel, you can click the **Close All** button on the first point tab to delete all points except the first one.

Figure 95 Probe panel



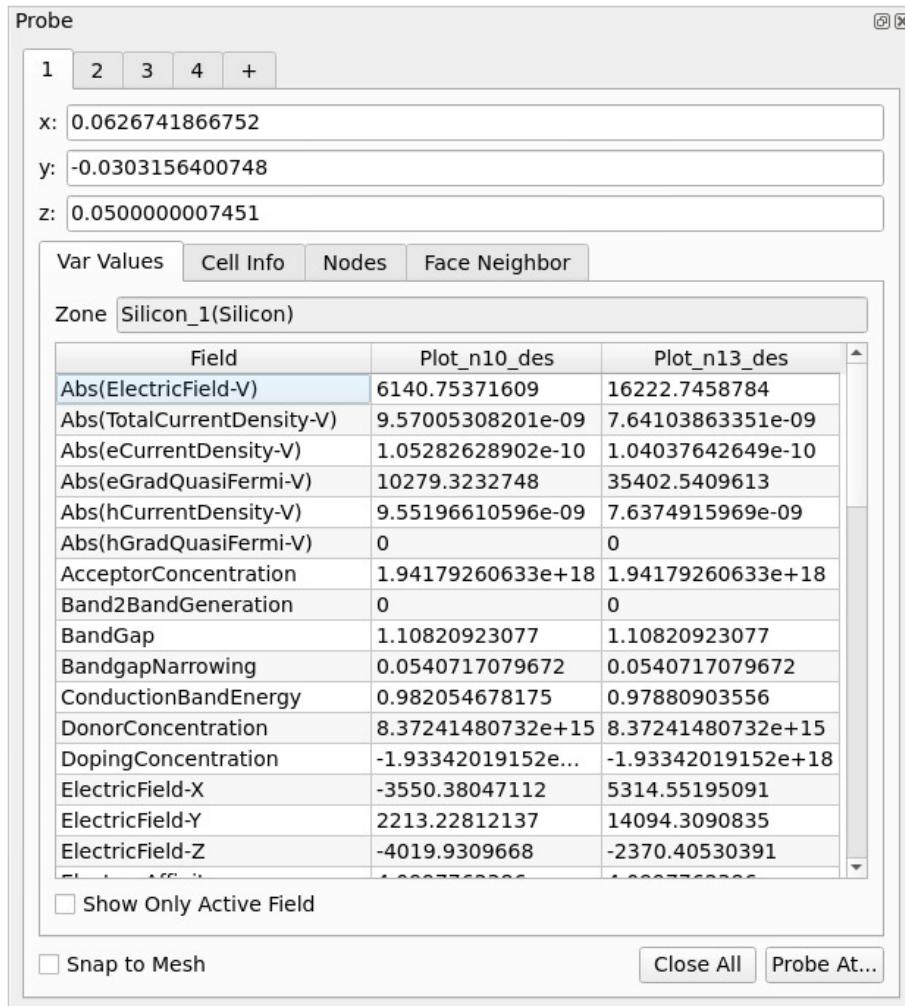
In addition, you can display information from different plot groups at the same time. However, only information that is available for all group members will be shown (see

Chapter 4: Working With 2D and 3D Plots

Probing

[Figure 96](#)). If one member of the plot group does not have the same information as the plot group leader, it will show *nan* (not a number) as the current value. This feature is compatible with displaying multiple crosshairs (see [Figure 97](#)).

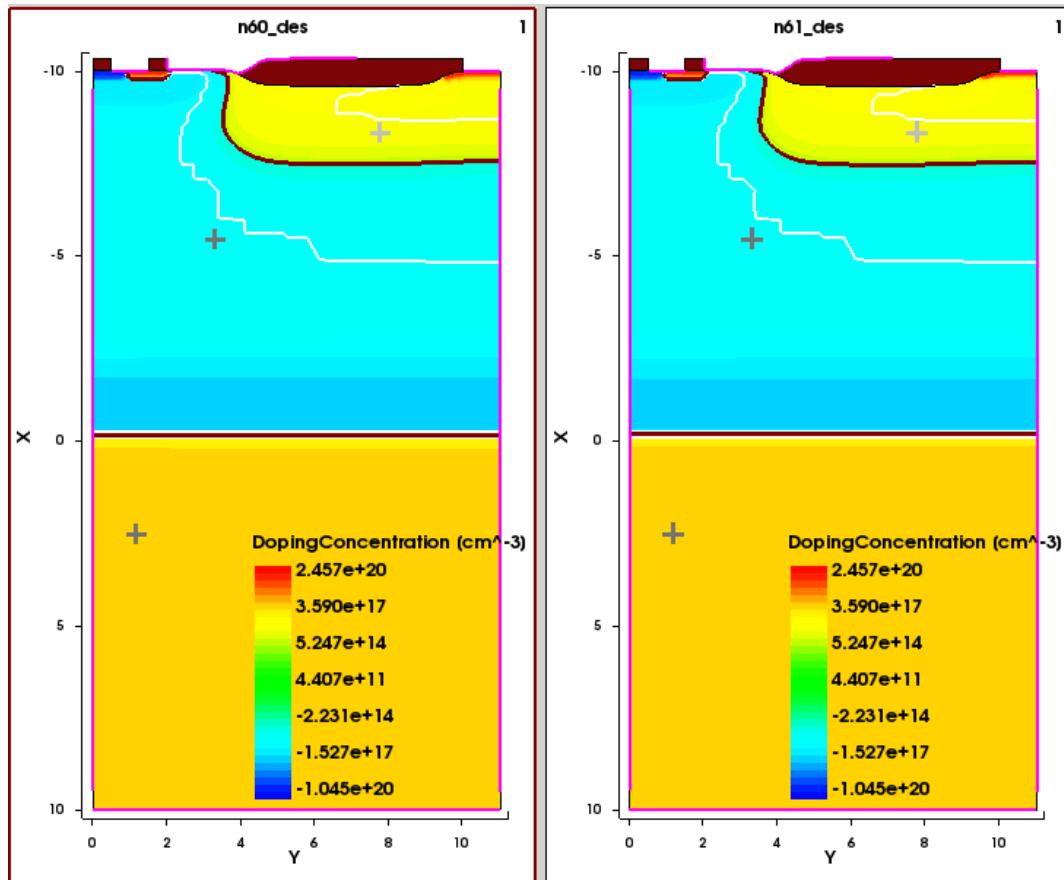
Figure 96 Probe panel showing multiple probe points on linked plots



Chapter 4: Working With 2D and 3D Plots

Probing

Figure 97 Multiple crosshairs shown in different plot groups



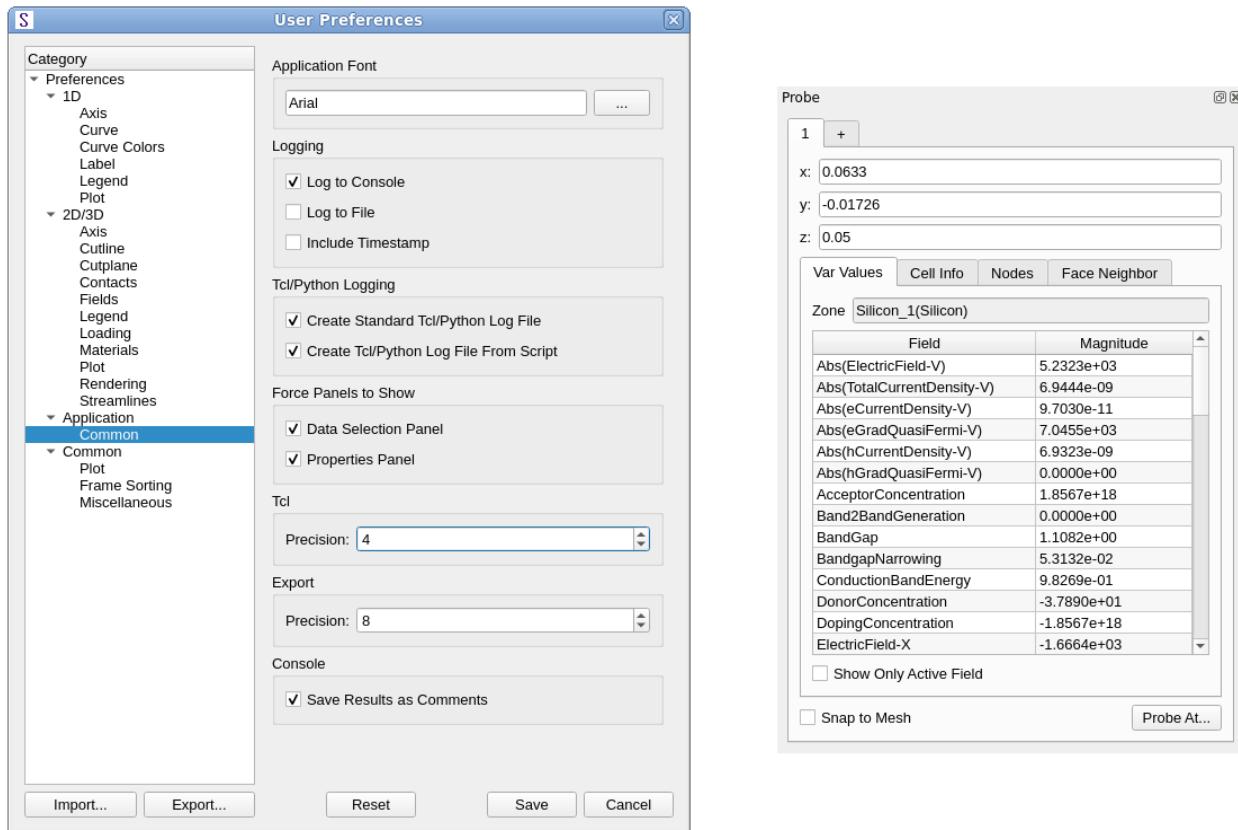
You can control the number of significant digits after a decimal point to be displayed. To specify the number of significant digits displayed after a decimal point:

1. Choose **Edit > Preferences**.
The User Preferences dialog box opens.
2. Expand **Application > Common**.
3. Under **Tcl**, change the value of **Precision**.
4. Click **Save**.

Chapter 4: Working With 2D and 3D Plots

Accessing Dataset Information

Figure 98 (Left) User Preferences dialog box with precision set to 4 and (right) Probe panel showing results with four significant digits after decimal point



Accessing Dataset Information

You can access 2D or 3D dataset information, such as the number of points or elements in a specific material or region. To display the Dataset Information dialog box, choose **Data > Dataset Information**.

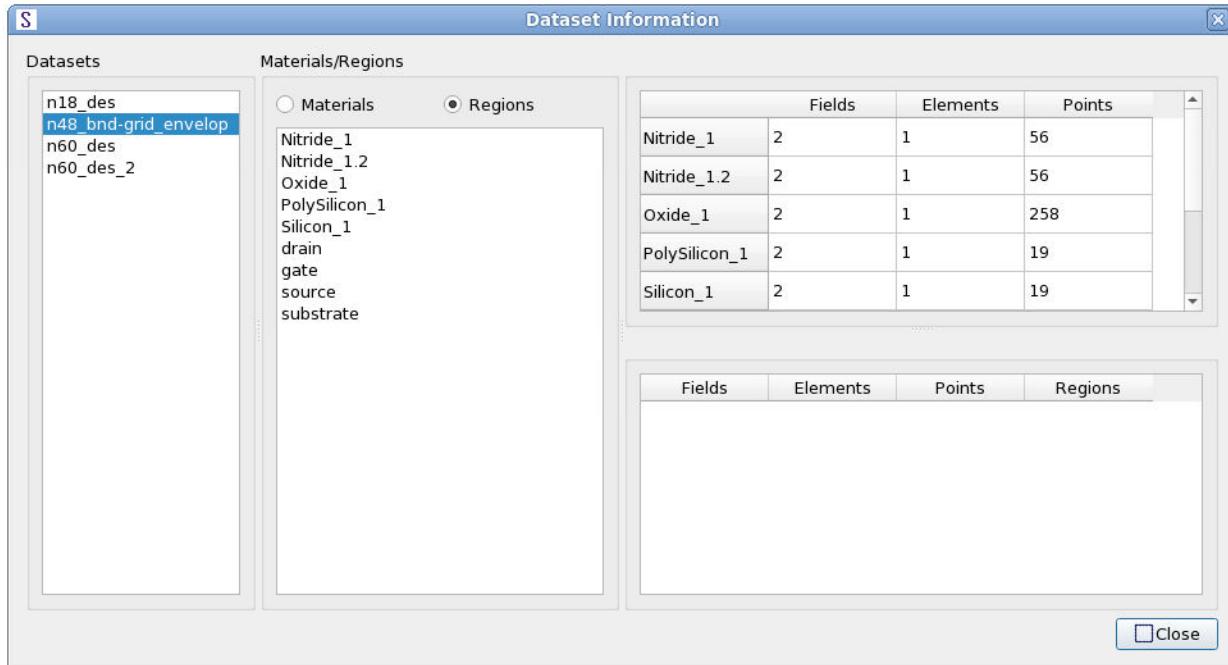
In this dialog box, relevant information about the dataset will be displayed, depending of the dataset and the materials or regions selected. The Datasets group box displays all the 2D and 3D datasets, so you can choose the dataset from which data is extracted (see [Figure 99 on page 162](#)). The number of fields, elements, and points are shown in the table located in the upper-right part of the dialog box.

In the Materials/Regions group box, you can select various materials or regions, and the sum of points and elements of each region selected is displayed in the table in the lower-right part of the dialog box.

Chapter 4: Working With 2D and 3D Plots

Maximum and Minimum Locations of Fields

Figure 99 Dataset Information dialog box



The total number of points and elements of a 2D or 3D dataset is displayed at any time at the bottom of the main window. It is important to mention that this value might not be the same at the one displayed in this dialog box, since this value is extracted directly from the geometry and the dialog box sums the points and elements of each region separately.

Maximum and Minimum Locations of Fields

Sentaurus Visual can easily display the maximum and minimum locations of a particular field.

To display these values:

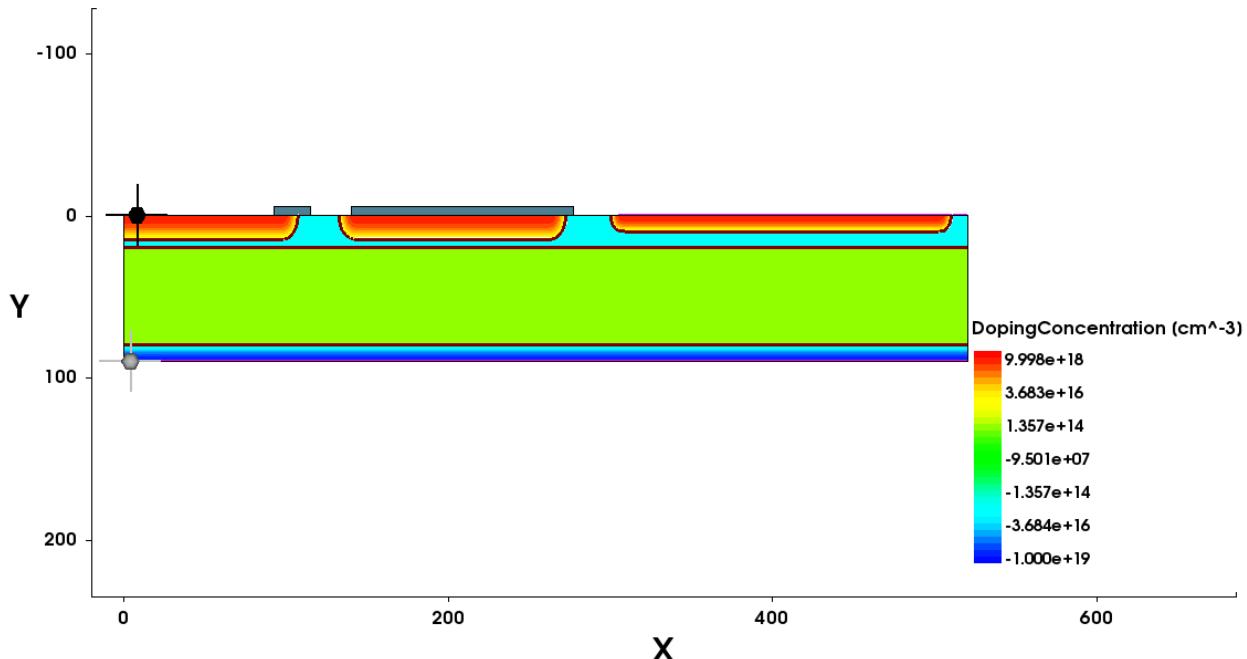
1. On the Plot Properties panel, click the **Markers** tab.
2. Select **Show Min** or **Show Max** or both options.

When either option is selected, a marker like the one shown in [Figure 100](#) is displayed.

Chapter 4: Working With 2D and 3D Plots

Maximum and Minimum Locations of Fields

Figure 100 Maximum marker (black circle with cross hairs) and minimum marker (gray circle with cross hairs) shown to left of structure



You also can define constraints for the search pool in the Minimum/Maximum Field Value dialog box. To display this dialog box, choose **Tools > Min/Max Field Value**.

In the Minimum/Maximum Field Value dialog box, you can select certain regions or materials for the search, and you can define a 3D box limiting the search area.

In the Minimum/Maximum Field Value dialog box, you can select certain regions or materials for the search on the **Region/Material** tab (see [Figure 101 on page 164](#)), and you can define a 3D box limiting the search area on the **Domain** tab. The **Reset** button on the **Domain** tab allows you to return to the initial values of all ranges (see [Figure 101 \(right\)](#)).

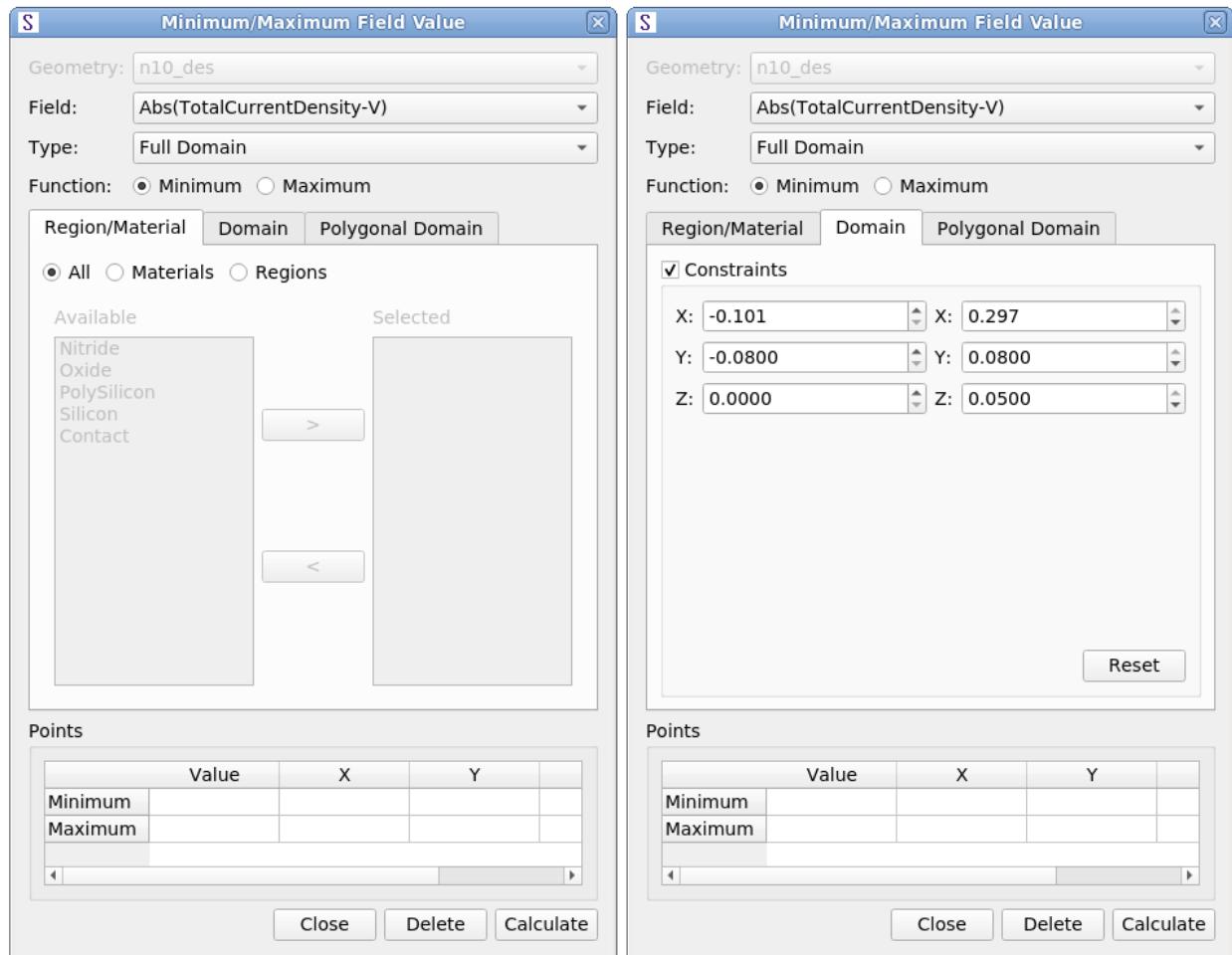
If you select the **Full Domain** type, then the minimum/maximum field value from the entire structure is returned. Otherwise, if you select the **Polygonal Domain** type, then the **Polygonal Domain** tab becomes available and the value is calculated from a defined polyhedron as shown in [Figure 102 on page 165](#).

First, select the **Extrusion Axis** (vertical axis in the resulting polygon). Then, define the **Selected Points** manually or click the **Add by Click** button. If a user-defined height is needed, then select **Height** to set the top and bottom heights of the polyhedron. A minimum of three points is required to perform this analysis.

Chapter 4: Working With 2D and 3D Plots

Maximum and Minimum Locations of Fields

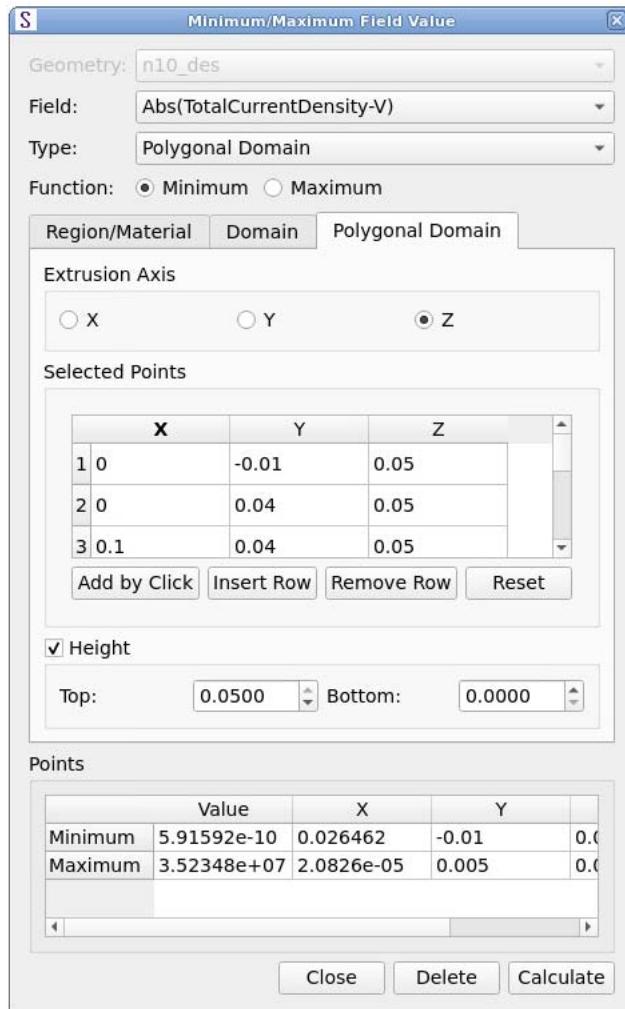
Figure 101 Minimum/Maximum Field Value dialog box: (left) Region/Material tab and (right) Domain tab



Chapter 4: Working With 2D and 3D Plots

Maximum and Minimum Locations of Fields

Figure 102 Minimum/Maximum Field Value dialog box showing Polygonal Domain tab



Changing Properties of Markers

You also can change the properties of the marker.

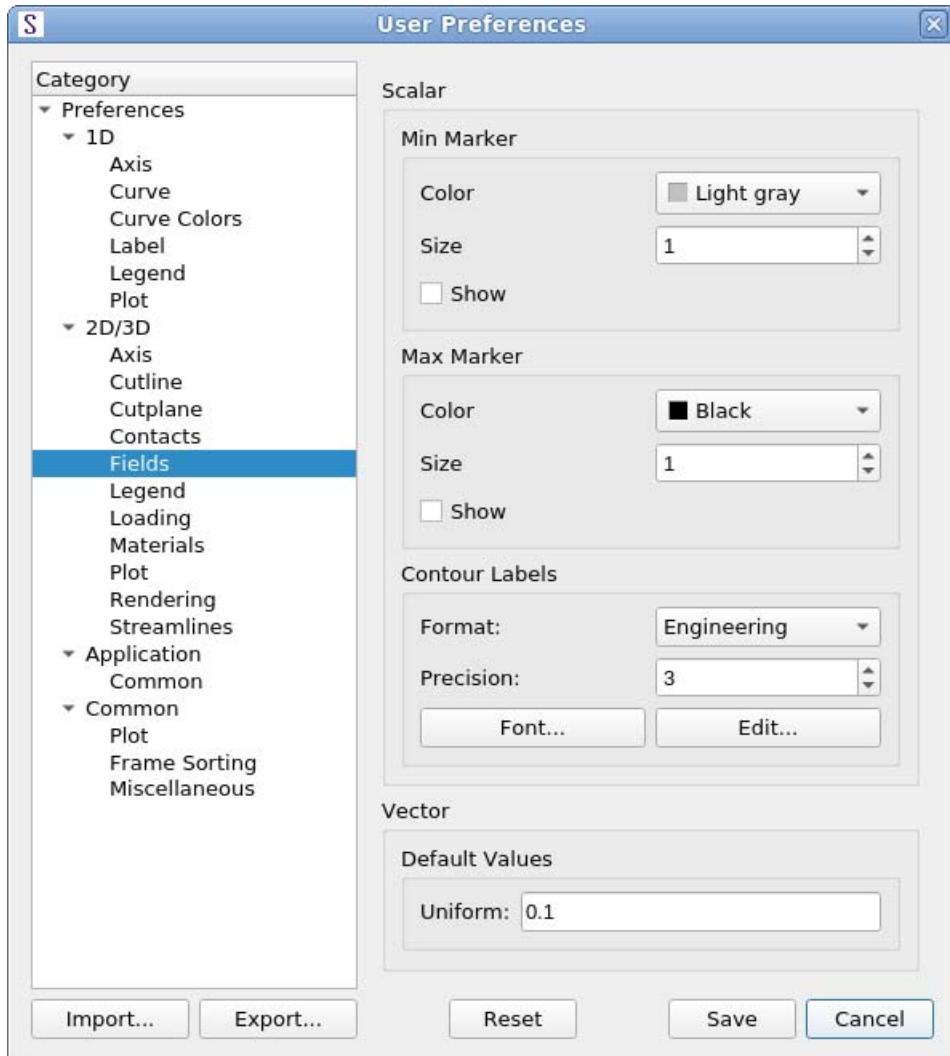
To change to the marker properties:

1. Choose **Edit > Preferences**.
2. In the User Preferences dialog box, expand **2D/3D > Fields** (see [Figure 103](#)).
3. Change the preferences as required.
4. Click **Save**.

Chapter 4: Working With 2D and 3D Plots

Value Blanking

Figure 103 Available preferences for minimum and maximum markers



You can change the color, the size, and the visualization of both the minimum and maximum markers. When you save the settings, they will be used in the next session of Sentaurus Visual.

Value Blanking

Value blanking allows you to display only the required areas of interest in a plot. You can enter multiple constraints to blank out areas that meet the criteria.

To use value blanking, click the  toolbar button. A dialog box is displayed (see Figure 104) where you can insert constraints on the required fields.

Chapter 4: Working With 2D and 3D Plots

Value Blanking

Value blanking keeps the enabled constraints even after closing the window. If you want to revert the changes, you must deactivate the specified constraint or reset all the constraints to return to the usual plot display.

Figure 104 Value Blanking dialog box



Choosing Constraints

In the Value Blanking dialog box, you can create a maximum of 10 constraints including different field values. Each constraint creates a particular set of data to be blanked. The sets constructed can be united or intersected, depending of the option selected for the particular constraint.

For example, if cons1 defines set *A* and cons2 defines set *B*, the result set to be blanked *C* can be either $C = A \cup B$ or $C = A \cap B$ depending of the option selected in cons2 (that is, either the **Union** option or the **Intersection** option) (see Figure 105).

Chapter 4: Working With 2D and 3D Plots

Value Blanking

Figure 105 Value Blanking dialog box showing constraints

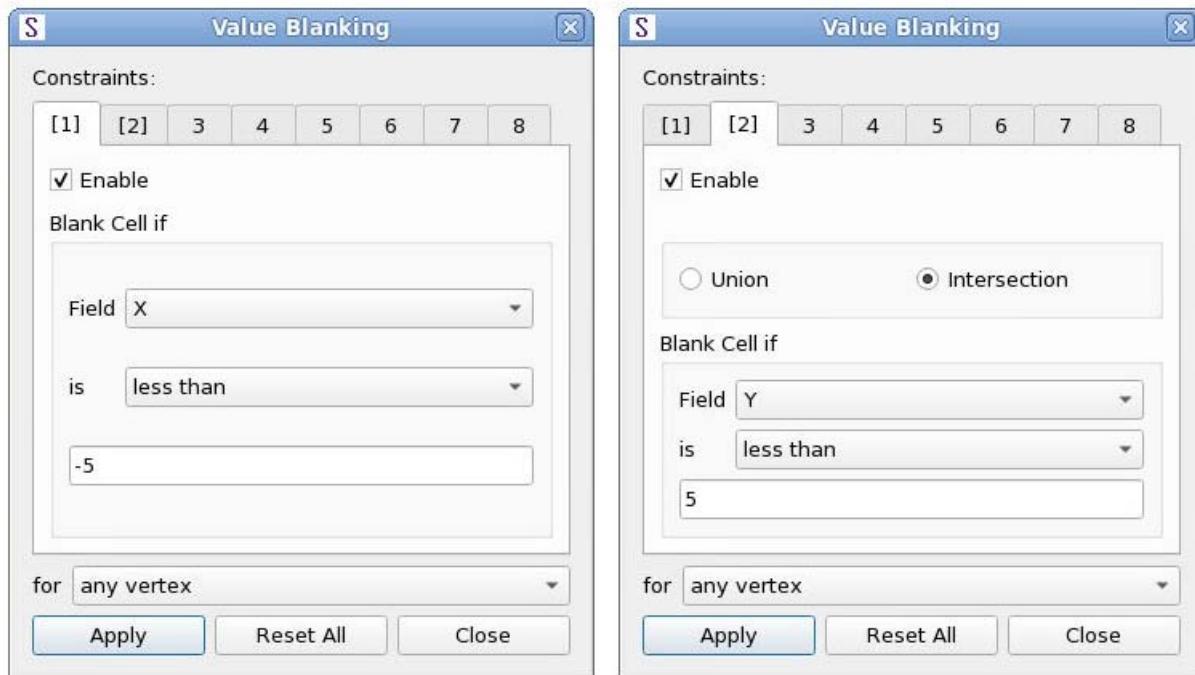
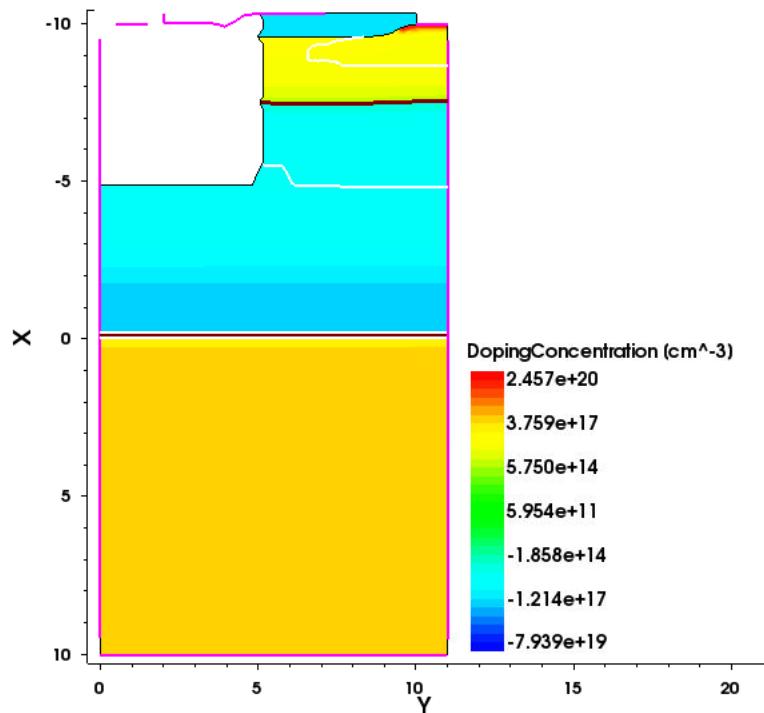


Figure 106 Blanked plot



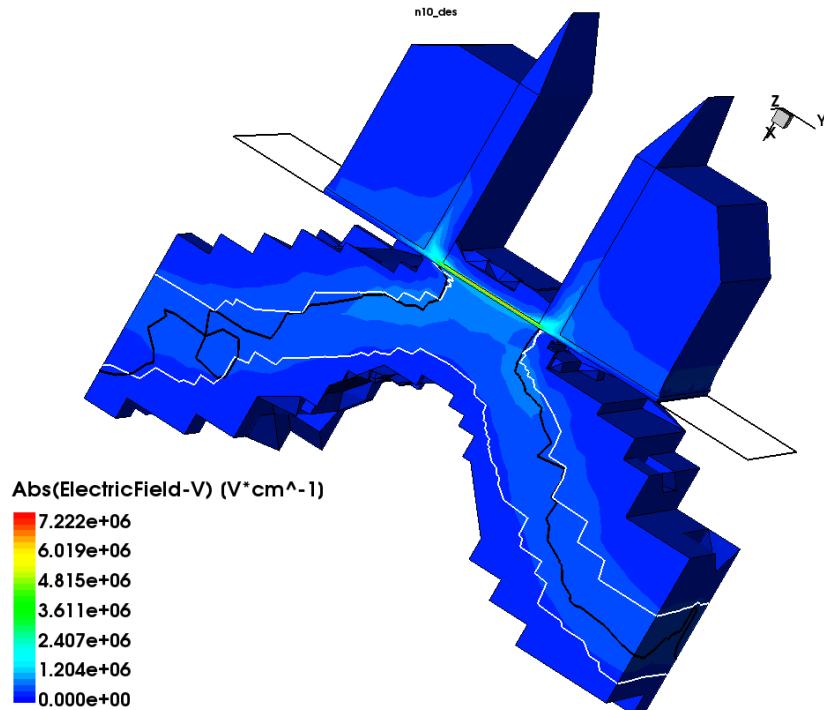
Options for Value Blanking

The options for value blanking are available from the **for** list of the Value Blanking dialog box (see [Figure 104](#)). The options are:

- **all vertices**
- **any vertex**
- **interpolate vertices**

The following figures show examples of using these options. In [Figure 109](#), the **interpolate vertices** option makes the surface of the field being blanked smoother than when the other options are selected.

Figure 107 Example of value blanking using the all vertices option



Chapter 4: Working With 2D and 3D Plots

Value Blanking

Figure 108 Example of value blanking using the any vertex option

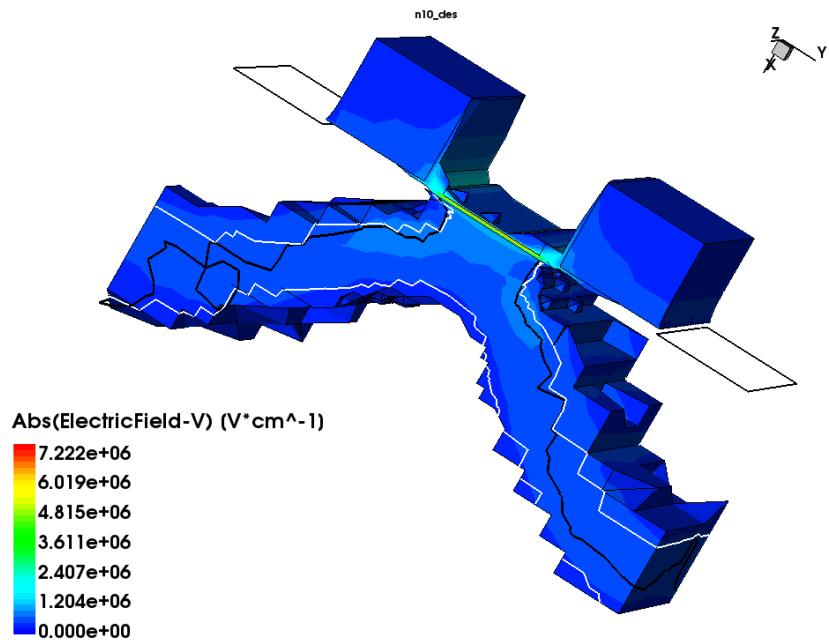
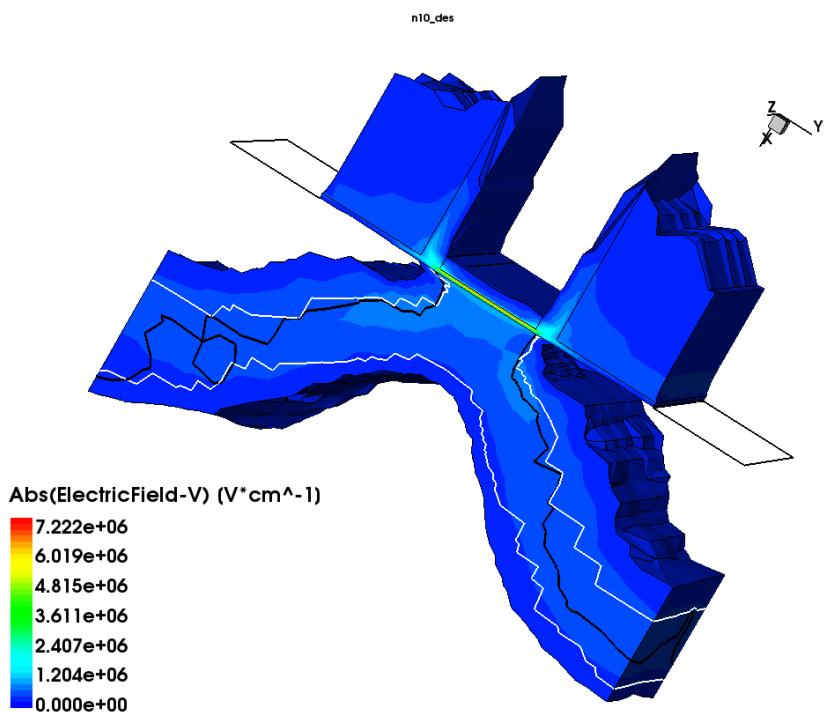


Figure 109 Example of value blanking using the interpolate vertices option



Visualizing Deformation of Structures

Sentaurus Visual allows you to deform a structure according to a certain vector field. In general, the required vector field that will be used to deform the structure is Displacement and, in Sentaurus Visual, it is called Displacement-V (all vector fields have the suffix -V).

Every region that contains the selected vector field will be deformed. This means that every point that defines the region will be moved in the direction and the magnitude of the vector. The magnitude of the displacement can be modified by a factor of the vector. This value is 1.0 by default (no factor is applied).

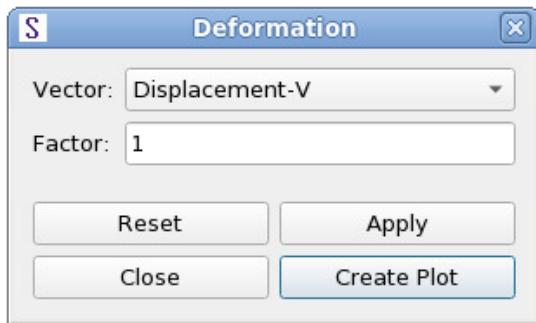
You can deform structures using either the user interface or the `set_deformation` Tcl command (see [set_deformation on page 351](#)).

In addition, you can apply a deformation factor or create a new deformation plot in a group of plots using the `link_plots` command. However, if you want to apply a deformation operation but not to all members of a plot group, you must create a special linked group without specifying the deformation property (see [link_plots on page 298](#)).

To display the Deformation dialog box, choose **Tools > Deformation** (see [Figure 110](#)). The Deformation dialog box includes the following fields and buttons:

- The **Vector** field is used to select the vector field that will be used to deform the structure.
- The **Factor** field is used to specify the factor of the magnitude vector to be used.
- The **Reset** button reverts the structure to its original state.
- The **Apply** button applies the specified deformation to the current plot.
- The **Create Plot** button creates a new plot with the same structure already deformed.

Figure 110 Deformation dialog box

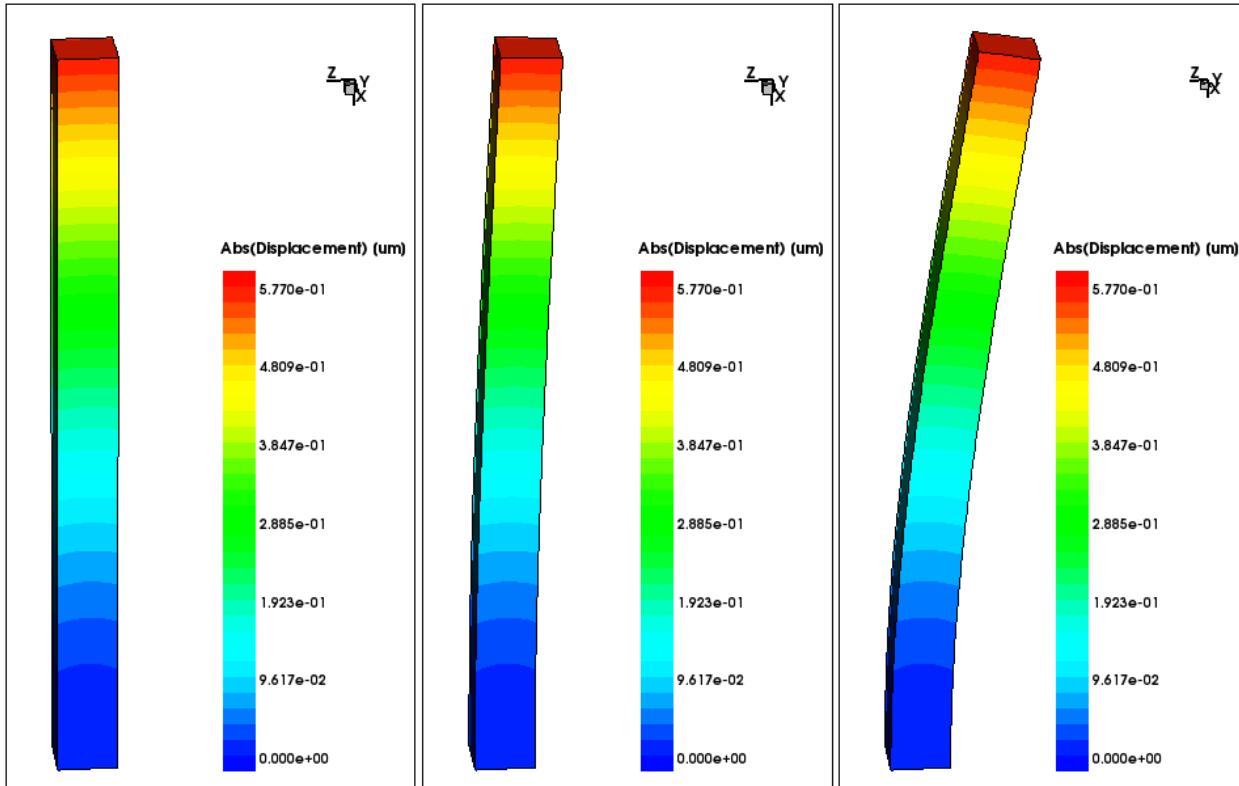


By default, if the **Displacement-V** vector field is available, it will be selected automatically in the **Vector** field. If this vector field is not available, no vector field will be selected. By default, the factor value is 1.0.

Chapter 4: Working With 2D and 3D Plots

Cutting Structures

Figure 111 (Left) Original structure, (middle) structure deformed by a factor of 3, and (right) structure deformed by a factor of 10



Cutting Structures

Sentaurus Visual provides tools for generating xy and 2D cuts, custom cutlines, and cutlines or cutplanes orthogonal to an axis.

Table 10 Tools for cutting structures

Toolbar button	Description
	Displays the Cutlines and Cutplanes dialog box, where you can generate non-orthogonal cutplanes or cutlines directly from a 3D plot, and you can cut in specific values.
	Creates a custom cutline in a 2D plot. The result is an xy plot of the selected field and datasets for all the fields on the cutline.
	Creates an orthogonal plot in one axis. The result is a 2D plot of the cutplane if cutting a 3D plot, or an xy plot from a cutline in a 2D plot. If an axis has a constant value, cuts for that axis are deactivated.

Note:

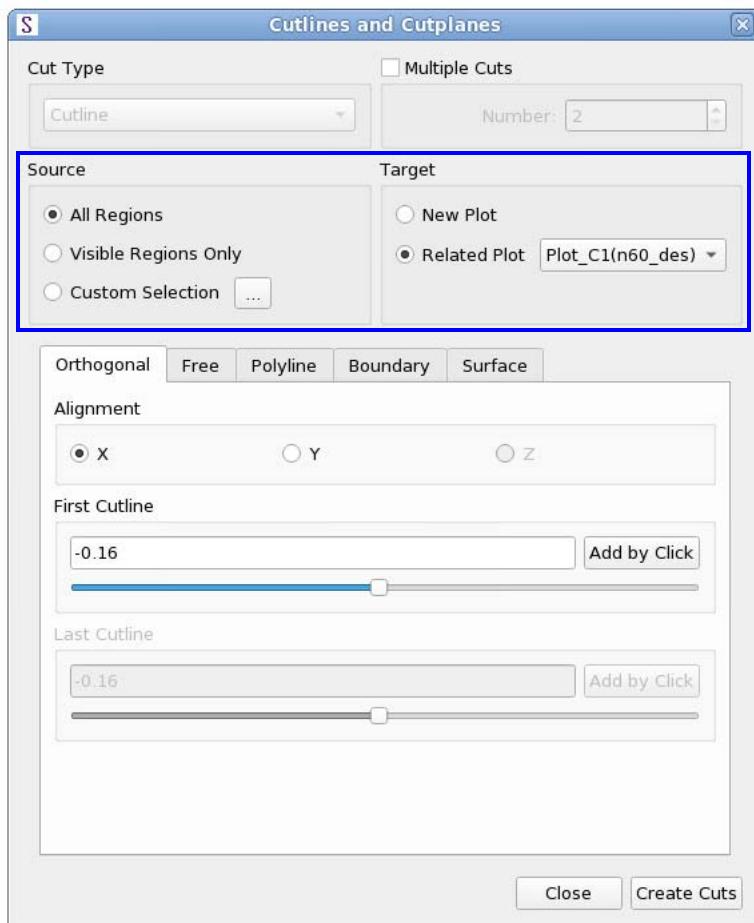
Cut tools ignore shifting or scaling transformations applied to the structure, regardless of visual appearance.

In linked plots, the newly cut structures are created by frame order *not* load order.

Generating Precise Cutlines and Cutplanes

Advanced options can be performed using the Cutlines and Cutplanes dialog box (see [Figure 112](#)), which allows a greater degree of precision than using mouse operations to generate cuts that require an exact point in the structure.

Figure 112 Cutlines and Cutplanes dialog box



Chapter 4: Working With 2D and 3D Plots

Cutting Structures

For 3D structures, you can perform cuts in one of the following ways:

- Orthogonally, by specifying an axis and a value
- Non-orthogonally, by specifying a normal and an origin point
- With a polyplane, by specifying an axis and the points of a polyline
- Creating a cutline directly from the structure

In addition, other cuts can be performed for 2D plots such as a polyline cut or a cut along boundaries. The Cutlines and Cutplanes dialog box provides various **Add by Click** buttons that allow you to click a point in the plot and to add it to a specific text box. The point clicked in this way will be marked in the plot.

To display the Cutlines and Cutplanes dialog box, click the  toolbar button.

For 2D and 3D structures, when **Cut Type** is set to **Cutline**, you can specify the regions to use as the source for the cutlines and specify the resulting target plot where the cutline curve will be displayed.

For the source, the following options are available under Source:

- **All Regions** (default): All regions are cut.
- **Visible Regions Only**: All visible regions are cut.
- **Custom Selection**: You select the regions or materials to be cut in the Custom Selection dialog box (see [Figure 113](#)).

Figure 113 Custom Selection dialog box



Chapter 4: Working With 2D and 3D Plots

Cutting Structures

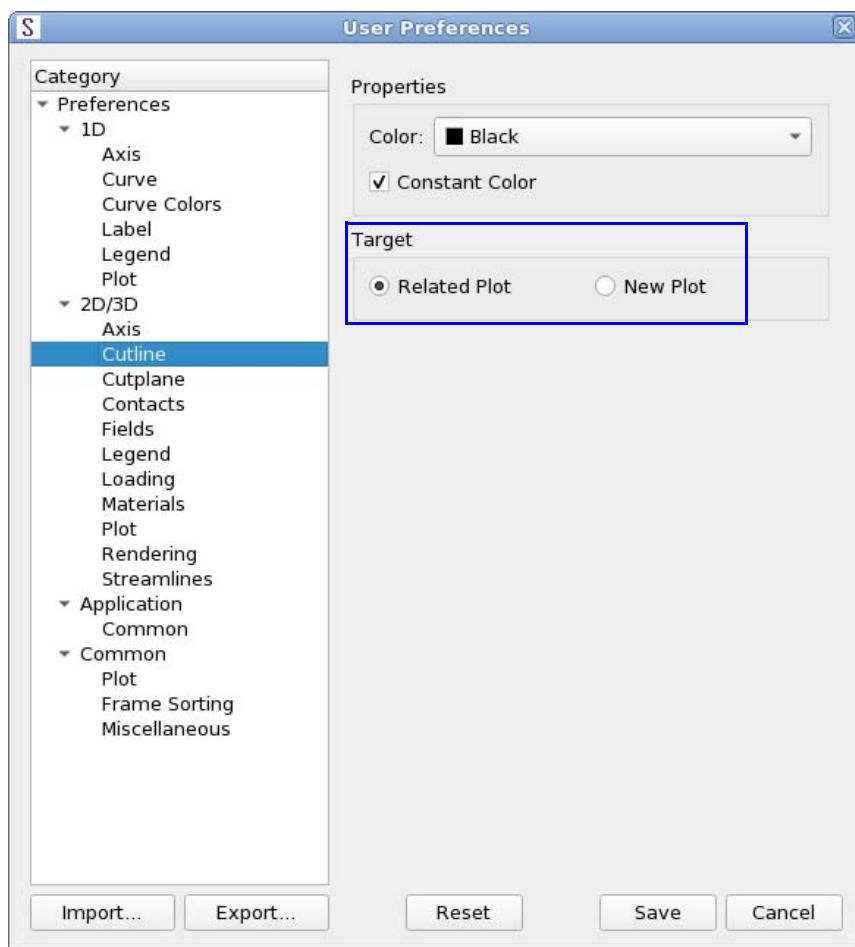
The default target plot is **Related Plot** (see [Figure 112](#)). A *related plot* refers to any xy plot that has already displayed at least one cutline dataset with the same type of the current cut. For example, if you perform an orthogonal x-cut, a related plot would be a plot created to display another orthogonal x-cut.

The list of the **Related Plot** option shows the plots associated with the cutline type specified:

- For 2D plots, this is for all cutline types (X, Y, Z, or Free), including polyline cuts and cuts along boundaries.
- For 3D plots, this is for free-type cutlines only.

You can use the User Preferences dialog box (see [Figure 114](#)) to override the default option of the Cutlines and Cutplanes dialog box.

Figure 114 Specifying the default for cutline target plots in the User Preferences dialog box



Cutlines in 2D Plots

Creating a new cutline is as easy as selecting an axis and a point in the plane to perform an orthogonal cut or drawing a line using the custom cutline button. The result is a new xy plot as shown in [Figure 115](#).

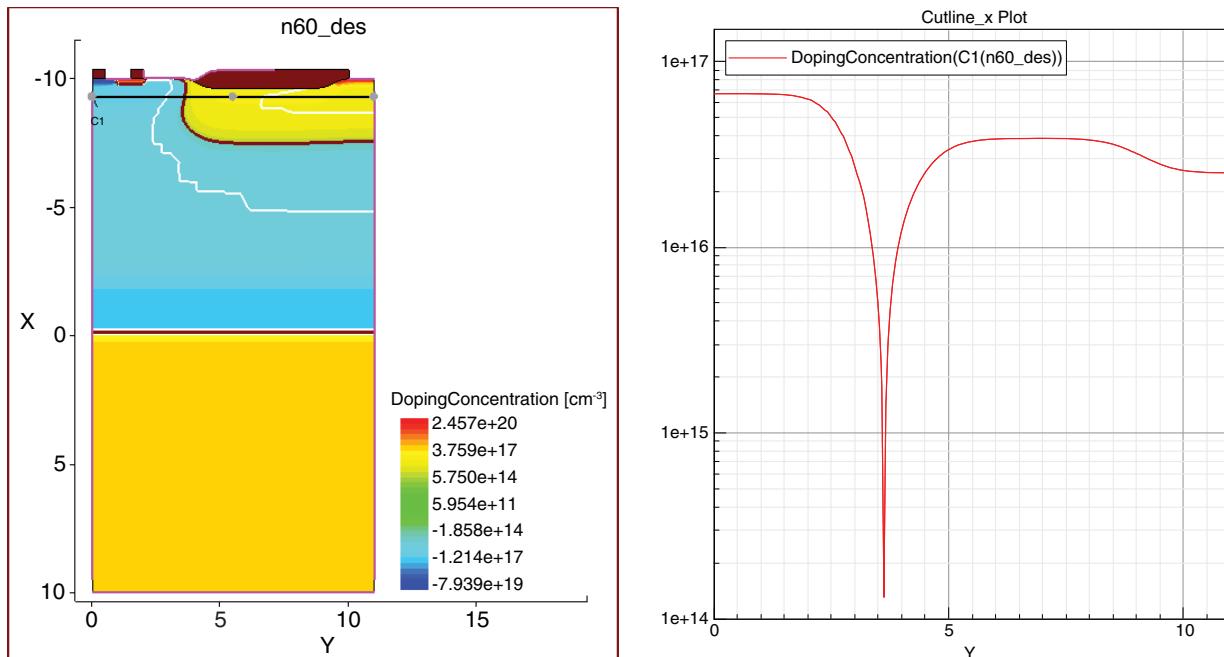
In the generated xy plot, the y-axis will be displayed in logarithmic scale if the active scalar field in the original 2D plot is visualized using one of the following scales: logarithmic (Log), logarithmic of the absolute (LogAbs), or hyperbolic arcsine (Asinh). Otherwise, the y-axis will be displayed in linear scale.

If the active scalar field uses a custom scale, the y-axis will also be displayed in linear scale. See [Visualizing Fields on page 107](#).

Note:

Visualization of the cutline result is confined to the current visible portion of the 2D plot. This occurs regardless of whether or not the auto-link or linking features are activated. This helps to maintain focus on the area of interest. To access all the data, click the **Reset Zoom** button.

Figure 115 (Left) Cutline drawn on 2D plot and (right) xy plot generated from cut



Manipulating Cutlines

Cutlines can be moved and resized by dragging the cutline handles (the circles at the ends of the cutline), and the cutline plot is updated automatically.

To delete a cutline, select the cutline and press the Delete key.

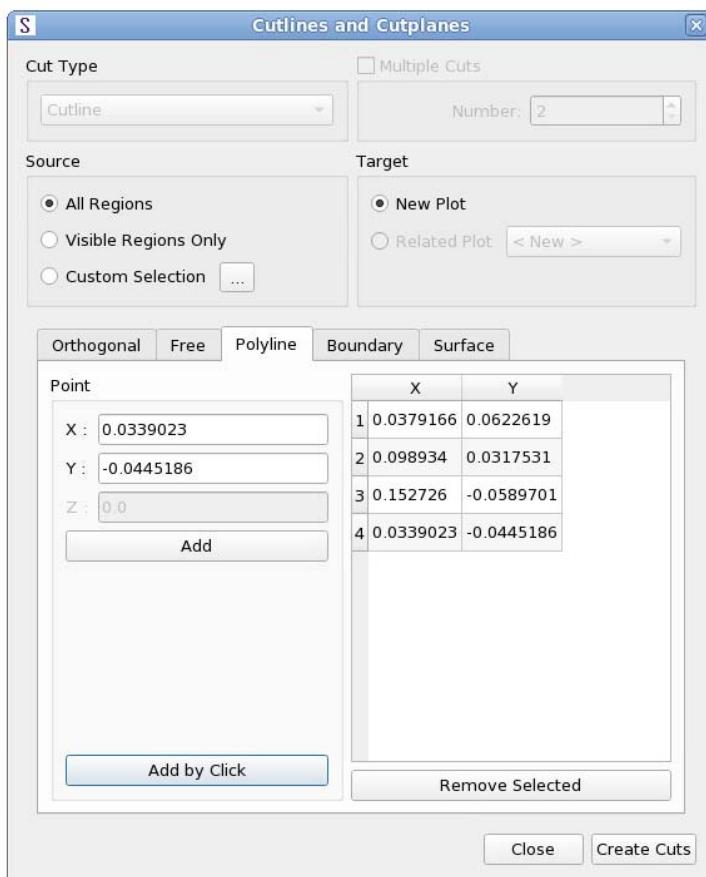
Note:

The xy plot created is not deleted. Deleting the xy plot does not delete the cutline in the 2D plot.

Polyline Cuts in 2D Plots

A polyline cut is the union of two or more cutlines where the end point of one is the start point of the other. Polyline cuts can be created selecting the **Polyline** tab in the Cutlines and Cutplanes dialog box (see [Figure 116](#)).

Figure 116 Polyline tab

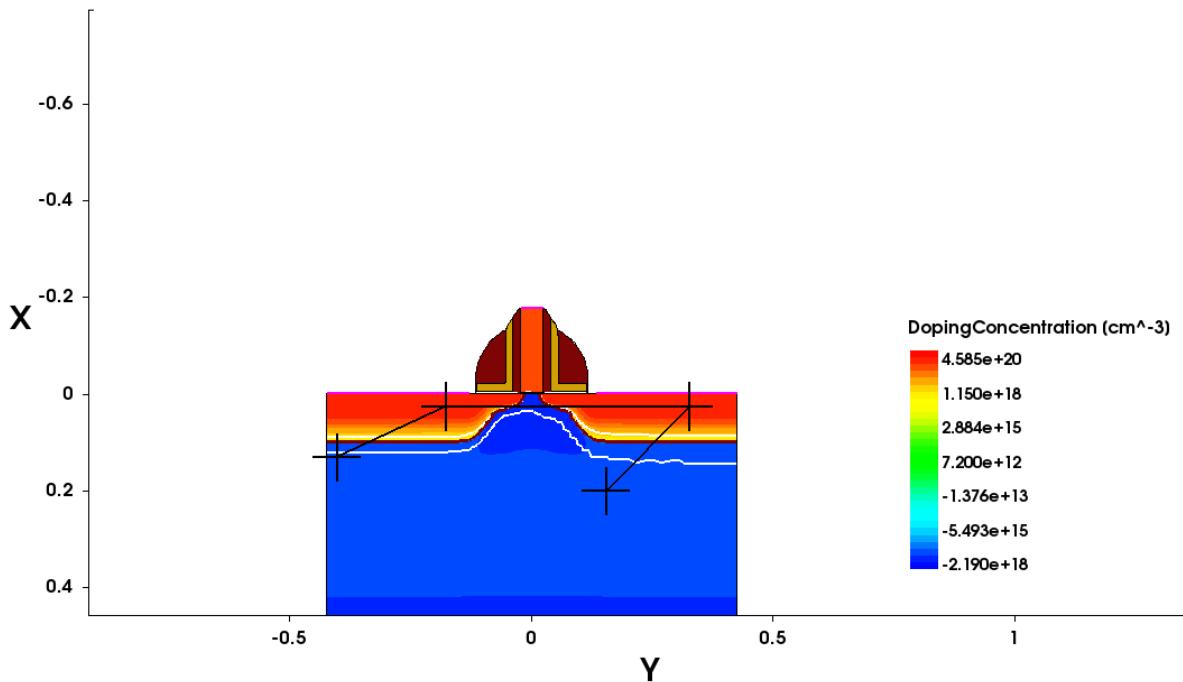


Chapter 4: Working With 2D and 3D Plots

Cutting Structures

The points that define the polyline can be added by using the keyboard, or using the fields in the Point group box, or clicking directly in the plot after clicking the **Start Add by Click** button. When a point is added, the position of the point is marked on the plot. For example, the points added in [Figure 116](#) will produce the marks shown in [Figure 117](#).

Figure 117 Marks indicate position of points generated from values in [Figure 116](#)

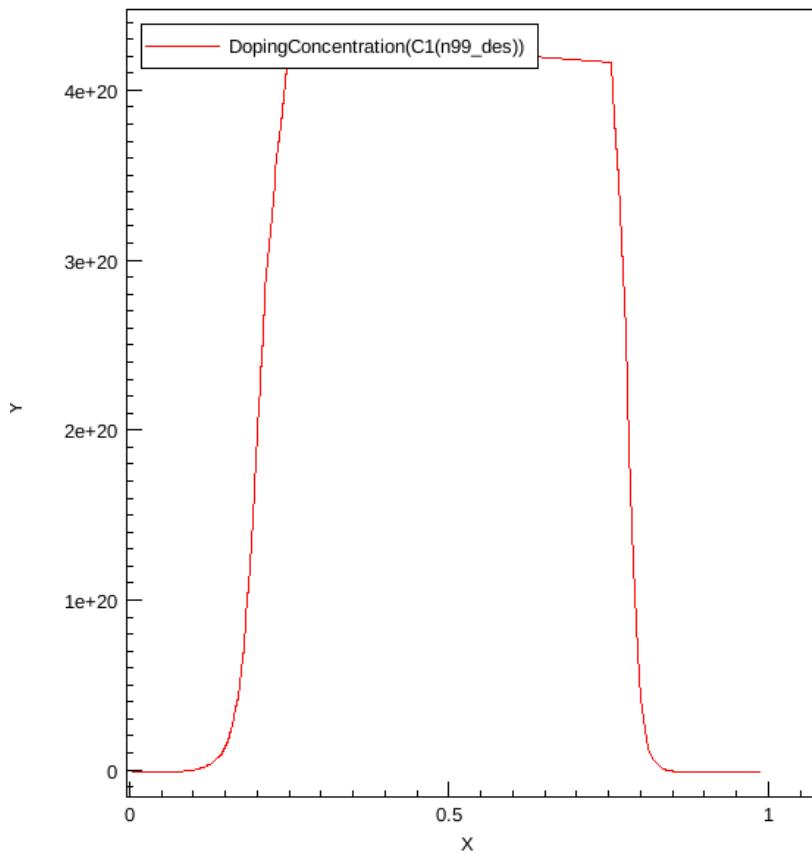


When all the points have been added, the cut is created by clicking **Create Cuts**. An xy plot is created immediately, showing the active field versus distance of the line. In addition, a new dataset is created containing the values along the line in all fields. [Figure 118](#) shows the plot created.

Chapter 4: Working With 2D and 3D Plots

Cutting Structures

Figure 118 New xy plot created from polyline cut



Manipulating the Polyline

The location of each point can be changed by dragging its handle to a new position. This will update the values of the already created xy plot using the new positions of the points. Other properties such as the line style, color, and size can be changed on the **CutPolyline Properties** tab of the Properties panel, when the polyline is selected.

Cutting Along Boundaries

You can cut along boundaries in 2D plots or a free cutplane by using the **Boundary** tab of the Cutlines and Cutplanes dialog box. Alternatively, use the `create_cut_boundary` command (see [create_cut_boundary on page 223](#)).

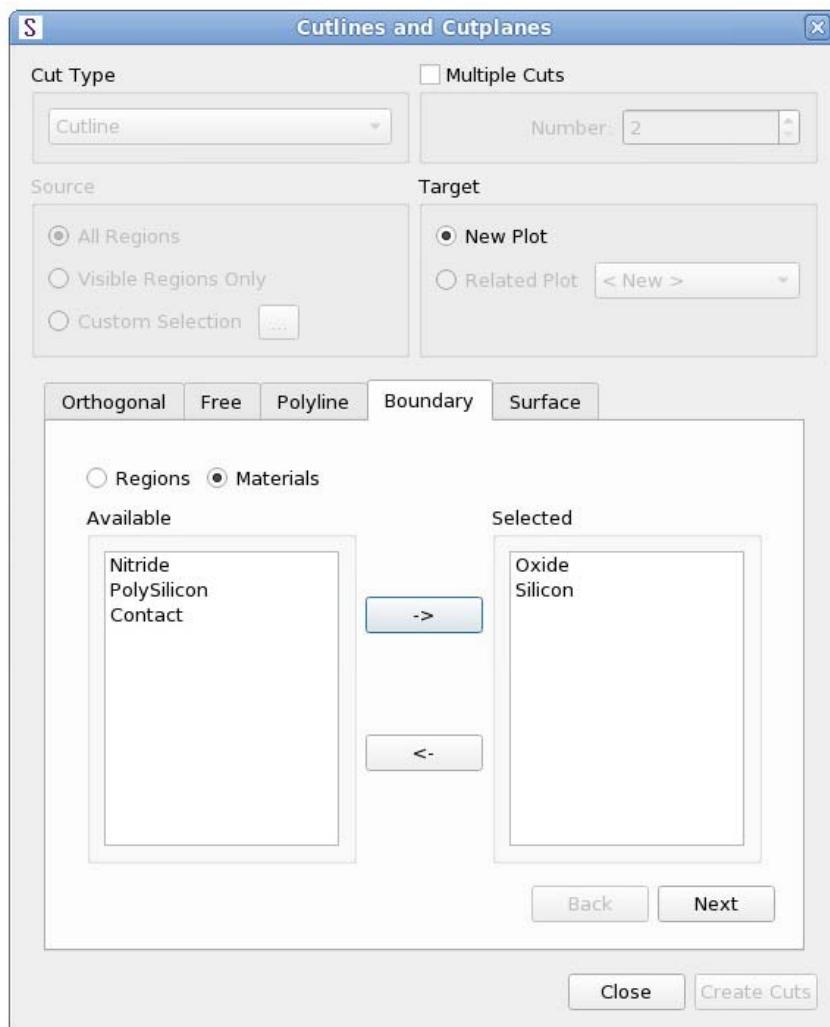
Step 1: Selecting Regions or Materials

Select the target regions or materials of interest. If you do not know which regions or materials you need, you can choose all of them and define them later.

To select regions or materials:

1. Select **Regions or Materials**.
2. Move the available regions or materials to the Selected pane as required.
3. When you are finished, click **Next**.

Figure 119 Step 1: selecting regions or materials



Step 2: Adding Vertex Points

Add the vertex points through which the line will pass. The first point and last point added will be the start and end of the line along the boundaries. The line will also pass through any middle point added in the respective order. If more than one path is possible, Sentaurus Visual will choose the shortest path along the available regions or materials selected in the past step.

To add points:

1. Add points in one of the following ways:

- Click **Start Add by Click** (which changes to the **End Add by Click** button). Click to add points inside the plot. When you have finished adding points, click **End Add by Click**.
- Use the Add Point fields to enter the x-, y-, and z-coordinates for a point. Click **Add Point**. The point is listed in the Selected Points pane. Continue to add points as required (see [Figure 120](#)).

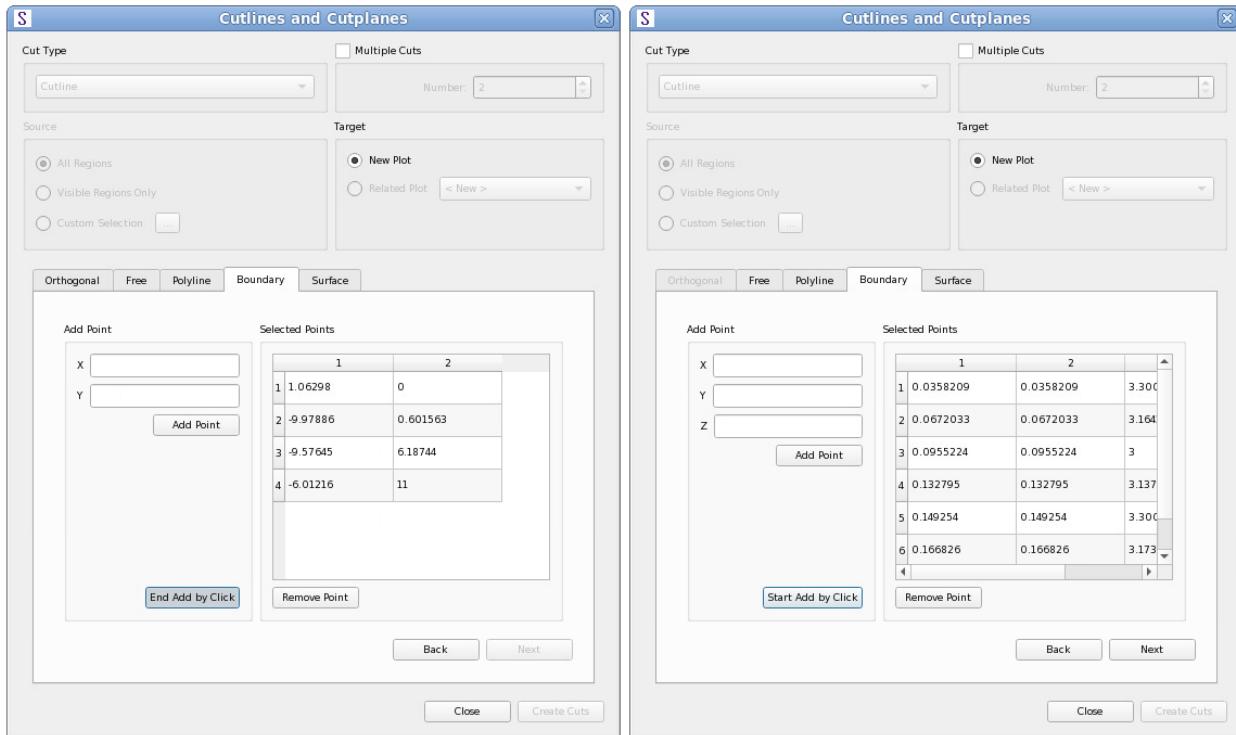
In both cases, if the point added is not on a boundary, Sentaurus Visual selects the nearest boundary to that point.

2. When you have finished adding points, click **Next**.

Chapter 4: Working With 2D and 3D Plots

Cutting Structures

Figure 120 Step 2: adding vertex points in (left) 2D plot and (right) free cutplane



Step 3: Choosing Segment Regions

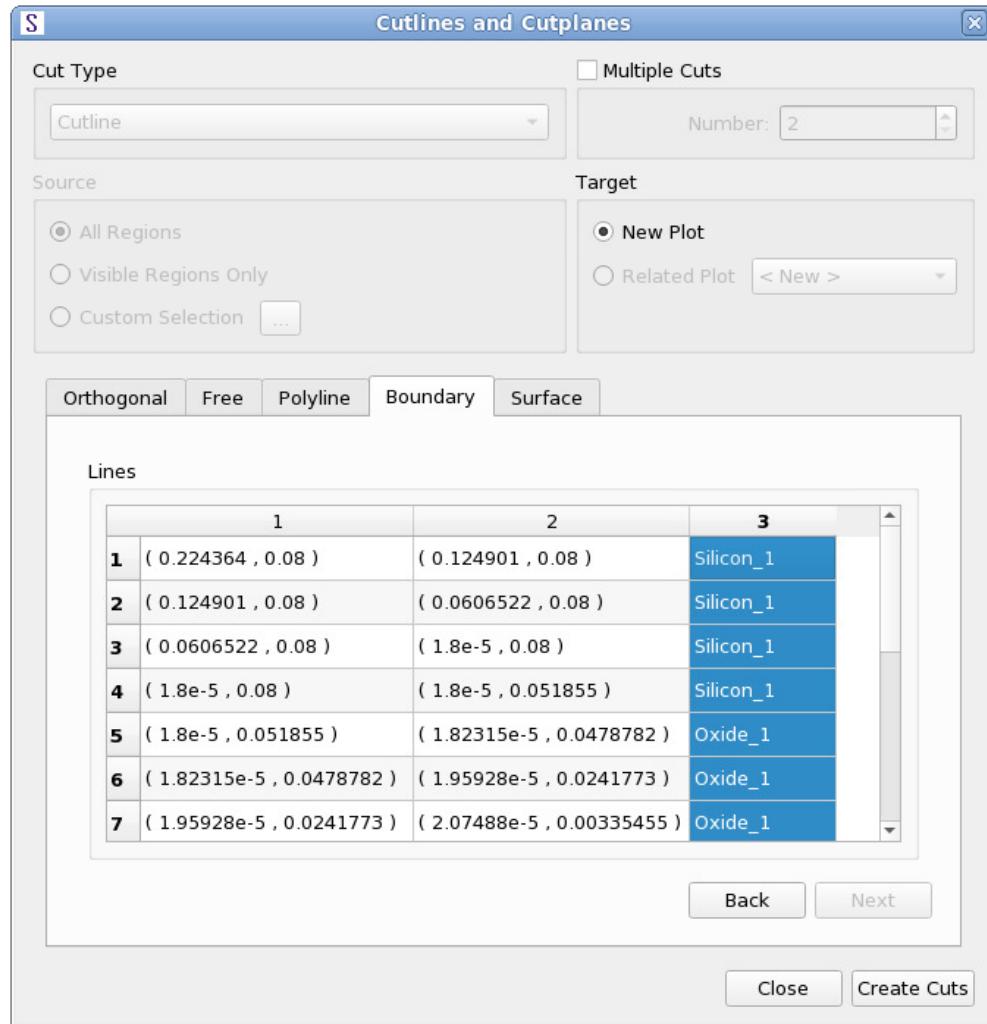
After you have added the vertex points, Sentaurus Visual divides the resulting line into various segments defined by the intersections of neighboring regions. In each segment, the required region from which the data will be extracted can be chosen by clicking the cell in the Region column (column 3), as shown in [Figure 121](#).

When all the regions in each section are selected, the cut can be created by clicking **Create Cuts**. The default segment regions are chosen by the order of regions or materials previously set.

Chapter 4: Working With 2D and 3D Plots

Cutting Structures

Figure 121 Step 3: choosing segment regions

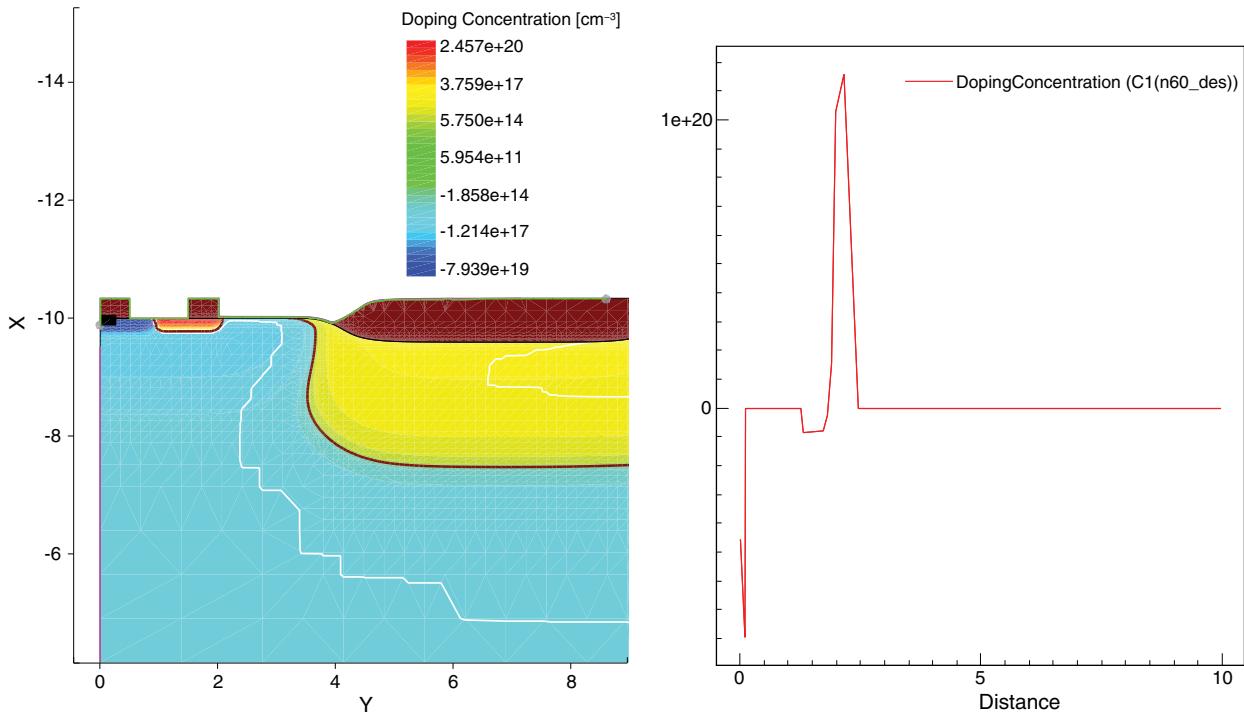


The resulting plot shows the values along the selected regions of the active field versus distance. In addition, a dataset containing all the respective fields is created (see [Figure 122](#)).

Chapter 4: Working With 2D and 3D Plots

Cutting Structures

Figure 122 (Left) Original 2D plot and (right) resulting xy plot from cutting along boundaries



Surface Cutlines From 3D Plots

You can extract data defined on the surface of a structure along an axis-aligned cutplane. The extraction generates 1D data, where a curve is displayed automatically with the active scalar field in the source plot.

Note:

This feature is available only for particle Monte Carlo (PMC) structures generated by Sentaurus Topography 3D.

To create a surface cutline from a PMC structure:

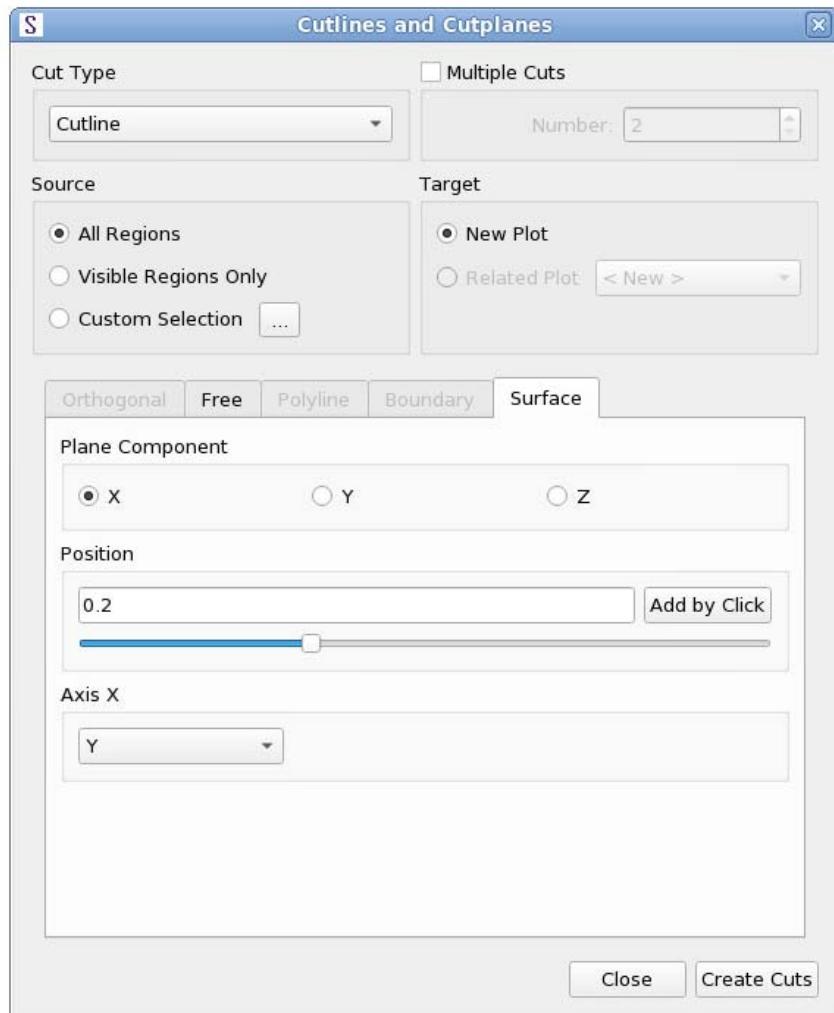
1. On the **Surface** tab of the Cutlines and Cutplanes dialog box, under Plane Component, select the axis to which the cutplane should be aligned (see [Figure 123 on page 185](#)).
2. Under Position, specify where the cutline should be located.
3. Under Axis X, select the component variable to use in the abscissa (x-axis) for xy plots.
4. Click **Create Cuts**.

Alternatively, you can use the `create_cutline` command (see [create_cutline on page 224](#)).

Chapter 4: Working With 2D and 3D Plots

Cutting Structures

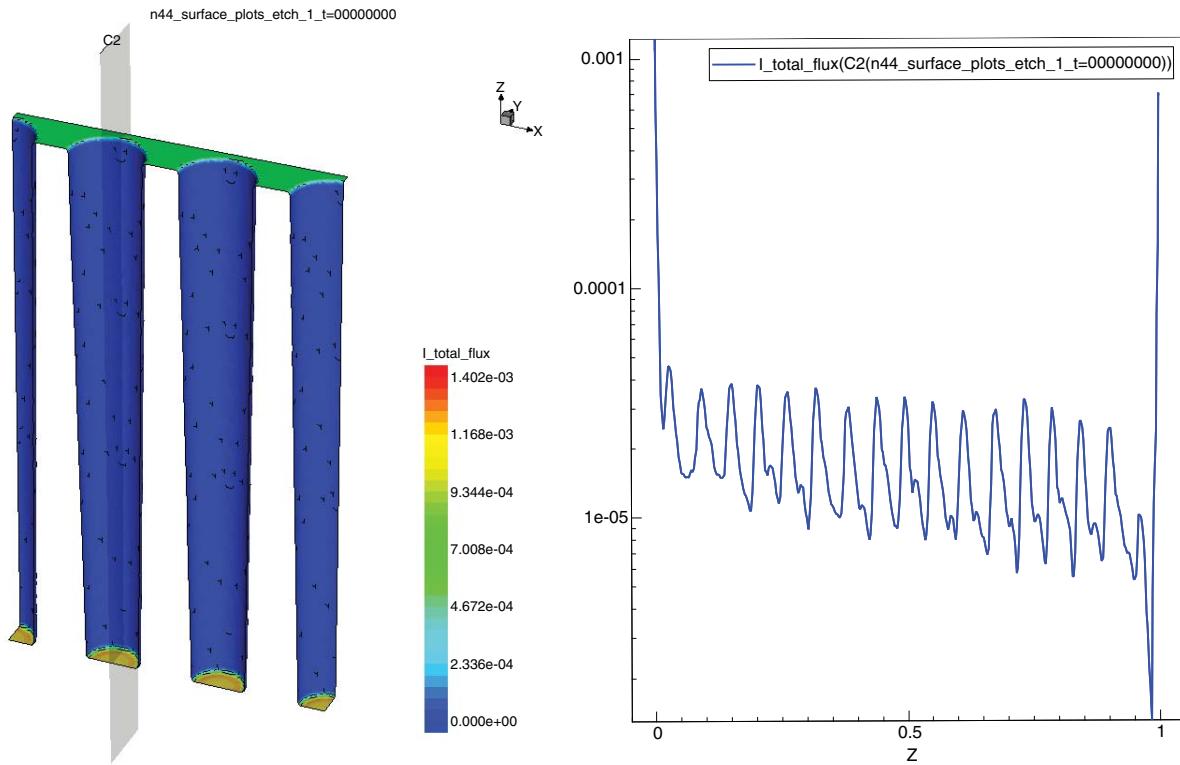
Figure 123 Surface tab



Chapter 4: Working With 2D and 3D Plots

Cutting Structures

Figure 124 (Left) Cutplane on the surface of a PMC structure and (right) xy plot generated from the surface cut



Changing Properties of Cutline Along Boundaries

Some properties such as the color, size, and type of the line can be changed in the **Cutline Properties** tab, which is displayed when the line is selected in the plot. The color and visibility of the handles of the first and last points can be changed as well.

Two-Dimensional Projections

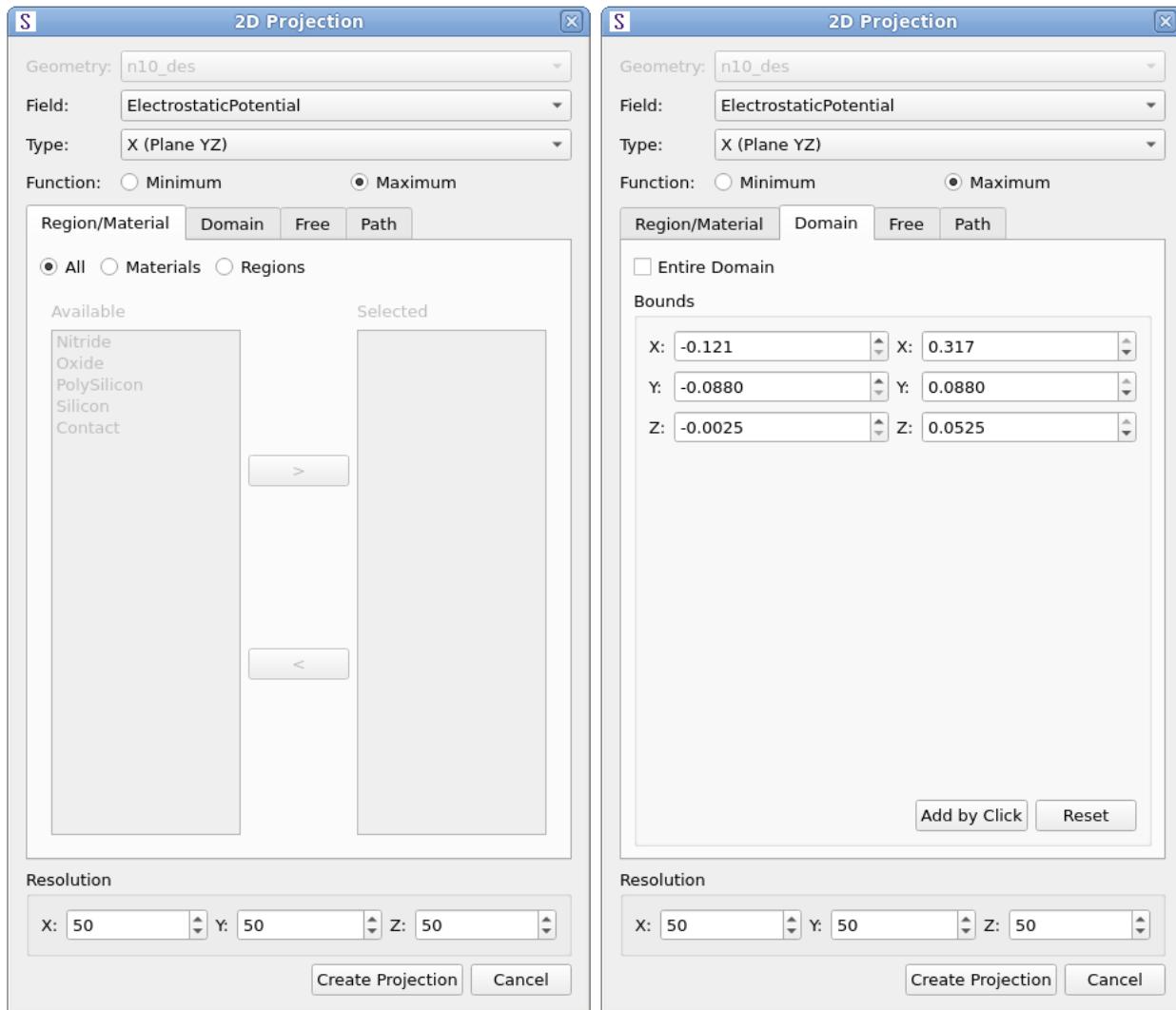
Two-dimensional projections can be obtained from 3D plots. The resulting plot is either the maximum or minimum field value projected to one plane, which can be aligned to the orthogonal axes or arbitrarily defined. In addition, you can define a polyline path to create a sequence of projected planes.

You can create a 2D projection of a 3D plot by either using the GUI (choose **Tools > Create Projection**) or using the command `create_projection` or `create_projection_path` (see [create_projection on page 232](#) and [create_projection_path on page 234](#)).

Chapter 4: Working With 2D and 3D Plots

Cutting Structures

Figure 125 2D Projection dialog box showing (left) Region/Material tab and (right) Domain tab



To create a 2D projection of a 3D plot, performed on all regions, or selected regions or materials:

1. Choose **Tools > Create Projection**.

The 2D Projection dialog box opens.

2. From the **Type** list, select the plane for the projection.

This functionality is valid only for projection types **X**, **Y**, and **Z**.

3. Select the function from either **Minimum** or **Maximum**.

Chapter 4: Working With 2D and 3D Plots

Cutting Structures

4. On the **Region/Material** tab, select whether you want to perform the projection on all regions or selected regions or materials (see [Figure 125 on page 187 \(left\)](#)).

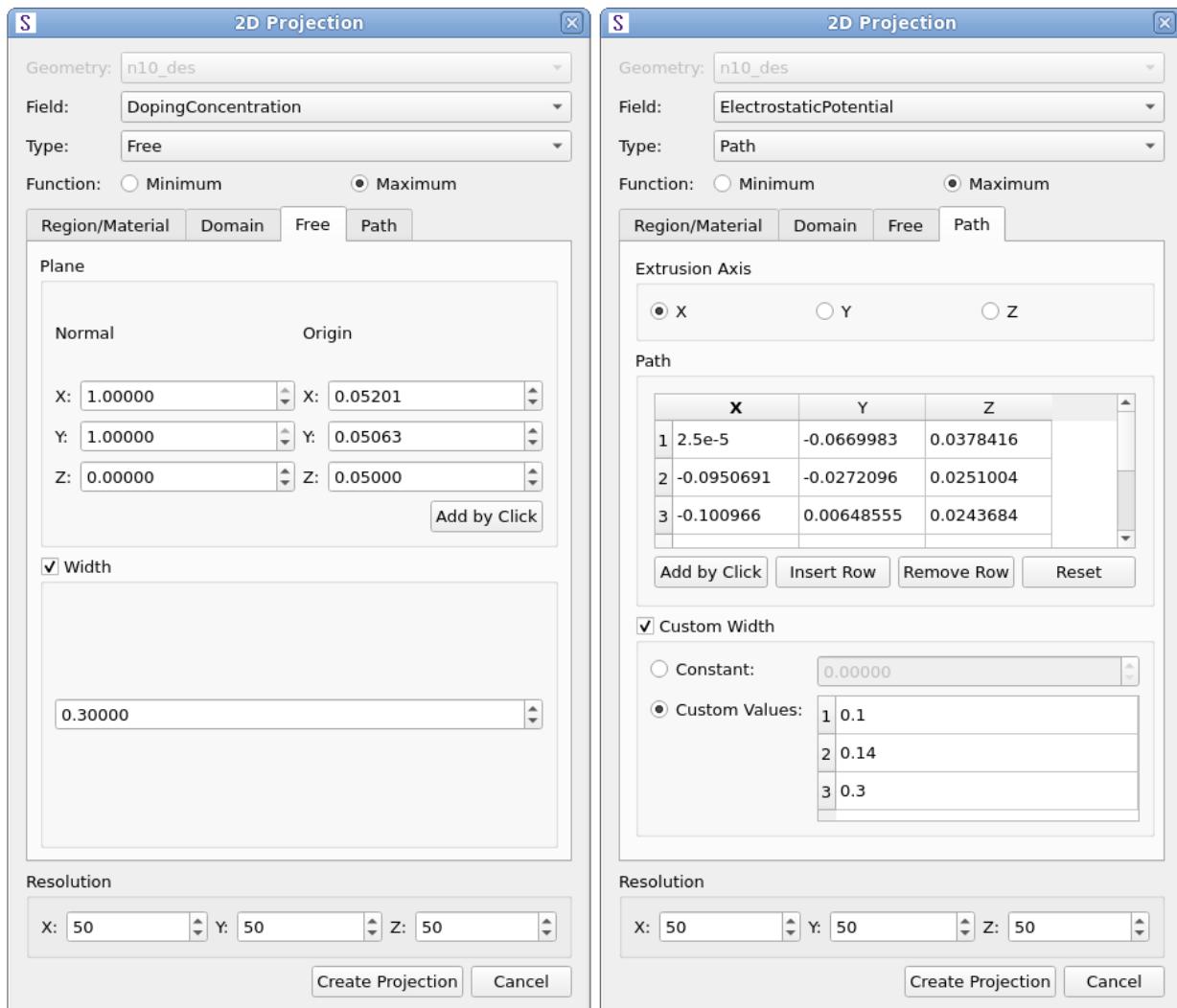
5. Under Resolution, specify the number of points to consider on each axis.

Greater resolution means more accurate data extraction, but with a longer processing time.

6. When are finished, click **Create Projection**.

The projection can be performed in all domains or in a smaller window defined on the **Domain** tab as shown in [Figure 125 \(right\)](#). This functionality is valid only for projection types **X**, **Y**, and **Z**.

Figure 126 2D Projection dialog box showing (left) Free tab and (right) Path tab



Chapter 4: Working With 2D and 3D Plots

Cutting Structures

The projection can be performed in arbitrary planes defined by you. You must select **Free** from the **Type** list. The **Free** tab becomes available and allows you to define the origin and normal for the plane (see [Figure 126 on page 188 \(left\)](#)). If you select **Width**, then it restricts the domain considered for the projection to all the points between a half-width distance to the plane.

To define a path across the plot to perform a projection:

1. From the **Type** list, select **Path**.

The **Path** tab becomes available (see [Figure 126 \(right\)](#)).

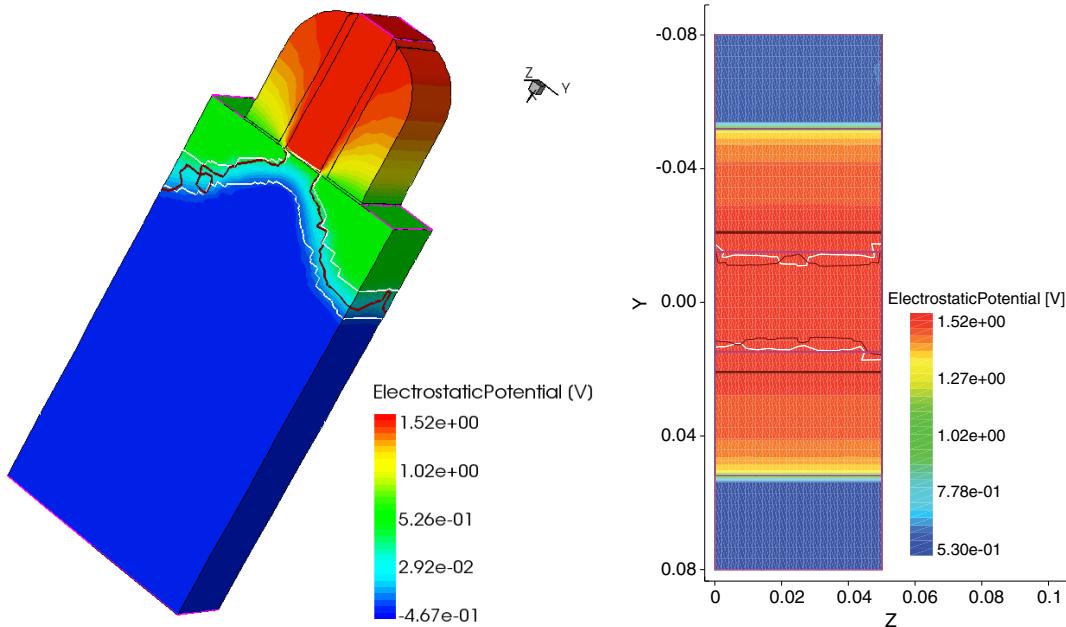
2. Select the **Extrusion Axis** (vertical axis in the resulting 2D plot).
3. Define the points manually or use the **Add by Click** button.
4. If a user-defined width is needed, then select **Custom Width** to allow you to set either the same width for all planes (**Constant**) or individual widths for each plane (**Custom Values**).

Note that the number of widths equals the number of planes to be generated. The number of planes to be generated equals the number of points previously defined minus one. At least two points are required to perform this type of analysis.

5. After you have defined the variables, click **Create Projection**.

[Figure 127](#) shows the final plot for the maximum value of a *yz* projection.

Figure 127 (Left) Original 3D plot and (right) 2D projection of the 3D plot

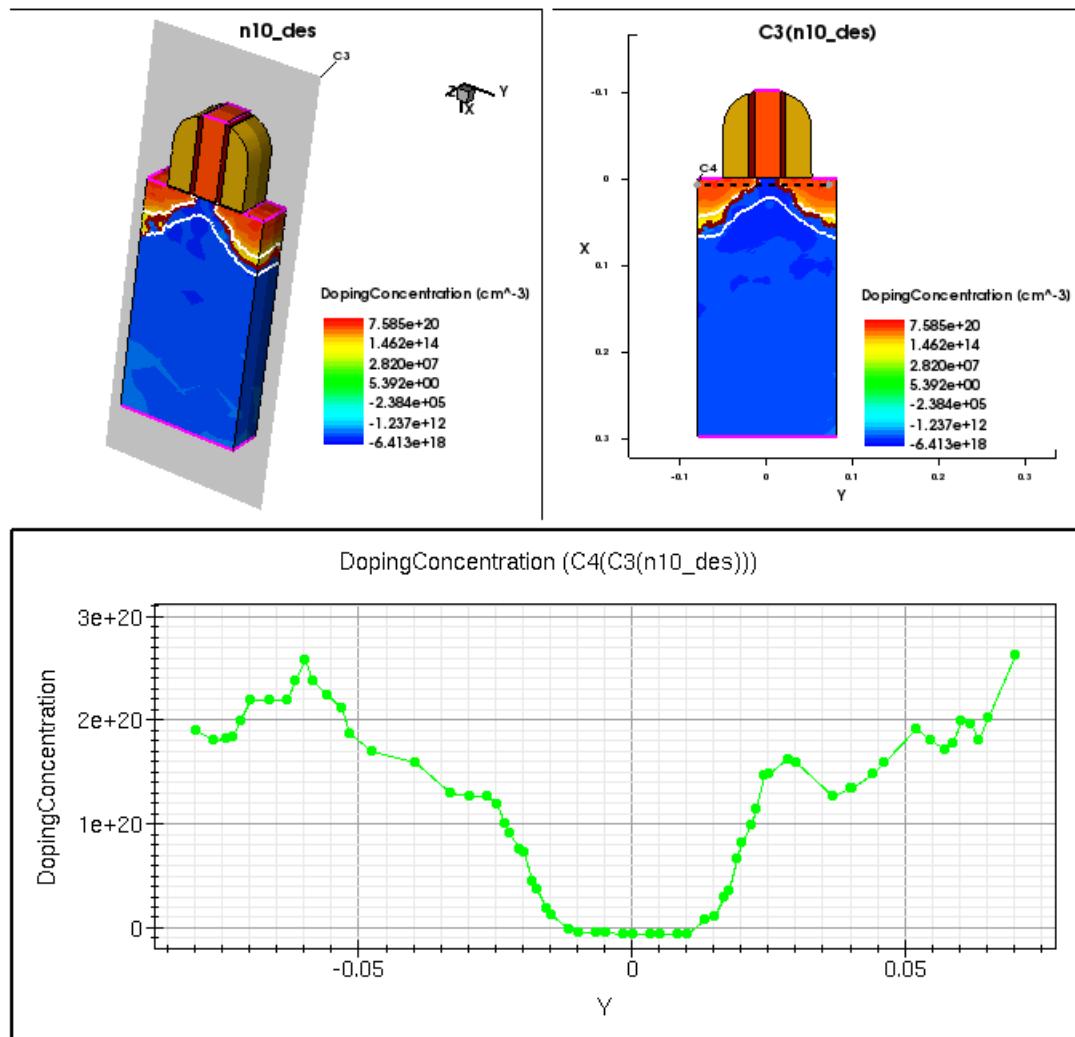


Cutplanes in 3D Plots

In 3D plots, orthogonal cutplanes can be created by selecting a cut axis and then clicking the required point of the plot. The result is a new 2D plot with the same fields as the original plot as seen in [Figure 128](#). Such a 2D plot can be cut further by a cutline to generate an xy cut.

Cutplanes also can be moved by dragging, and the 2D plot is updated automatically. In addition, xy plots created from such 2D cuts are updated automatically.

Figure 128 Cutplane in a 3D plot and the generated 2D plot, which is cut further to generate an xy plot



Chapter 4: Working With 2D and 3D Plots

Extracting the Path of Minimum or Maximum Values of a Scalar Field

To delete a cutplane, select the cutplane and press the Delete key.

Note:

Deleting the 2D plot does not delete the cutplane in the 3D plot.

The mesh shown on the cutplane is recalculated by triangulating the resulting points of the cut, which means, for example, that an axis-aligned cut of a rectangular mesh shows a triangular mesh.

Extracting the Path of Minimum or Maximum Values of a Scalar Field

You can extract the path of either the minimum or maximum values of a specified scalar field.

You can use a Tcl or Python command to extract the path (see [extract_path on page 251](#)) or the corresponding dialog box, which is available from **Tools > Extract Path**.

The Extract Path dialog box has different modes of operation that use different algorithms for extracting the path:

- The 2D mode applies to 2D plots and its algorithm extracts the minimum or maximum values of a specified scalar field along the horizontal axis. This extraction does not refer to a specific axis, so interchanging the x-axis and y-axis of a 2D plot generates different results.

The 2D algorithm recalculates the mesh for the 2D structure, normalizing the mesh to the smallest cell width in the horizontal direction. However, if this recalculation exceeds millions of divisions, Sentaurus Visual resolves this to one million divisions to maintain tool performance (see [Figure 129 on page 196](#)).

- The 3D mode applies to 3D plots and its algorithm extracts the minimum or maximum values of a specified scalar field along the mesh of the structure. This algorithm does not depend on the orientation of the structure.

The 3D algorithm analyzes the existing mesh to find the minimum or maximum path between two points that can be constructed using the mesh elements.

To extract a path:

1. Choose **Tools > Extract Path**.

The Extract Path dialog box opens.

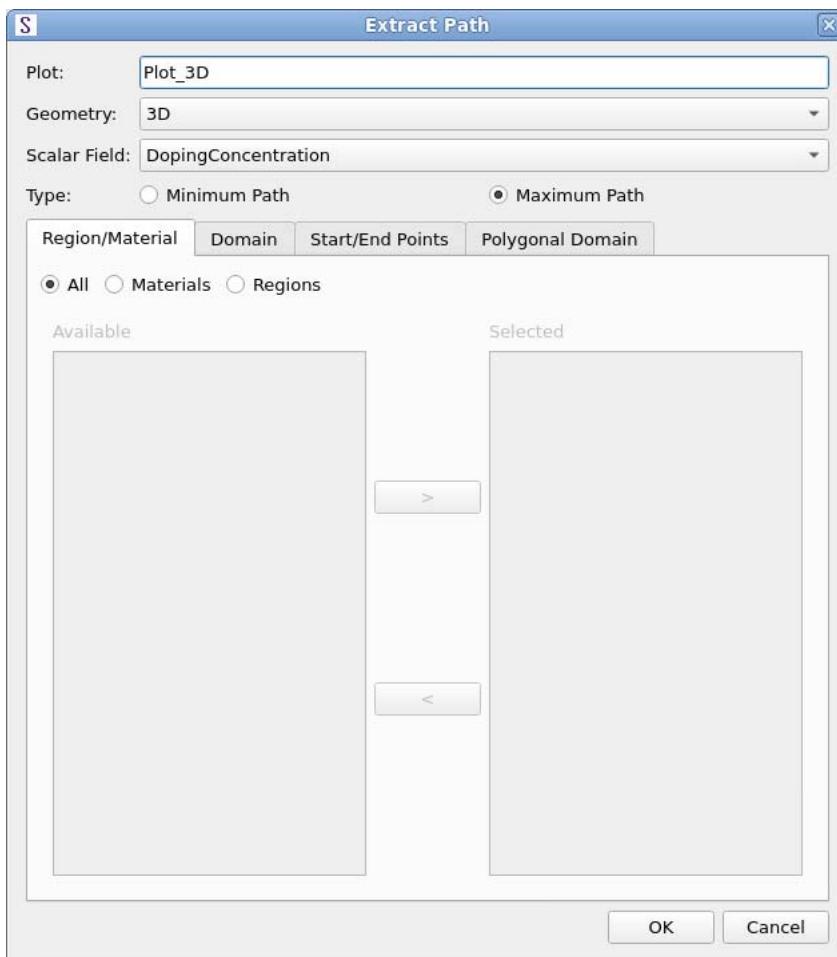
2. Leave the plot name in the **Plot** field. This is the name of the active plot.
3. Select the geometry from the **Geometry** list.
4. Select the scalar field for extraction from the **Scalar Field** list.

Chapter 4: Working With 2D and 3D Plots

Extracting the Path of Minimum or Maximum Values of a Scalar Field

5. Select which values you want to extract: **Minimum Path** or **Maximum Path**.
6. On the **Region/Material** tab, select one of the following:
 - **All**: To extract the path over all materials and regions
 - **Materials**: To extract the path over selected materials
 - **Regions**: To extract the path over selected regions

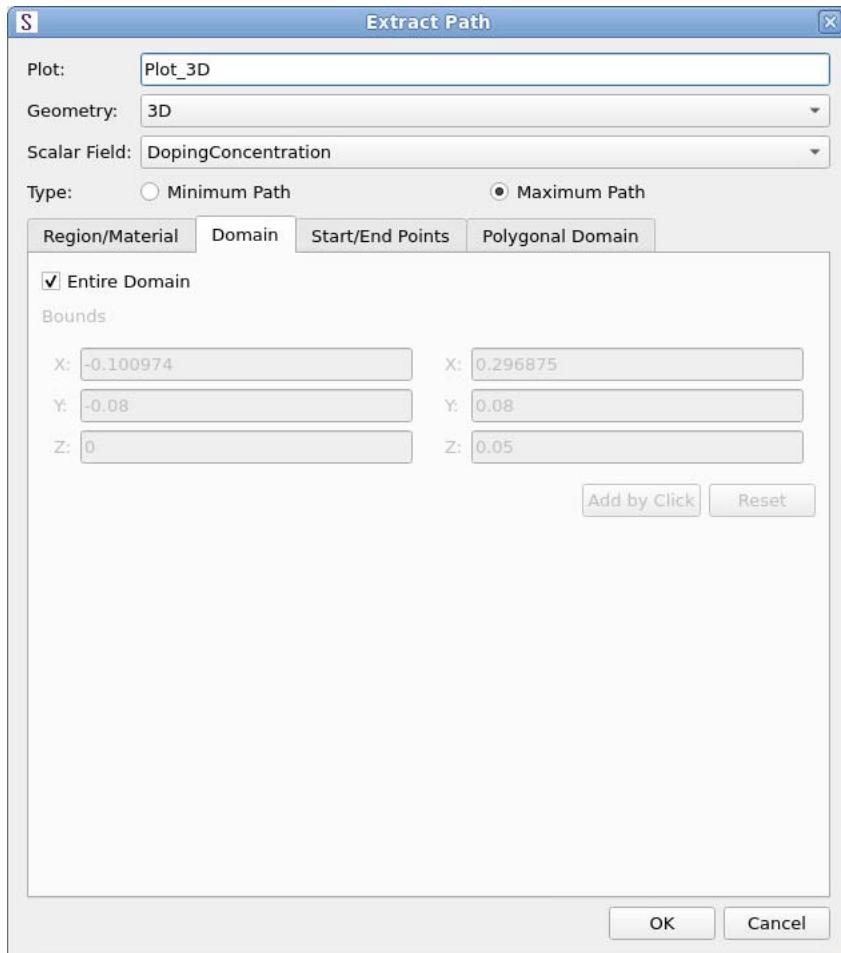
The materials and regions shown in the **Available** pane depend on which ones are present in the plot.



7. (Optional) For greater precision, click the **Domain** tab and specify the start and end points of a smaller window of analysis.

Chapter 4: Working With 2D and 3D Plots

Extracting the Path of Minimum or Maximum Values of a Scalar Field

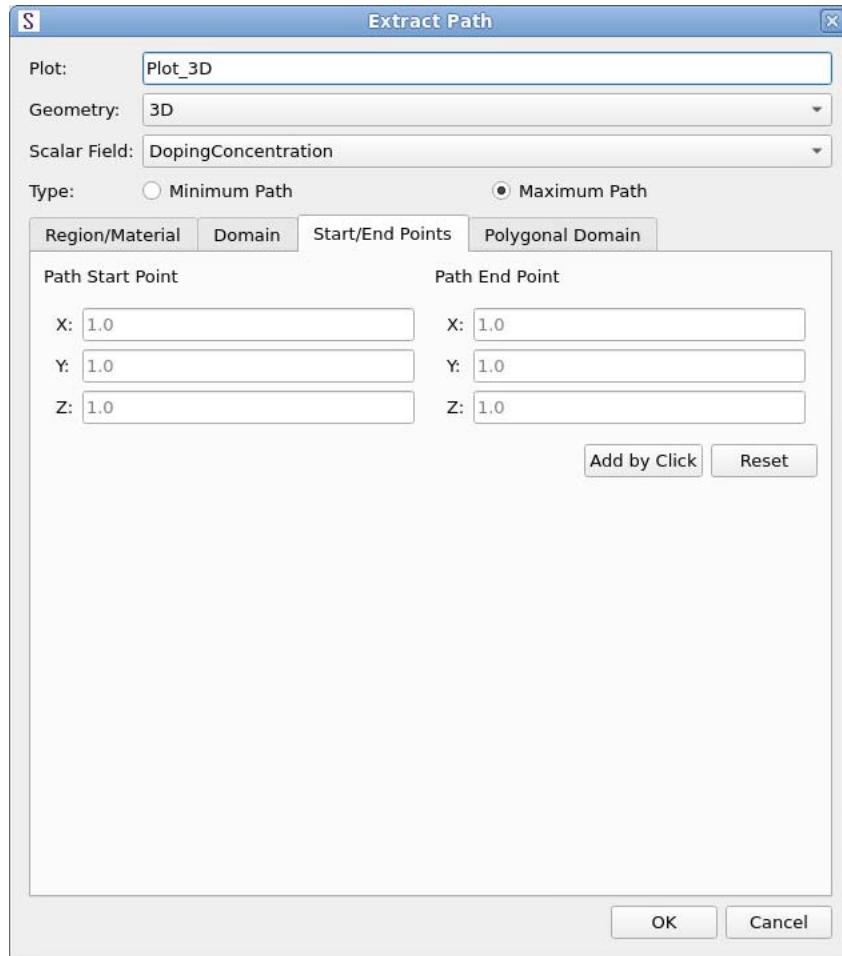


8. If the plot is three dimensional, then the **Start/End Points** tab and **Polygonal Domain** tab are available.

Chapter 4: Working With 2D and 3D Plots

Extracting the Path of Minimum or Maximum Values of a Scalar Field

Specify the start and end points of the path on the **Start/End Points** tab.

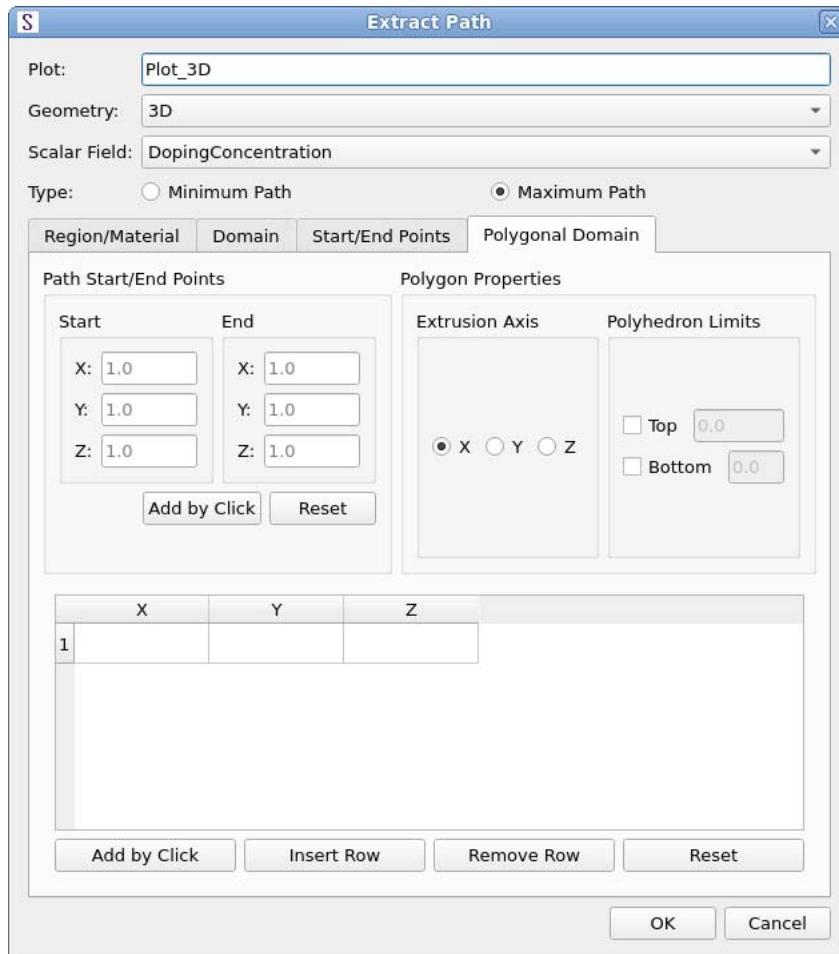


9. The domain of the path can be bounded by a polyhedron that results from extruding a polygon drawn in the 3D model through any of the x-, y-, or z-axis. This can be configured on the **Polygonal Domain** tab.

Chapter 4: Working With 2D and 3D Plots

Extracting the Path of Minimum or Maximum Values of a Scalar Field

Specify the start and end points, the polygon points, and the extrusion axis. Optionally, the top and bottom faces can be set if the start or end points are inside the model.

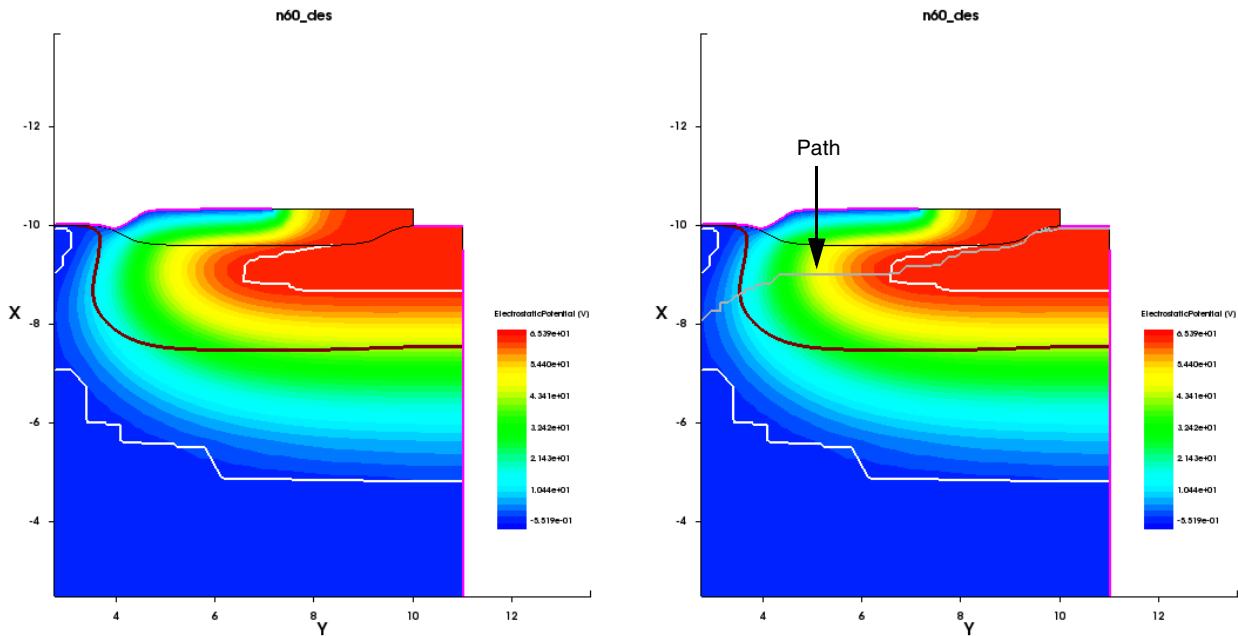


10. Click **OK**.

Chapter 4: Working With 2D and 3D Plots

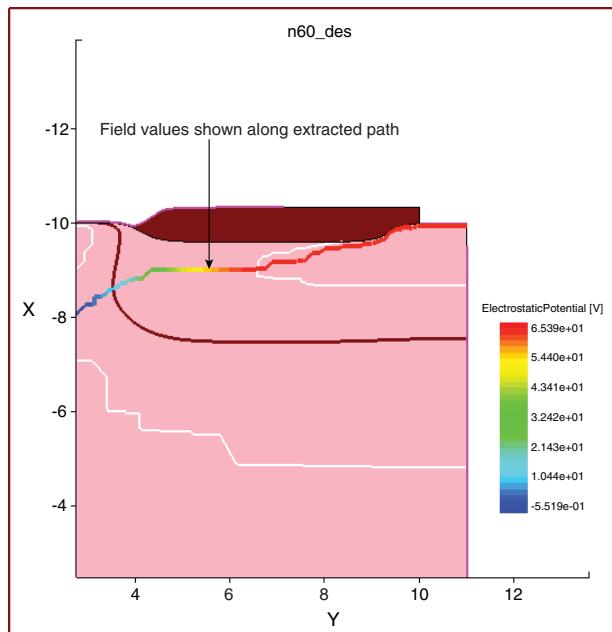
Extracting the Path of Minimum or Maximum Values of a Scalar Field

Figure 129 (Left) Original 2D structure and (right) extracted path over the entire 2D structure



The extraction results in the creation of a new geometry in the structure that behaves like an interface region, allowing you to visualize the field values in the path, even if the main geometry does not display field data (see [Figure 130](#)).

Figure 130 Field values along the extracted path, with main geometry displaying no field data



Chapter 4: Working With 2D and 3D Plots

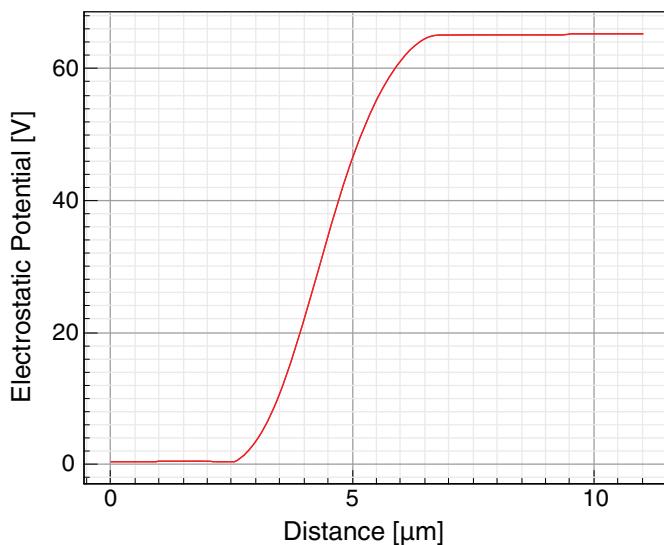
Surface Plots

In addition, in 3D mode, the new geometry has an extra field that represents the saddle points of the extracted path. See [find_values on page 256](#) for more information about how to obtain the exact position of the saddle points.

See [Visualizing Fields on page 107](#) for more information about visualizing scalar fields.

When you extract a path using the Extract Path dialog box, which is not a *cutting* operation, in addition to the path shown in the plot, a regular xy plot is generated showing a curve of the extracted path field (see [Figure 131](#)). This xy plot is generated using the `create_curve` and `create_plot` commands.

Figure 131 Extracted path displayed as an xy plot; curve represents the banded field (ElectrostaticPotential)



Surface Plots

A surface plot is a 3D plot generated from a 2D dataset (or plot), where the constant component (the z-axis if the original structure plane is the xy plane) is filled with an existing scalar field. As the new dataset contains the three components (x-axis, y-axis, and z-axis) defined, it behaves as a typical 3D dataset. If a 3D dataset is created, it will be shown as a new plot.

The new 3D dataset will contain the same regions and fields as the source dataset, which can be independent of the source plot. In addition, Sentaurus Visual recalculates the junction line and the depletion region, to show them according to the new surface.

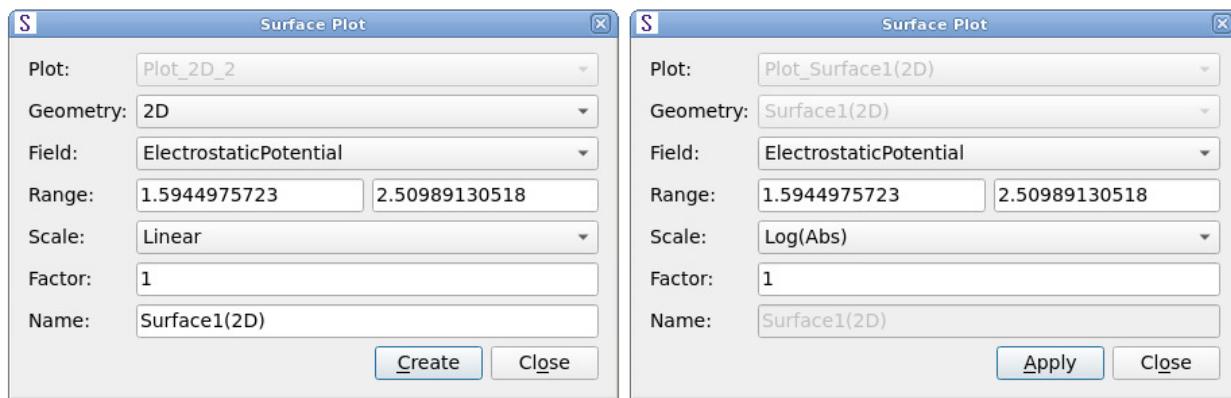
The created 3D surface plot inherits the current field with filled contour bands and the visibility options of all regions. This means that the surface plot hides regions that are not shown in the source plot.

Creating Surface Plots

You can create a surface plot using either the Surface Plot dialog box (choose **Tools > Surface Plot**) or the `create_surface` command (see [create_surface on page 239](#)).

In the Surface Plot dialog box, you can choose the geometry of the plot (if there is more than one), the field to be used, the range, the scaling type, the factor to define how the values of the field will affect the constant component, and the name of the new dataset generated.

Figure 132 Surface Plot dialog box: (left) creating a surface plot and (right) modifying the newly created surface plot

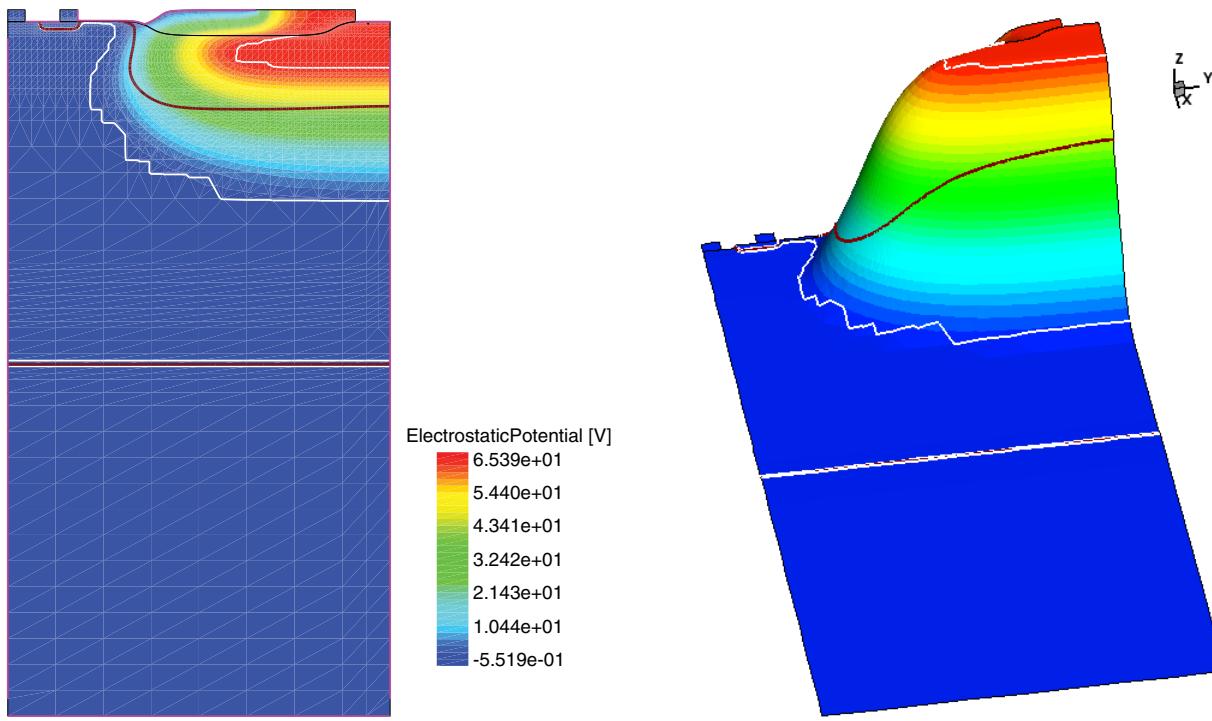


After the new surface plot is created, the dialog box does not close. It changes appearance to allow you to modify the last generated surface plot (see [Figure 132, right](#)). Some fields are deactivated. The remaining fields can be changed to fine-tune the surface plot. With each change that is applied, the plot is updated.

Chapter 4: Working With 2D and 3D Plots

Isosurfaces and Isolines

Figure 133 (Left) Two-dimensional source plot and (right) generated surface plot using the ElectrostaticPotential field



Isosurfaces and Isolines

Sentaurus Visual can extract isosurfaces and isolines from 3D and 2D structures, respectively. These iso-geometries are extracted using a constant value (isovalue) over a specified field and structure. The extracted iso-geometry is displayed in the same plot as the source geometry, such as an overlay plot. The new iso-geometry is divided into different regions and contains the same fields as the source geometry. This means that it is possible to display the same or different fields in both geometries. By default, the new iso-geometry has a constant color to help identify it easily. A plot can contain as many iso-geometries as you want.

The new iso-geometry does not contain any line or particle region from the source geometry and contains only the regions that have values of the specified field.

Creating Iso-Geometries

You can create a new iso-geometry using either the Create Isovalue Geometry dialog box or the `create_iso` command (see [create_iso on page 230](#)).

Chapter 4: Working With 2D and 3D Plots

Isosurfaces and Isolines

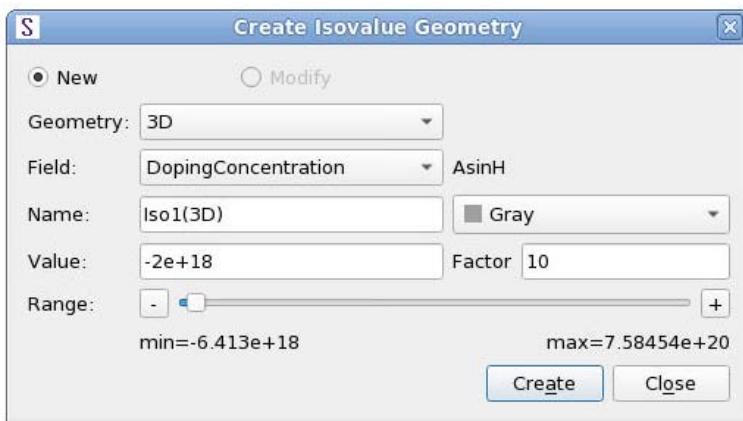
Note:

After you create an iso-geometry, the Create Isovalue Geometry dialog box remains open and changes automatically to the modification mode, so that you can modify the iso-geometry if required (see [Modifying Iso-Geometries on page 201](#)).

To create a new iso-geometry using the Create Isovalue Geometry dialog box:

1. Choose **Tools > Create Isovalue**.

The Create Isovalue Geometry dialog box opens. The **New** option is selected by default.



2. Select the geometry of the plot (if there is more than one).
3. Select the field to be used.
4. Enter the name of the new iso-geometry (or dataset).
5. Select the color with which to display the new iso-geometry.
6. In the **Value** field, enter the isovalue to use to build the iso-geometry.
7. In the **Factor** field, enter a factor.

For nonlinear fields, the **Factor** field defines how the slider value changes between steps. By default, this value is 10. For linear fields, the Delta value defines the size between steps.

8. Use the **Range** slider or the **-** and **+** buttons to identify where the value lies in the range of the field.

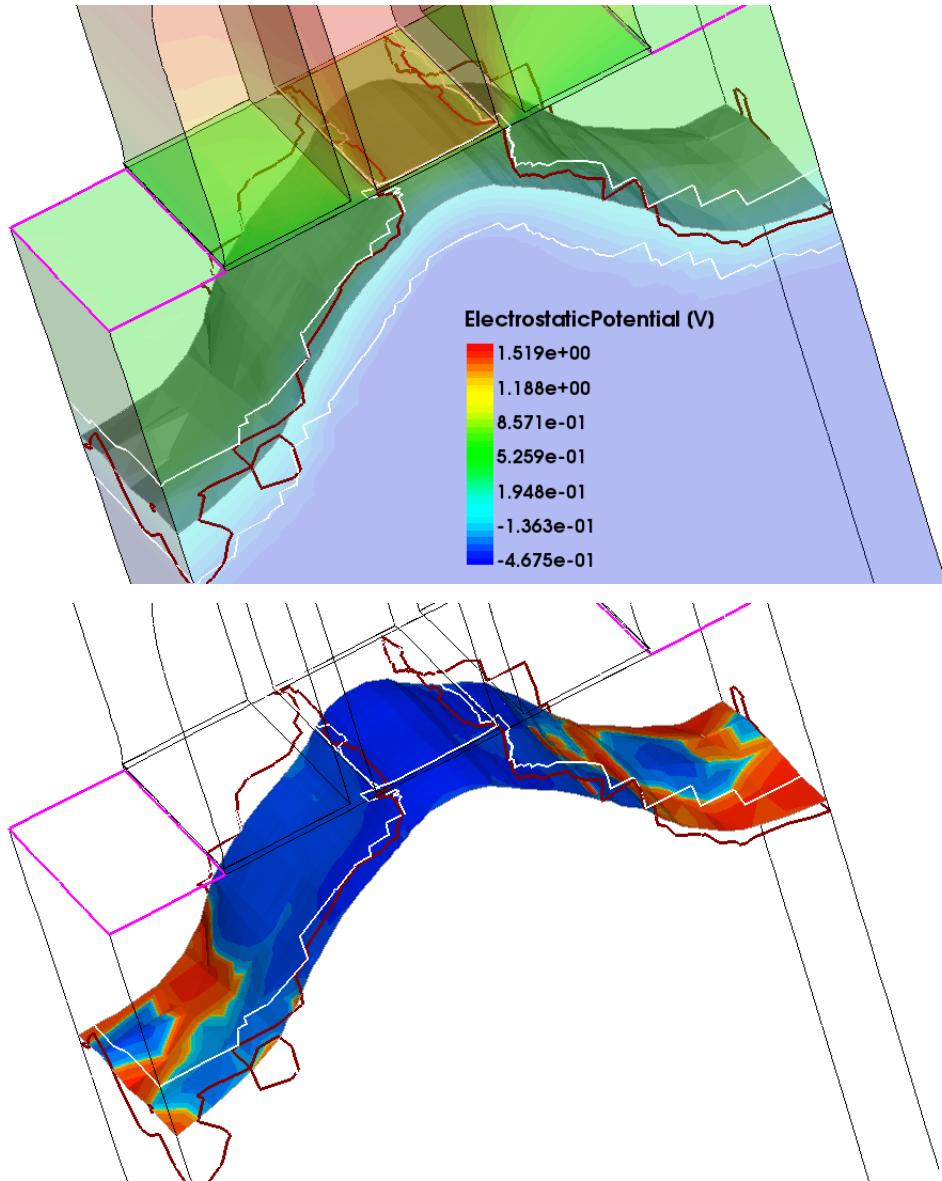
The slider responds according to the selected field scale, shown to the right of the **Field** list.

9. Click **Create**.

Chapter 4: Working With 2D and 3D Plots

Isosurfaces and Isolines

Figure 134 (Top) Source geometry with translucency showing the new iso-geometry generated (*ElectrostaticPotential* = 0 V) and (bottom) iso-geometry displaying the *DopingConcentration* field



Modifying Iso-Geometries

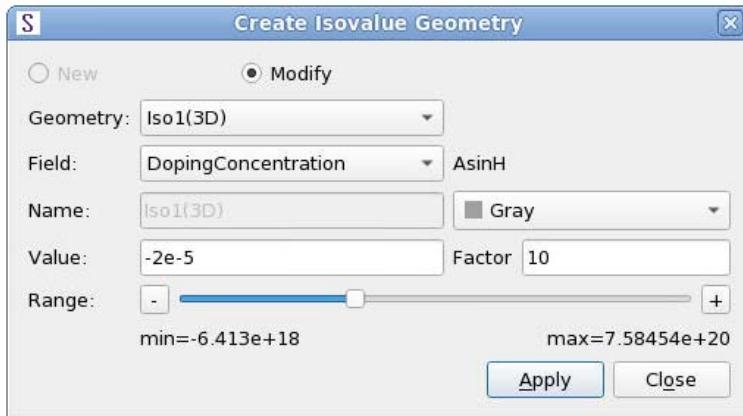
You can modify iso-geometries using either the Create Isovalue Geometry dialog box (see [Creating Iso-Geometries on page 199](#)) or the `create_iso` command (see [create_iso on page 230](#)).

Chapter 4: Working With 2D and 3D Plots

Streamlines

To modify an iso-geometry using the Create Isovalue Geometry dialog box:

1. Ensure the **Modify** option is selected.



2. Change the fields as required.

Note:

You can change all the fields except the name of the iso-geometry.

3. Click **Apply**.

Streamlines

Streamlines are a family of curves that are instantaneously tangent to the velocity vector of the flow. Sentaurus Visual allows you to visualize these streamlines for the available vector fields in 2D or 3D plots.

Streamlines are created only in the active plot by default, even if the active plot is part of a linked plot group. This is mainly because the velocity vector might not be present in all plots belonging to the group and the extensibility of the `create_streamlines`, `extract_streamlines`, and `set_streamlines_prop` commands cannot be ensured. However, if you want to apply streamlines to a plot group, you must create a special linked plot group to allow streamlines for that plot group (see [Linking Plots on page 51](#) and [link_plots on page 298](#)).

Note:

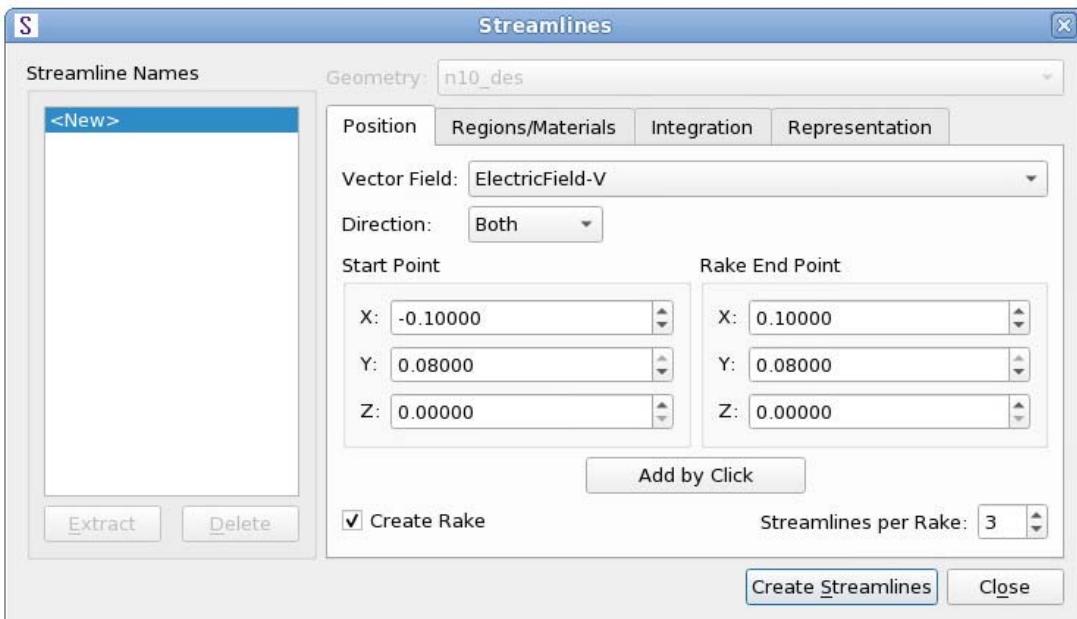
If plot group members do not have similar data, then the results might be unexpected.

Displaying Streamlines

Click the  toolbar button to display the Streamlines dialog box (see [Figure 135](#)), where you can select the vector field, the starting point, and the display properties.

In this dialog box, several properties can be defined to customize the display of the streamlines.

Figure 135 Streamlines dialog box showing Position tab, before creating streamlines



Position Tab

The fields of the **Position** tab are:

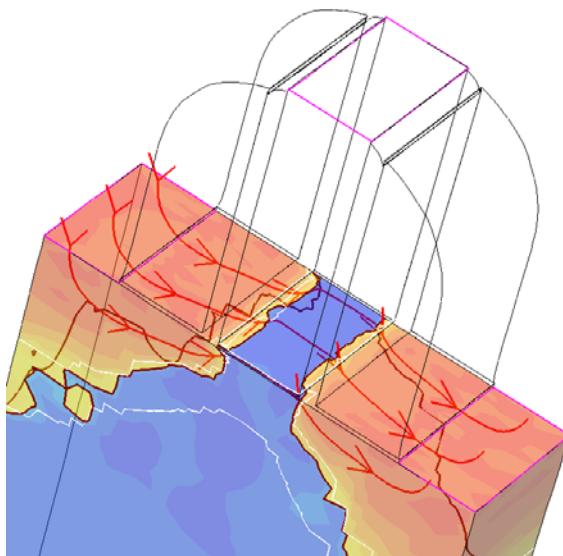
- The **Vector Field** box is where you select the field used to calculate the streamlines.
- The **Direction** box allows you to show only streamlines ending on a point, starting from a point, or both.
- The **Create Rake** option allows you to create multiple streamlines between the start and end points. The number of streamlines is defined by the value in the **Streamlines per Rake** box.
- The **Add by Click** button allows you to add the start point and the endpoint for the rake using the mouse to click the selected plot. If the **Create Rake** option is selected, you can add two positions. If the **Create Rake** option is not selected, you can add one position.

Chapter 4: Working With 2D and 3D Plots

Streamlines

- The **Create Streamlines** button allows you to create a family of streamlines going from a starting point to a rake end point. This button also changes its behavior when existing streamlines are selected from the list, allowing you to modify their attributes without creating new streamlines.

Figure 136 Example of displaying streamlines on plot



Specifying Regions or Materials

On the **Regions/Materials** tab, you can specify in which regions or materials the streamlines will be plotted. Not selecting a region or material causes Sentaurus Visual to plot the streamlines over the complete structure.

Representing the Streamlines

On the **Representation** tab, you can make cosmetic changes to the appearance of streamlines such as the line style, width, and resolution, as well as the color and the size of the vector field arrows.

Integration Settings

Sentaurus Visual has default integration settings that work with most of the simulation results obtained from other TCAD tools. However, users have the opportunity to fine-tune these values if needed. The **Integration** tab is for this purpose.

Integration Tab

By default, the Runge-Kutta 4 (RK4) algorithm is used for numeric integration of the fields. Some details about this integration can be modified. These values are included in the `create_streamline` command (see [create_streamline on page 237](#)). When you use the Streamlines dialog box, you can select between the values calculated by Sentaurus Visual or the default values specified in the User Preferences dialog box.

Step options are:

- **Initial** sets the initial step for the vector field integration. In the Runge-Kutta 4 (RK4) algorithm, the initial step is also a constant length for all steps.
- **Max Step** sets the maximum number of steps until the end of the integration. For termination constraints, either the **Max Step** value or the **Terminal Speed** value can be changed.

Others options are:

- **Maximum Propagation** controls the length of the streamline. If the **Both Direction** option is selected, the maximum length will be two times this value.
- **Terminal Speed** sets an end constraint for the numeric integration. If the particle speed is reduced to a value less than this number, the integration will end. For termination constraints, either the **Terminal Speed** value or the **Max Step** value can be changed.

Managing Created Streamlines

The Streamlines dialog box includes a list of the created streamlines in the Streamline Names pane. If you select **<New>** in this pane, you enter the creation mode, in which you can create streamlines. In the same way, if you select another streamline, the update mode is activated.

In the update mode, the Streamlines dialog box executes the `set_streamline_prop` Tcl command, which changes the representation of the streamlines by updating their properties without creating new ones in a faster way. Selecting a streamline in the Streamline Names pane also highlights the streamline in the plot, allowing you to easily identify the active streamline.

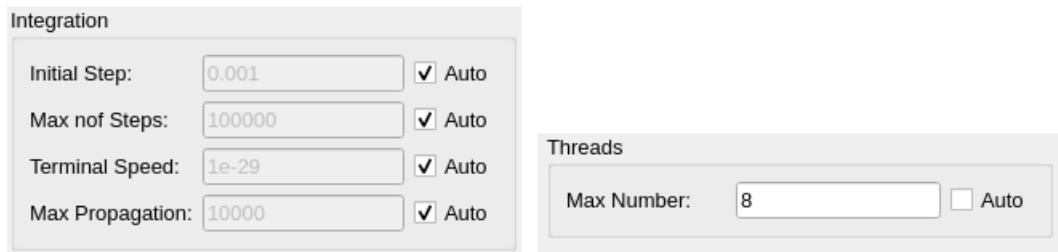
Configuring General Parameters of Streamlines

In the User Preferences dialog box (choose **Edit > Preferences**), several parameters are available that can be changed to improve the performance of creating streamlines (expand **2D/3D > Streamlines**) (see [Figure 137](#)).

Chapter 4: Working With 2D and 3D Plots

Streamlines

Figure 137 User Preferences dialog box showing (left) parameters for streamlines and (right) parameters for threads (used by streamlines)



Sentaurus Visual calculates the best integration parameters depending on the selected structure and vector fields. You can define the number of threads used to create rakes of streamlines in the User Preferences dialog box: expand **Common > Miscellaneous** and, under Threads, define the maximum number of threads. You can select **Auto** next to the **Max Number** field to let Sentaurus Visual compute the ideal number of threads to use or set a preferred value (see [Figure 137](#)).

Extracting Data From Streamlines

Sentaurus Visual can extract data from existing streamlines. Each streamline generates its own 1D dataset that contains the coordinates data defining the streamline as well as all scalar fields defined in the geometry.

Extracting data using the Streamlines dialog box will create a new xy plot (if it is not created already) and one curve for each streamline extracted, displaying the current contour-banded field (from the source plot) versus distance. If the field is scaled with something other than **Linear** or **Custom**, Sentaurus Visual will set the vertical axis (left y-axis) to logarithmic scale (see [Visualizing Fields on page 107](#)).

To extract data from streamlines:

1. Select one or more streamlines from the **Streamline Names** pane.

This makes the **Extract** and **Delete** buttons available (see [Figure 138](#)).

2. Click **Extract**.

Chapter 4: Working With 2D and 3D Plots

Streamlines

Figure 138 Streamlines dialog box showing Representation tab: streamlines have been created and some are selected, and the Extract button is available

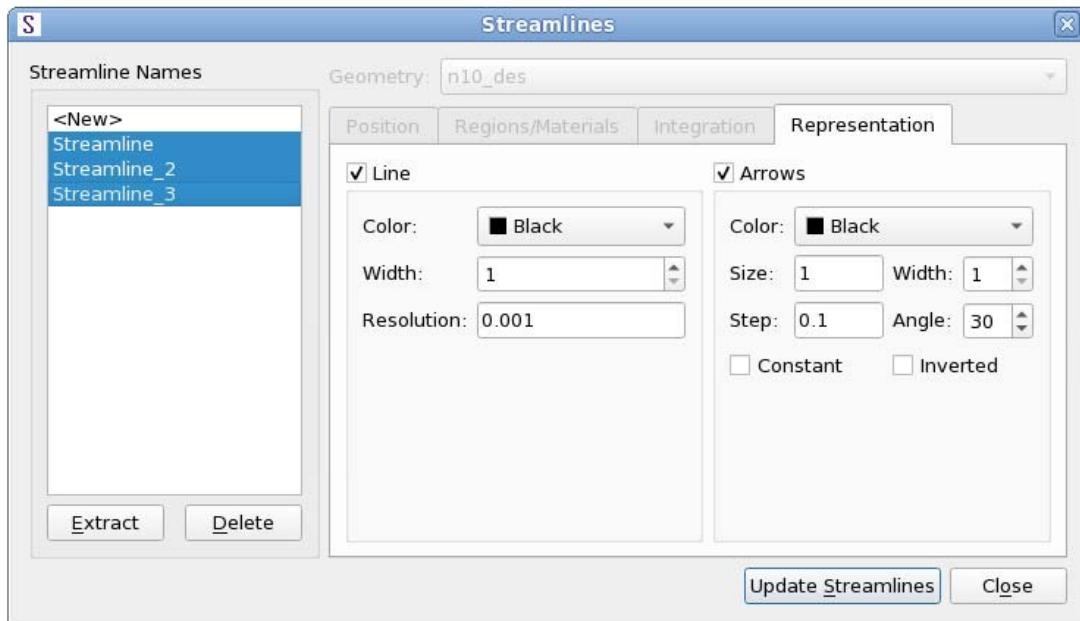


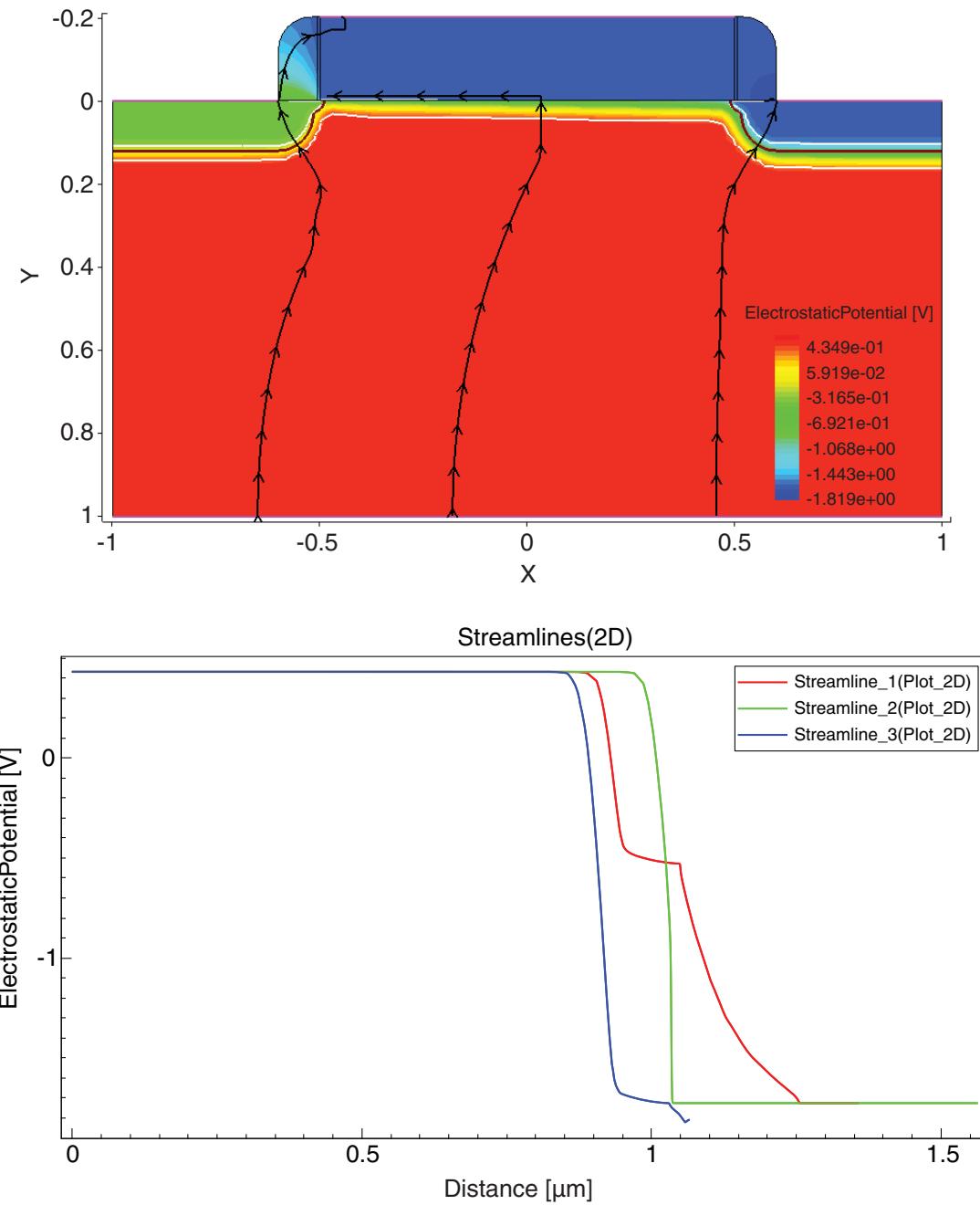
Figure 139 shows the results of the extraction operation.

The equivalent command for extracting data from streamlines is `extract_streamlines` (see [extract_streamlines on page 253](#)).

Chapter 4: Working With 2D and 3D Plots

Streamlines

Figure 139 (Top) Two-dimensional plot displaying three streamlines and (bottom) xy plot displaying three curves from the data extracted from the streamlines



5

Automated Tasks

*This chapter presents how to automate tasks with *Tcl* or *Python* scripting and *Inspect* compatibility.*

Running *Tcl* or *Python* Scripts

Sentaurus Visual allows scripts to be called from the command line or the user interface.

To run a *Tcl* script from the command line, enter:

```
svisual /path/to/script.tcl
```

To run a *Python* script from the command line, enter:

```
svisual -python /path/to/script.py
```

To run a script from the user interface:

► Choose **File > Run Script**.

The following examples illustrate some typical scripting uses in the context of batch scripts.

Example: Plot Id–Vg Curve

The contents of the script `plot_idvg.tcl` are:

```
# Load PLT data file.
set mydata [load_file IdVg_n62_des.plt]

# Create new empty xy plot.
set myplot [create_plot -1d]

# Create Id-Vg curve using loaded dataset and display on new xy plot.
set IdVgcurve [create_curve -plot $myplot -dataset $mydata \
    -axisX "gate InnerVoltage" -axisY "drain TotalCurrent"]

# Customize the curve.
set_curve_prop $IdVgcurve -plot $myplot -show_markers -markers_size 7 \
```

Chapter 5: Automated Tasks

Running Tcl or Python Scripts

```
-color red -label "nMOS"

# Display grid and set grid properties.
set_plot_prop -show_grid
set_grid_prop -show_minor_lines \
    -line1_style dash -line1_color #a0a0a4 \
    -line2_style dot -line2_color #c0c0c0

# Assign axis labels and set range.
set_axis_prop -plot $myplot -axis x -title "Vgate \[V\]"
set_axis_prop -plot $myplot -axis y -title "Idrain \[A/\mu m\]" -type log
set_axis_prop -plot $myplot -axis y -range {1e-09 0.0002}

# Export plot into PNG file.
export_view "curve.png" -plots $myplot -resolution 500x500 \
    -format PNG -overwrite
```

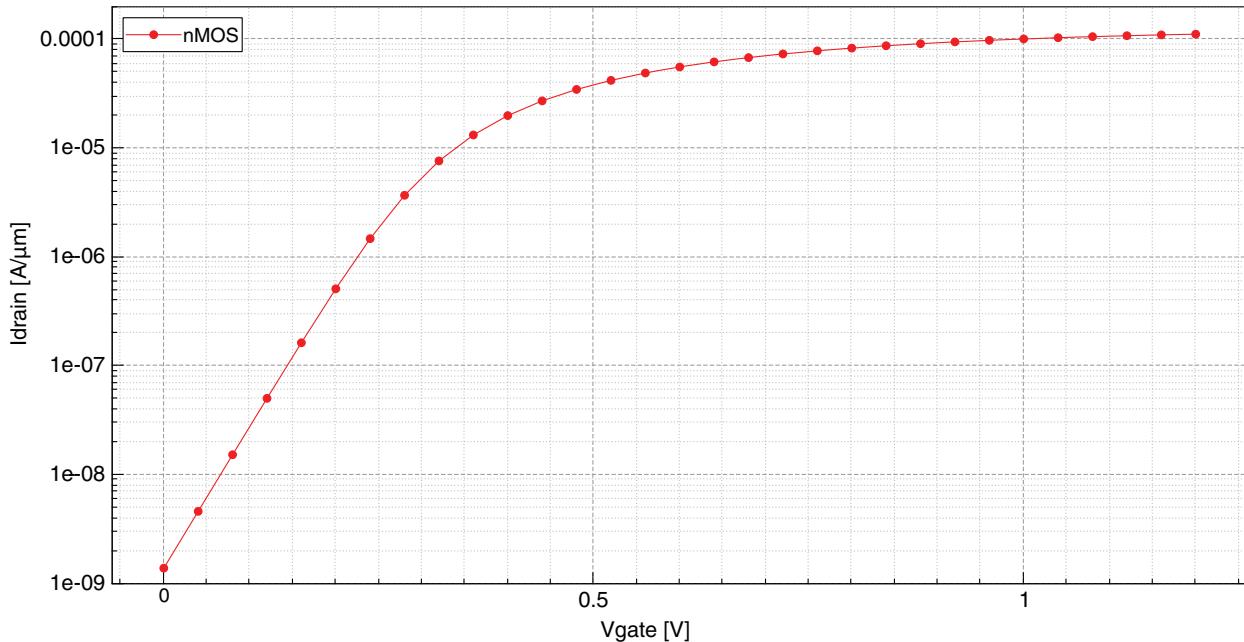
The first three commands of this script open a .plt file and create an I_d - V_g curve. Next, the plot is customized to make it more readable. Finally, the plot is exported to a .png file (see [Figure 140](#)).

Note:

This script must be executed with the virtual X server option to allow graphics export in batch mode. For example:

```
% svisual -batchx plot_idvg.tcl
```

Figure 140 I_d - V_g plot from xy data



Example: Create Cutline and Export Cutline Data to CSV File for Further Processing

The contents of the script `plot_npn.tcl` are:

```
# Load TDR file.  
set mydata2D [load_file npn_msh.tdr]  
  
# Create new plot.  
set myplot2D [create_plot -dataset $mydata2D]  
  
# Create 1D cutline normal to x-axis at point x=-0.005.  
set mydata1D [create_cutline -plot $myplot2D -type x -at -0.005]  
export_variables {DopingConcentration xMoleFraction Y} \  
-dataset $mydata1D -filename "data.csv" -overwrite
```

The first two commands load and display a TDR file. The next `create_cutline` command creates a cutline at the specified location. The last command exports the selected variables from the cutline to a CSV file.

Note:

This script can be run solely in batch mode, with the command:

```
% svisual -batch plot_npn.tcl
```

Saving Command History

Almost every action performed in Sentaurus Visual is replicated in the Tcl or Python Console. These actions can be saved to be executed in another session by clicking the **Save** button in the Console.

Running Inspect Command Files

Sentaurus Visual can run Inspect command files.

You can run an Inspect command file in the same way as for a native Sentaurus Visual Tcl script.

Script Library

Sentaurus Visual allows you to add Tcl script files as libraries, which can be loaded automatically at startup or manually using the Tcl command `load_library` (see [load_library on page 320](#)).

Chapter 5: Automated Tasks

Script Library

A script library has the file name formatted as `<libraryName>.tcl`.

The default library path is `$STROOT_LIB/svisuallib`. In addition, it includes a user-defined library path, which is set by default to `${HOME}/svisuallib`, but it can be modified in the user preferences.

Note:

The default value for the `STROOT_LIB` variable is `$STROOT/tcad/$STRELEASE/lib`.

Both paths can be checked for Tcl scripts (any file with the extension `.tcl`) for auto-loading at startup, which can be switched on or off in the user preferences.

The following options related to launching Sentaurus Visual are only valid when the auto-loading of the script library is switched on:

- `-nolibrary` deactivates the auto-loading of scripts from the library.
- `-librarypath <customPath>` adds a custom path to the list of library paths to look for script files when auto-loading is switched on.

Restrictions

Every procedure defined in a script library must begin with the prefix `lib_` to avoid the possible redefining of any existing Sentaurus Visual command.

At the time of loading one or more script files from the script library paths, if there are procedures that have been defined without this prefix, a warning message will be displayed, listing these procedures.

Moreover, if there are procedures that redefine Sentaurus Visual commands, a second warning message is displayed.

A

Tcl Commands

This appendix describes the tool command language (Tcl) commands that can be used in Sentaurus Visual.

The Tcl commands apply to all plots and structures unless stated otherwise.

Syntax Conventions

The following conventions are used for the syntax of Tcl commands:

- Angle brackets – <> – indicate text that must be replaced, but they are *not* part of the syntax.
- Braces – {} – are used for lists of values, and they must be included in the syntax.
- Brackets – [] – indicate that the argument is optional, but they are *not* part of the syntax.
- Parentheses – () – are used solely to group arguments to improve legibility of commands, but they are *not* part of the syntax.
- A vertical bar – | – indicates options, only one of which can be specified.

Object Names: -name Argument

For all Tcl commands that use the `-name` argument, if a name conflict is detected, Sentaurus Visual will print an error message and stop execution of the command. For example:

```
create_plot -name newPlot -dataset 3D
#-> newPlot
create_plot -name newPlot -dataset 2D
#-> "Error: create_plot: The plot couldn't be created. Plot name
'newPlot' already exists."
```

If you do not specify the `-name` argument in a command, Sentaurus Visual will generate an internal name that will remain consistent in a script. If the name generated conflicts with a

Appendix A: Tcl Commands

Common Properties

name defined later in a script for the same type of element (such as a curve or cutline), Sentaurus Visual will print an error message and stop execution of the command.

Common Properties

The following properties are used in several Tcl commands.

Colors

In Tcl commands that allow you to specify color properties (such as the `-color <#rrggbb>` option), a string specifying red, green, and blue components of the RGB system is expected. The string is preceded by a hash (#) character, and each value is provided in hexadecimal form. Common colors also have aliases.

Table 11 Common colors

Alias	General form	Description
white	#ffffff	White
black	#000000	Black
red	#ff0000	Red
darkRed	#800000	Dark red
lime	#00ff00	Light green
green	#008000	Dark green
darkGreen	#006400	Darker green
blue	#0000ff	Blue
darkBlue	#000080	Dark blue
cyan	#00ffff	Cyan
darkCyan	#008080	Dark cyan
magenta	#ff00ff	Magenta
darkMagenta	#800080	Dark magenta
yellow	#ffff00	Yellow

Appendix A: Tcl Commands

Common Properties

Table 11 Common colors (Continued)

Alias	General form	Description
olive	#808000	Olive or dark yellow
orange	#ffa500	Orange
darkOrange	#ff8c00	Dark orange
gray	#a0a0a4	Gray
darkGray	#808080	Dark gray
lightGray	#c0c0c0	Light gray
skyblue	#87ceeb	Sky blue
slategray	#708090	Slate gray
chocolate	#d2691e	Chocolate

Fonts

For Tcl commands that allow you to adjust font properties, Sentaurus Visual defines a specific list of font families and attributes.

Table 12 Font families and their attributes

Font family	Attribute
Arial	Bold
Courier	Italic
Times	Normal
	Strikeout
	Underline

Note:

In xy plots, the font size of different elements of the plot are set with the `font_size` argument; whereas in 2D and 3D plots, the font size cannot be set directly. Instead, the font size is set as a factor of the plot frame (the default value is 1.0), with the `font_factor` argument.

Appendix A: Tcl Commands

Common Properties

Lines

For Tcl commands that allow you to adjust line properties (such as the `-line_style` option), Sentaurus Visual defines a specific list of line styles. You can provide the name of the style or its short form directly.

Table 13 Line styles

Name of line style	Short form of line style	Description
solid	-	Continuous line: _____
dot	.	Dotted line:
dash	-	Dashed line: -----
dashdot	- .	Alternating dash-and-dot line: -.-.-.-.-
dashdotdot	- ..	Alternating dash-and-two-dots line: -...-.-.-

Markers

Different markers are available to use in xy plots in Sentaurus Visual. Tcl commands allow you to use the name or the short form of each marker.

Table 14 Marker types

Name of marker type	Short form of marker type	Description
circle	o	○
circlef	of	●
diamond		◇
diamondf		◆
square		□
squaref		■
plus	+	+
cross	x	X

Appendix A: Tcl Commands

add_custom_button

add_custom_button

Adds a custom button to the Scripts toolbar.

Note:

This command works only in interactive mode. It has no effect in batch mode.

If both `-icon` and `-name` are specified, only the icon is shown in the toolbar.

Syntax

```
add_custom_button
    -file <stringValue> | -script <stringValue> | -separator
    [-desc <stringValue>] [-icon <stringValue>]
    [-label <stringValue>] [-name <stringValue>]
```

Argument	Description
<code>-desc <stringValue></code>	Text that describes the button. This appears as the tooltip of the button.
<code>-file <stringValue></code>	Name of a Tcl file to execute.
<code>-icon <stringValue></code>	Specifies a graphics file to be used as the button icon. Supported file formats are BMP, GIF, PNG, and SVG.
<code>-label <stringValue></code>	Specifies a label for the button.
<code>-name <stringValue></code>	Name of the button in the Scripts toolbar. See Object Names: -name Argument on page 213 .
<code>-script <stringValue></code>	Name of a Tcl script to execute.
<code>-separator</code>	Adds a separator between buttons in the Scripts toolbar.

Returns

String with the name of the custom button specified by `-name`. If this argument is not specified, the command returns the default name, which is `S<number>`, where `<number>` starts at 1 and increases each time a custom button is created.

Example

```
add_custom_button -script "echo Hello World!" -name Greetings \
    -desc "Echoes a Hello World Message."
#-> Greetings
```

Appendix A: Tcl Commands

add_frame

add_frame

Adds a new frame to the frame buffer.

Note:

You must use of the `start_movie` command before using the `add_frame` command.

Syntax

```
add_frame [-name <stringValue>] [-plot <stringValue>]
```

Argument	Description
<code>-name <stringValue></code>	Name of the new frame to be captured. See Object Names: -name Argument on page 213 .
<code>-plot <stringValue></code>	Name of the plot where the new frame will be saved. Current active plot is used by default.

Returns

String.

Example

```
add_frame -name Frame1  
#-> Frame1
```

See Also

[start_movie on page 382](#)

Appendix A: Tcl Commands

calculate

calculate

This function extracts FET parameters from I_d-V_d or I_d-V_g curves.

Note:

This command applies to xy plots only.

Syntax

```
calculate <stringValue> -op vth | gmmax | idsat | ioff | rout | ron  
[-plot <stringValue>]
```

Argument	Description
<stringValue>	Name of the curve on which to apply the parameter extraction.
-op vth gmmax idsat ioff rout ron	Parameter to be extracted from the curve. For more detailed information about the extraction parameters, see Computing Electrical Characteristics on page 97 .
-plot <stringValue>	Name of the plot on which to apply the parameter extraction.

Returns

Double.

Example

```
calculate Curve_1 -op ron  
#-> 0.0554013
```

Appendix A: Tcl Commands

calculate_field_value

calculate_field_value

Calculates the minimum and maximum values of a particular field, and shows (with a marker) the location of these values.

Note:

This command applies to 2D and 3D plots only.

Syntax

```
calculate_field_value -min | -max  
  [-dataset <stringValue> | -plot <stringValue>]  
  [-field <stringValue>]  
  [-geom <stringValue>]  
  [-materials <stringList> | -regions <stringList>]  
  [-ranges {<x1> <x2> <y1> <y2> [<z1> <z2>]}]
```

Argument	Description
-min -max	Selects whether the point returned will be the minimum or maximum value.
-dataset <stringValue> -plot <stringValue>	Selects either the dataset (for batch mode only) or the plot from which the data should be extracted.
-field <stringValue>	Name of the field from which the data will be obtained. If not specified, the command uses the current contour-band field displayed in the plot.
-geom <stringValue>	Selects the geometry from which the data will be extracted.
-materials <stringList> -regions <stringList>	Sets a list of materials or regions that will be used to find the minimum or maximum value.
-ranges {<x1> <x2> <y1> <y2> [<z1> <z2>]}	Sets a specific range of values to find the minimum or maximum value. After the command is executed, the minimum or maximum value and its position are returned.

Returns

Double and a list of coordinate values.

Example

```
calculate_field_value -plot Plot_n9_des \  
  -field Abs(TotalCurrentDensity-V) -min  
# position:  
# min: 271.172 5.39062 0 value 0.0206237
```

Appendix A: Tcl Commands

calculate_scalar

```
# max: 277 -0.546875 0 value 426.765
#-> 0.0206237318885 {271.172 5.39062 0}
```

calculate_scalar

Calculates a scalar value.

This command operates over curves or variables. If `-curves` is specified, `-plot` can specify a plot in memory. Otherwise, the command uses the curve in the selected plot.

Note:

This command applies to xy plots only.

Syntax

```
calculate_scalar
  (-curves [-plot <stringValue>] | -variables)
  -function <stringValue>
```

Argument	Description
<code>-curves</code> <code>-variables</code>	Select whether the command parses the specified formula as a function of curves or variables.
<code>-function <stringValue></code>	Specifies a formula to be used to extract a scalar value. All mathematical operations can be used; however, you must ensure that the last operation that encloses the entire set of functions is a scalar value function. Otherwise, the command fails. For information about which functions return a double value (scalars), see Appendix D on page 405 .
<code>-plot <stringValue></code>	Name of the plot where the command will search for curves. If not specified, the command uses the selected xy plot.

Returns

Double.

Example

```
calculate_scalar -curves -plot Plot_1 \
  -function vecmax(<Curve_1>)+vecmin(<Curve_1>)
#-> 0.000351277048757
```

Appendix A: Tcl Commands

create_curve

create_curve

Creates a new curve for an xy plot.

If `-plot` is not specified, the command draws the curve on the selected plot. If there are no xy plots created or the selected plot is not an xy plot, the command returns an error.

Note:

This command applies to xy plots only.

Syntax

```
create_curve
    -dataset <stringList> -axisX <stringValue>
    (-axisY <stringValue> | -axisY2 <stringValue>) |
    -function <stringValue>
    [-name <stringValue>] [-plot <stringValue>] [-y <intValue>]
```

Argument	Description
<code>-dataset <stringList></code> <code>-function <stringValue></code>	Specify either a list of dataset names where the information is extracted or a formula from which to create a curve.
<code>-axisX <stringValue></code>	Specifies the variable to be used for the x-axis.
<code>-axisY <stringValue></code> <code>-axisY2 <stringValue></code>	Specifies the variable to be used for the y-axis or the y2-axis.
<code>-name <stringValue></code>	Name of the new curve. If not specified, the command assigns a default curve name. See Object Names: -name Argument on page 213 .
<code>-plot <stringValue></code>	Name of the plot where the new curve will be displayed. If not specified, the command draws the curve on the selected xy plot.
<code>-y <intValue></code>	Specifies whether a function is created against a particular axis: • <code>-y 1</code> specifies the y-axis. • <code>-y 2</code> specifies the y2-axis. This argument works only with <code>-function</code> .

Returns

List.

Appendix A: Tcl Commands

create_cut_boundary

Example

```
create_curve -plot Plot_1 -dataset IdVd_example \
    -axisX "drain OuterVoltage" \
    -axisY "drain TotalCurrent"
#-> Curve_1
```

create_cut_boundary

Creates a cutline along the specified structure boundary.

The command produces a list of line segments that define the cutline. A segment is defined as the union of two vertices, where a vertex is a point that defines a region (angle > 30° with its neighboring points).

If `-plot` or `-dataset` is not specified, the command uses the selected 2D plot dataset or free cutplane.

Note:

This command applies to 2D plots and free cutplanes only.

Syntax

```
create_cut_boundary
    (-materials <stringList> | -regions <stringList>)
    -points <pointList>
    [-dataset <stringValue> | -plot <stringValue>] [-name <stringValue>]
    [-reverse] [-segments <stringList>]
```

Argument	Description
<code>-materials <stringList></code> <code>-regions <regionsList></code>	Materials or regions to which the boundary belongs.
<code>-points <stringList></code>	Vertices on the boundary through which the cutline will pass. Points must be defined as: <code>{"x0 y0 [z0]" "x1 y1 [z1]" ...}</code> Any component not present in the plot must be set to 0.
<code>-dataset <stringValue></code> <code>-plot <stringValue></code>	Specifies the plot or dataset from which to retrieve the regions or materials. If not specified, the command uses the selected plot.
<code>-name <stringValue></code>	Name of the cutline dataset. If not specified, the command generates a default name. See Object Names: -name Argument on page 213 .
<code>-reverse</code>	Reverses the direction of the cutline creation backwards.

Appendix A: Tcl Commands

create_cutline

Argument	Description
-segments <stringList>	Specifies the regions from which the data will be extracted in each segment line. The length of the list must be exactly the number of vertices along the cutline minus 1.

Returns

String (the name of the resultant 1D dataset).

Example

```
create_cut_boundary -plot Plot_2D -regions {R.Gateox R.PolyReox} \
    -segments {R.Gateox R.PolyReox}
    -points {"-0.5125 -0.002 0" "-0.6 -0.002 0" "-0.6 0 0"} -name myCut
#->myCut
```

create_cutline

Creates a new cutline.

If the type of cutline is aligned to an axis, you must specify the `-at` argument.

If `-type free` is specified, you must specify the `-points` argument.

The new plot created has the same name as the cutline dataset, with the prefix `Plot_`.

Note:

This command applies to 2D and 3D plots only. Axis-aligned cutlines can be generated from 3D plots only if they are particle Monte Carlo (PMC) structures, even though they accept only `-type free` cutlines.

Syntax

```
create_cutline
    -type x | y | z
    -at <doubleValue> [-axisX x | y | z [-surface]] |
    -type free -points {<x1> <y1> [<z1>] <x2> <y2> [<z2>]}
    [-dataset <stringValue> | -plot <stringValue>]
    [-materials <stringList> | -regions <stringList>]
    [-name <stringValue>]
```

Appendix A: Tcl Commands

create_cutline

Argument	Description
<code>-at <doubleValue> [-axisX x y z [-surface]] -points {<x1> <y1> [<z1>] <x2> <y2> [<z2>]}</code>	If an axis is selected using <code>-type</code> , the <code>-at</code> argument must be used. If <code>-type free</code> is specified, two (x,y) points must be specified with the <code>-points</code> argument. (In 3D plots, you must specify two (x, y, z) points.) If the source structure is a 3D PMC structure, you can specify <code>-surface</code> to extract the surface data as a 1D dataset. If <code>-surface</code> is specified, <code>-axisX</code> sets the reference axis against which curves are displayed.
<code>-type x y z free</code>	Selecting <code>x</code> , <code>y</code> , or <code>z</code> ties the cutline to the specified axis. The <code>free</code> option allows you to create a cutline, drawing a line between two (x,y) coordinates.
<code>-dataset <stringValue> -plot <stringValue></code>	Name of the dataset or plot from where the cutline is generated. If neither is specified, the command uses the selected 2D or 3D plot dataset.
<code>-materials <stringList> -regions <stringList></code>	If specified, the cutline operation is performed only on the materials or regions listed.
<code>-name <stringValue></code>	Name of the new cutline dataset. If not specified, the command generates a default name. See Object Names: -name Argument on page 213 .

Returns

String (the name of the resultant 1D dataset).

Example

```
create_cutline -plot Plot_2D -type free -points {-0.45 -0.15 0.30 0.80} \  
                  -name myCutlineDataset  
#-> myCutlineDataset
```

Appendix A: Tcl Commands

create_cutplane

create_cutplane

Creates a new cutplane.

The new plot created has the name of the cutplane. If `-plot` or `-dataset` is not specified, the command uses the selected 3D plot dataset.

Note:

This command applies to 3D plots only.

Syntax

```
create_cutplane
  -type x | y | z | free -at <doubleValue> |
  -type free -origin {<x> <y> <z>} -normal {<x> <y> <z>}
  [-dataset <stringValue> | -plot <stringValue>] [-name <stringValue>]
```

Argument	Description
<code>-at <doubleValue></code> <code>-origin {<x> <y> <z>}</code>	With <code>-at</code> , cuts the structure at the value specified in the axis defined by <code>-type</code> (it must not be <code>free</code>). With <code>-origin</code> and <code>-normal</code> , cuts the structure with a plane defined by the given origin and normal. The argument <code>-type</code> must be <code>free</code> .
<code>-type x y z free</code>	Selects the axis from which the cutplane is generated.
<code>-dataset <stringValue></code> <code>-plot <stringValue></code>	Name of the dataset or plot from where the cutplane is generated. If not specified, the command uses the selected plot.
<code>-name <stringValue></code>	Name of the new cutplane dataset. If not specified, the command generates a default name as a function of the original 3D dataset. See Object Names: -name Argument on page 213 .

Returns

String (the name of the resultant 2D dataset).

Example

```
create_cutplane -plot Plot_3D -name Cut1 -type y -at 0.3
#-> Cut1
```

Appendix A: Tcl Commands

create_cutpolyline

create_cutpolyline

Creates a new cutline with the specified number of vertex points.

Note:

This command applies to 2D plots only.

Syntax

```
create_cutpolyline -points <pointList>
    [-dataset <stringValue> | -plot <stringValue>]
    [-materials <stringList> | -regions <stringList>]
    [-name <stringValue>]
```

Argument	Description
-points <pointList>	Points in the region where the cutline will pass through. Points must be defined as { "x0 y0" "x1 y1" ...}.
-dataset <stringValue> -plot <stringValue>	Name of the dataset or plot from where the cutline will be created. If neither is specified, the command uses the selected 2D plot dataset.
-materials <stringList> -regions <stringList>	If specified, the cutline will be performed only on the materials or regions listed.
-name <stringValue>	Name of the cutline. See Object Names: -name Argument on page 213 .

Returns

String.

Example

```
create_cutpolyline -plot Plot_2D \
    -points {"0.05872 -0.260434" "0.46536 -0.034674" "0.26 0.074126"}
#-> C1(2D)
```

Appendix A: Tcl Commands

create_cutpolyplanes

create_cutpolyplanes

Creates a series of cutplanes using a polyline as input. The new plot has the name of the cutplane.

Note:

This command applies to 3D plots only.

Syntax

```
create_cutpolyplanes
    -type x | y | z
    -points <pointList>
    [-dataset <stringValue> | -plot <stringValue>]
    [-geom <stringValue>] [-name <stringValue>]
```

Argument	Description
-type x y z	Selects the extrusion axis for the polyline defined by the points.
-points <pointList>	Sets the vertices for the polyline to be extruded along the chosen axis to create cutplanes. Points must be defined as: { {x0 y0 z0} {x1 y1 z1} ... }
-dataset <stringValue> -plot <stringValue>	Name of the dataset or plot from where the cutplanes will be created. If neither is specified, the command uses the selected 3D plot dataset.
-geom <stringValue>	Name of the geometry to use for the cut. If not specified, the command uses the first geometry of the plot.
-name <stringValue>	Name of the new dataset. If not specified, the command generates a default name as a function of the original 3D dataset.

Returns

String (the name of the resultant 2D dataset).

Example

```
create_cutpolyplanes -type x
    -points { {0.3 0.1 0.2} {0.1 0.5 1.6} {2.4 0.5 1.3} }
    -plot Plot_3D -name Cut1
#-> Cut1
```

Appendix A: Tcl Commands

create_field

create_field

Creates a new field using data from the plot or the dataset specified in the arguments.

The command uses the selected 2D or 3D dataset if `-plot` or `-dataset` is not specified.

Note:

This command applies to 2D and 3D plots only.

Syntax

```
create_field -function <stringValue> -name <stringValue>
  [-dataset <stringValue> | -plot <stringValue> [-geom <stringValue>]]
  [-show]
```

Argument	Description
<code>-function <stringValue></code>	Specifies the function to be evaluated. For a complete list of operations, see Appendix D on page 405 .
<code>-name <stringValue></code>	Name of the new field. See Object Names: -name Argument on page 213 .
<code>-dataset <stringValue> -plot <stringValue> [-geom <stringValue>]</code>	Name of the plot or dataset from where the field is created. If not specified, the command uses the active plot. The argument <code>-geom</code> can be used only with <code>-plot</code> . It specifies the name of the geometry in the plot. If not specified, the command uses the first geometry associated with the plot.
<code>-show</code>	Shows immediately the newly created field if specified.

Returns

String.

Example

```
create_field -name newFld -dataset 3D -function "log(<ElectricField>)"
#-> newFld
```

Appendix A: Tcl Commands

create_iso

create_iso

Creates a new iso-geometry using an isovalue from the field of a geometry, or modifies an existing iso-geometry.

Note:

This command applies to 2D and 3D plots only.

Syntax

```
create_iso -field <stringValue> -value <doubleValue>
           [-color <#rrggbb>] [-geom <stringValue>]
           [-modify | -name <stringValue>]
           [-plot <stringValue>]
```

Argument	Description
-field <stringValue>	Selects the field on which the isovalue is used. If not specified, the command uses the current contour band field displayed in the plot.
-value <doubleValue>	Specifies the isovalue with which to create the new iso-geometry.
-color <#rrggbb>	Specifies the color to be used for the new iso-geometry. If not specified, the command uses the default color (gray).
-geom <stringValue>	Name of the geometry in the plot. If not specified, the command uses the first geometry associated with the plot. If <code>-modify</code> is specified, this argument is the name of the iso-geometry to be modified.
-modify -name <stringValue>	Specifies either the name of the new iso-geometry to be created or that the geometry defined by <code>-geom</code> will be modified by the arguments in the command. See Object Names: -name Argument on page 213 .
-plot <stringValue>	Specifies the plot from which to retrieve the geometry. If not specified, the command uses the selected plot.

Returns

String naming the new geometry or the modified geometry.

Example

```
create_iso -plot Plot_3D -geom 3D -field ElectrostaticPotential \
           -value 0.0 -name Isol(3D) -color blue
```

Appendix A: Tcl Commands

create_plot

```
#-> Isol(3D)  
  
create_iso -modify -geom Isol(3D) -value 0.8 -color green  
#-> Isol(3D)
```

create_plot

Creates an empty xy plot, or creates a plot from 2D or 3D datasets.

Syntax

```
create_plot  
    -1d | -dataset <stringValue> | -duplicate <stringValue>  
    [-name <stringValue>]  
    [-ref_plot <stringValue>]  
    [-tdr_state_index <intValue>]
```

Argument	Description
-1d -dataset <stringValue> -duplicate <stringValue>	The arguments are: <ul style="list-style-type: none">-1d creates an empty xy plot.-dataset creates a plot from a loaded 1D, 2D, or 3D dataset.-duplicate replicates the properties of the plot in a new plot (applies to xy plots only).
-name <stringValue>	Name of the new plot. See Object Names: -name Argument on page 213 .
-ref_plot <stringValue>	Name of the plot to be used as a reference to inherit the fields and the region properties. This argument applies to 2D and 3D plots only.
-tdr_state_index <intValue>	The new plot will load only the specified TDR state index from an already loaded dataset. The resulting plot is not considered be to a multistate plot. This argument applies to 2D and 3D plots only.

Returns

String.

Example

```
create_plot -dataset 3D  
#-> Plot_3D
```

Appendix A: Tcl Commands

create_projection

create_projection

Creates a 2D plot with maximum or minimum values along a 3D plot axis.

The argument `-resolution` increases the precision of the maximum or minimum calculation. Higher values of resolution lead to longer calculation times.

Note:

This command applies to 3D plots only.

Syntax

```
create_projection
    -field <stringValue>
    -function max | min
    -type x | y | z | free
    [-dataset <stringValue> | -plot <stringValue>] [-geom <stringValue>]
    [-name <stringValue>]
    [-normal <doubleList> | -materials <stringList> |
     -regions <stringList> |
     -window {<x1> <y1> [<z1>] <x2> <y2> [<z2>]}]
    [-origin <doubleList>] [-resolution <x><y><x><z>]
    [-width <doubleValue>]
```

Argument	Description
<code>-field <stringValue></code>	Name of the field to be projected.
<code>-function max min</code>	Specifies either the maximum values projection or minimum values projection.
<code>-type x y z free</code>	Specifies the axis to be projected: <ul style="list-style-type: none">• <code>x</code> projects a 2D <code>yz</code> plot.• <code>y</code> projects a 2D <code>xz</code> plot.• <code>z</code> projects a 2D <code>xy</code> plot.• <code>free</code> projects a 2D plot using an arbitrary plane.
<code>-dataset <stringValue> -plot <stringValue></code>	Specifies the plot or dataset from which to retrieve the regions or materials. If not specified, the command uses the selected plot.
<code>-geom <stringValue></code>	Geometry that has the given field. If not specified, the command uses the first shown geometry from the selected dataset.
<code>-name <stringValue></code>	Name of the resultant 2D projection plot. See Object Names: -name Argument on page 213 .

Appendix A: Tcl Commands

create_projection

Argument	Description
-normal <doubleList> -materials <stringList> -regions <stringList> -window {<x1> <y1> [<z1>] <x2> <y2> [<z2>]}	For free projection types, -normal sets the normal for the plane. It must contain three values, which will be normalized if required. If -materials or -regions is used, then they specify the materials or regions where the maximum or minimum data will be extracted. If -window is used, then all regions inside that window are selected. If none of these options is specified, then the command uses all regions in the entire domain.
-origin <doubleList>	Sets the origin for the plane of free projection types.
-resolution <x>x<y>x<z>	Specifies the resolution of the maximum or minimum search data algorithm. If not specified, then -resolution 50x50x50 is used by default.
-width <doubleValue>	Sets the width of the domain considered for free projection types.

Returns

String.

Example

```
create_projection -plot Plot_3D -field DopingConcentration \  
    -function max -normal x -resolution 50x50x50  
#-> Projection_max(3D)
```

Appendix A: Tcl Commands

create_projection_path

create_projection_path

Creates a 2D plot with maximum or minimum values extracted from several planes across a path defined by points.

Note:

This command applies to 3D plots only.

Syntax

```
create_projection_path
    -field <stringValue>
    -function max | min
    -points <pointList>
    -type x | y | z
    [-dataset <stringValue> | -plot <stringValue>]
    [-geom <stringValue>]
    [-name <stringValue>]
    [-resolution <x>x<y>x<z>]
    [-width <doubleList>]
```

Argument	Description
-field <stringValue>	Name of the field to be projected.
-function max min	Specifies either the maximum values projection or minimum values projection.
-points <pointList>	Specifies the points in the structure defining the path to construct the projection. Points must be defined as: {"x0 y0 z0" "x1 y1 z1" ...}
-type x y z	Specifies the extrusion axis for the projection. The resulting 2D plot will have the axis selected as the vertical axis.
-dataset <stringValue> -plot <stringValue>	Specifies the dataset or plot from which to retrieve the regions or materials. If not specified, the command uses the selected plot.
-geom <stringValue>	Specifies the geometry that has the given field. If not specified, the command uses the first shown geometry from the selected dataset.
-name <stringValue>	Name of the resultant 2D projection plot. See Object Names: -name Argument on page 213 .
-resolution <x>x<y>x<z>	Specifies the resolution of the maximum or minimum search data algorithm. If not specified, then -resolution 50x50x50 is used by default.

Appendix A: Tcl Commands

create_projection_path

Argument	Description
-width <doubleList>	Sets the width for every plane in the path. If not set, then the full extent of the structure is used. If a single value is used, then all planes use the same width extent. A list can be provided to set each plane width individually (the number of width values must equal the number of points minus one).

Returns

String.

Example

```
create_projection_path -plot Plot_3D -field DopingConcentration
    -type x -function max -resolution 50x50x50 -width {0.1 0.14}
    -points {"2.5e-05 -0.0669983 0.0378416" "-0.0950691 -0.0272096
        0.0251004" "-0.100966 0.00648555 0.0243684"}
#-> Projection_max(3D)
```

Appendix A: Tcl Commands

create_ruler

create_ruler

Measures the distance between two points.

Note:

The command applies to 2D and 3D plots only.

Syntax

```
create_ruler -id <intValue> -point1 {x y [z]} -point2 {x y [z]}
[-plot <stringValue>] [-snap_on | -snap_off]
```

Argument	Description
-id <IntValue>	Specifies the ID of the new ruler to be created.
-point1 {x y [z]}	Specifies the first point of the ruler.
-point2 {x y [z]}	Specifies the second point of the ruler.
-plot <stringValue>	Name of the plot where the command will measure the distance between two points. If not specified, the command uses the selected plot.
-snap_on -snap_off	Specifies whether to activate the snap-to-mesh feature.

Returns

The distance between the ruler points.

Example

```
create_ruler -id 2 -point1 {0.1 0.1 0.1} -point2 {0.5 0.5 0.2} -snap_off
#-> 0.574456264654
```

Appendix A: Tcl Commands

create_streamline

create_streamline

Creates a new streamline on a 2D or 3D plot.

Note:

The command applies to 2D and 3D plots only.

Syntax

```
create_streamline
    -direction forward | backward | both
    -field <stringValue>
    (-point {<x> <y> [<z>]} |
     -p1 {<x> <y> [<z>]} -p2 {<x> <y> [<z>]} -nofpoints <intValue>)
    [-geom <stringValue>]
    [-integ_initial_step <doubleValue>]
    [-integ_max_propagation <doubleValue>]
    [-integ_max_steps <intValue>]
    [-integ_terminal_speed <doubleValue>]
    [-materials <stringList> | -regions <stringList>]
    [-name <stringValue>] [-plot <stringValue>]
```

Argument	Description
-direction forward backward both	Direction of the streamline created. Default: both.
-field <stringValue>	Selects the field on which the streamline is created.
-point {<x> <y> [<z>]} -p1 {<x> <y> [<z>]} -p2 {<x> <y> [<z>]} -nofpoints <intValue>	Use the -point argument to create one streamline as follows: <ul style="list-style-type: none">If the direction is forward, the point specified is the starting point of the streamline.If the direction is backward, the point specified is the end point of the streamline.If the direction is both, the point specified is the middle of the streamline. Use the -nofpoints argument to create a custom number of streamlines going from point 1 to point 2 with the -p1 and -p2 arguments. For example, if you specify -nofpoints 9, then 7 streamlines in addition to the streamlines originating from point 1 and point 2 are created. Analogous to the -point argument, the direction determines the type of point.
-geom <stringValue>	Name of the geometry in the plot. If not specified, the command uses the first geometry associated with the plot.
-integ_initial_step <doubleValue>	Specifies the initial step used in all the calculations of the Runge-Kutta 4 algorithm. This step remains unchanged in all the mathematical processes.

Appendix A: Tcl Commands

create_streamline

Argument	Description
<code>-integ_max_propagation <doubleValue></code>	Maximum propagation is the maximum length of the streamline after calculating the magnitude of each step iteration. This is related to the 'line' length at the calculation level. This represents the maximum amount of data that will be calculated to generate a line of 'x'-length. However, you can represent the line with less data using the <code>-line_resolution</code> argument of the <code>set_streamline_prop</code> command. The value of <code>-line_resolution</code> must <i>not</i> be less than the value of <code>-integ_max_propagation</code> . Otherwise, it can lead to unexpected results.
<code>-integ_max_steps <intValue></code>	Specifies the number of iterations of the Runge-Kutta 4 algorithm to be applied.
<code>-integ_terminal_speed <doubleValue></code>	For each iteration, the Runge-Kutta 4 algorithm calculates the differential value of the vector field in the point and, using the Picard–Lindelöf iteration method, this value is equal to the speed value of the field in that point. This argument specifies the 'terminal speed' of the algorithm. If the next result iteration of the Runge-Kutta 4 algorithm is lower than this value, the integration stops.
<code>-materials <stringList> -regions <stringList></code>	Specify either: <ul style="list-style-type: none">Materials on which the streamline will be created. Default: All materials.Region on which the streamline will be created. Default: All regions.
<code>-name <stringValue></code>	Identifier for the new streamline created. If not specified, the command generates a default name. See Object Names: -name Argument on page 213 .
<code>-plot <stringValue></code>	Name of the plot. If not specified, the command uses the selected plot.

Returns

List.

Example

```
create_streamline -field ElectricField -point {0.5 0.2} -direction both  
#-> Streamline_1
```

Appendix A: Tcl Commands

create_surface

create_surface

Creates a new 3D dataset using a 2D dataset or geometry of a plot as source, or modifies an existing 3D surface dataset.

Note:

This command applies to 2D datasets (or plots) and 3D surface datasets (or plots) only.

Syntax

```
create_surface  
    [-dataset <stringValue> | -plot <stringValue> [-geom <stringValue>]]  
    [-factor <doubleValue>] [-field <stringValue>] [-name <stringValue>]  
    [-range <doubleList>] [-scale linear | logabs | asinh]
```

Argument	Description
-dataset <stringValue>	Specifies the dataset to be used to build the surface dataset.
-factor <doubleValue>	Specifies the factor that is used to multiply the coordinate to generate the surface. If not specified, the command uses one (1).
-field <stringValue>	Selects the field on which the surface is used. If not specified, the command uses the current contour band field displayed in the plot.
-geom <stringValue>	Name of the geometry in the plot. If not specified, the command uses the first geometry associated with the plot.
-name <stringValue>	Name of the surface dataset to be created. See Object Names: -name Argument on page 213 .
-plot <stringValue>	Specifies the plot from which to retrieve the geometry. If the plot is not specified, the command uses the selected plot.
-range <doubleList>	Pair of values (min and max) that defines the range to be used to limit the new coordinate values that generate the surface. If not specified, the command uses the entire range of the field.
-scale linear logabs asinh	Specifies the scale to be used on the coordinates to generate the surface dataset. If not specified, the command uses the linear scale.

Returns

String naming the new dataset.

Appendix A: Tcl Commands

create_variable

Example

```
create_surface -plot Plot_2D -geom 2D -field ElectrostaticPotential \
    -factor 0.2 -name Surface1(2D)
#-> Surface1(2D)
```

create_variable

Creates a new variable.

Note:

This command applies to xy plots only.

Syntax

```
create_variable
    -dataset <stringValue>
    (-function <stringValue> | -values <doubleList>)
    -name <stringValue>
```

Argument	Description
-dataset <stringValue>	Dataset from which values are obtained to evaluate functions.
-function <stringValue> -values <doubleList>	Expression to evaluate or the list of values to add to the dataset specified.
-name <stringValue>	Name of the new variable. See Object Names: -name Argument on page 213 .

Returns

String.

Example

```
create_variable -name nVar -dataset idvd -values {0.1 0.3 0.5 0.7 0.9}
#-> nVar
```

Appendix A: Tcl Commands

diff_plots

diff_plots

Creates a new dataset with the difference in the common fields of the selected plots.

Note:

This command applies to 2D and 3D plots only.

Syntax

```
diff_plots <stringList> [-display] [-normalized]
```

Argument	Description
<stringList>	List of plots to use to create the differential plot.
-display	Creates a new plot with the field difference dataset.
-normalized	Normalizes the values between the two plots.

Returns

String.

Example

```
diff_plots {Plot1 Plot2}  
#-> Plot1-Plot2
```

Appendix A: Tcl Commands

draw_ellipse

draw_ellipse

Draws an ellipse at the specified position. The ellipse is represented in relation to the rectangle that envelops it, although the rectangle is not drawn.

Note:

This command applies to xy plots only.

Syntax

```
draw_ellipse -p1 {<x1> <y1>} -p2 {<x2> <y2>}  
[-plot <stringValue>]
```

Argument	Description
-p1 {<x1> <y1>}	Specifies the upper-left corner of the rectangle that envelops the ellipse.
-p2 {<x2> <y2>}	Specifies the lower-right corner of the rectangle that envelops the ellipse.
-plot <stringValue>	Name of the plot where the ellipse will be drawn. If not specified, the command uses the selected plot.

Returns

Returns a string naming the ellipse. The ellipse will be numbered if other ellipses are already drawn.

Example

```
draw_ellipse -plot Plot_XY -p1 {0 0.5} -p2 {0.75 0.25}  
#-> Ellipse_1
```

Appendix A: Tcl Commands

draw_line

draw_line

Draws a line connecting two points.

Note:

This command applies to xy and 2D plots only.

Syntax

```
draw_line -p1 <doubleList> -p2 <doubleList>
           [-plot <stringValue>]
```

Argument	Description
-p1 <doubleList>	List of double values representing a point in the plot. This point is the first point of the line.
-p2 <doubleList>	List of double values representing a point in the plot. This point is the second point of the line.
-plot <stringValue>	Name of the plot in which the line will be drawn.

Returns

Returns a string naming the line. The line will be numbered if other lines are already drawn.

Example

```
draw_line -plot Plot_n9_des -p1 {62.9161 49.8411} -p2 {138.117 60.5841}
#-> Line_1
```

Appendix A: Tcl Commands

draw_rectangle

draw_rectangle

Draws a rectangle in the current plot.

Note:

This command applies to xy and 2D plots only.

Syntax

```
draw_rectangle -p1 <doubleList> -p2 <doubleList>
[-plot <stringValue>]
```

Argument	Description
-p1 <doubleList>	List of double values representing the upper-left corner of the rectangle.
-p2 <doubleList>	List of double values representing the lower-right corner of the rectangle.
-plot <stringValue>	Name of the plot in which the rectangle will be drawn.

Returns

Returns a string naming the rectangle. The rectangle will be numbered if other rectangles are already drawn.

Example

```
draw_rectangle -plot Plot_n9_des -p1 {46.0341 38.0749} \
               -p2 {97.1916 112.253}
#-> Plane_1
```

Appendix A: Tcl Commands

draw_textbox

draw_textbox

Draws a text box with a label at a position indicated by the `-at` argument, and inserts an arrow that points to the direction of the text box indicated by the `-anchor` argument.

Note:

This command applies to xy and 2D plots only. The arrow and its properties work only in 2D plots.

Syntax

```
draw_textbox -at <doubleList> -label <stringValue>
              [-anchor <doubleList>] [-plot <stringValue>]
```

Argument	Description
<code>-at <doubleList></code>	List of two double values indicating the lower-right corner of the text box.
<code>-label <stringValue></code>	Specifies the text to be displayed in the text box.
<code>-anchor <doubleList></code>	List of double values representing a position where the arrow will point to.
<code>-plot <stringValue></code>	Name of the plot in which the text box will be drawn.

Returns

Returns a string naming the text box. The text box will be numbered if other text boxes are already drawn.

Example

```
draw_textbox -at {219.458 41.6559} -anchor {219.458 41.1443} -label Text
#-> Text
```

Appendix A: Tcl Commands

echo

echo

Prints a string in the Console.

Syntax

```
echo [<stringValue>]
```

Argument	Description
<stringValue>	Specify the text to be printed in the Console.

Returns

None.

Example

```
echo "Hello World"  
#-> Hello World
```

exit

Exits Sentaurus Visual with the status given as an argument.

Syntax

```
exit [<intValue>]
```

Argument	Description
<intValue>	Exit code as an integer. Default: 0.

Returns

None.

Example

```
exit 1  
#-> Exit status: 1
```

Appendix A: Tcl Commands

export_curves

export_curves

Exports a curve to the specified file format.

Syntax

```
export_curves -filename <stringValue> -plot <stringValue>
    [<stringList>] [-format csv | plx] [-overwrite]
```

Argument	Description
-filename <stringValue>	Name of the exported file or files.
-plot <stringValue>	Exports the curves from the specified plot.
<stringList>	Specifies a list of curves to be exported.
-format csv plx	Format of the output file. Default: csv.
-overwrite	Overwrites existing files if specified.

Returns

Integer.

Example

```
export_curves -plot Plot_1 -filename testFile.csv -format csv
#-> 0
```

Appendix A: Tcl Commands

export_movie

export_movie

Creates a new movie by exporting the selected frames into a GIF file.

Note:

You must use the `start_movie` and `add_frame` commands before using the `export_movie` command.

Syntax

```
export_movie -filename <stringValue>
    [-frame_duration <intValue>]
    [-frames <stringList>] [-overwrite]
```

Argument	Description
<code>-filename <stringValue></code>	Name of the file for the new movie. Add <code>.gif</code> extension if necessary.
<code>-frame_duration <intValue></code>	Specifies the duration of each frame with 1/100 s as unit. Default: 50.
<code>-frames <stringList></code>	Specifies a list of frames to be exported. The entire frame buffer is used by default.
<code>-overwrite</code>	Overwrites existing files if specified.

Returns

String.

Example

```
export_movie -filename Movie.gif -frame_duration 2 -overwrite
#-> Movie.gif
```

See Also

[add_frame on page 218](#)
[start_movie on page 382](#)

Appendix A: Tcl Commands

export_settings

export_settings

Exports Sentaurus Visual settings to a file.

Syntax

```
export_settings <stringValue>
```

Argument	Description
<stringValue>	Specifies the name of the file. It must have the file extension .conf.

Returns

Integer.

Example

```
export_settings settings.conf  
#-> 0
```

export_variables

Exports variables from a curve to a file.

Syntax

```
export_variables -dataset <stringValue> -filename <stringValue>  
[-overwrite] [<stringList>]
```

Argument	Description
-dataset <stringValue>	Specifies the name of the dataset where specified variables (by default, all) are read for export.
-filename <stringValue>	Specifies the path of the exported file.
-overwrite	Overwrites existing files if specified.
<stringList>	Specifies a list of variables to be saved. If not specified, the command exports all variables from the dataset provided.

Returns

Integer.

Appendix A: Tcl Commands

export_view

Example

```
export_variables -dataset Data_1 -filename exportedVars.csv  
#-> 0
```

export_view

Exports a plot to the specified file format.

If `-plots` is used, the command exports only the specified plots. If it is not specified, the command exports all plots.

Syntax

```
export_view <stringValue>  
  [-format <stringValue>] [-overwrite] [-plots <stringList>]  
  [-resolution <width>x<height>]
```

Argument	Description
<code><stringValue></code>	Name of the exported file or files.
<code>-format <stringValue></code>	Specifies the type of file format to use when exporting the plots. The supported formats are BMP, EPS, JPEG, JPG, PNG, PPM, TIF, TIFF, XBM, and XPM.
<code>-overwrite</code>	Overwrites existing files if specified.
<code>-plots <stringList></code>	Exports the list of plots specified. If not specified, the command exports all the plots.
<code>-resolution <width>x<height></code>	Specifies the output resolution in pixels.

Returns

Integer.

Example

```
export_view /path/to/examplePlot.bmp -format bmp  
#-> 0
```

Appendix A: Tcl Commands

extract_path

extract_path

Extracts the path of the minimum or maximum values of a scalar field. This command returns the name of a new geometry that is added automatically to the plot. See [Extracting the Path of Minimum or Maximum Values of a Scalar Field on page 191](#).

Syntax

```
extract_path <stringValue> -max | -min  
[-bottom <doubleValue>] [-geom <stringValue>]  
[-materials <stringList> | -regions <stringList>]  
[-plot <stringValue>] [-points <pointList>]  
[-start {<x> <y> <z>} -end {<x> <y> <z>}]  
[-top <doubleValue>] [-type x | y | z]  
[-window {<x1> <y1> [<z1>] <x2> <y2> [<z2>]}]
```

Argument	Description
<stringValue>	Name of the scalar field whose values are extracted along a path.
-max -min	Specifies whether the command extracts the maximum or minimum values of the scalar field.
-bottom <doubleValue>	Sets the axis coordinate of the bottom face of the extruded polygon.
-geom <stringValue>	Specifies the dataset (or geometry) where the command will search for the scalar field. If not specified, the command uses the main one from the active plot.
-materials <stringList> -regions <stringList>	Specifies either a list of materials or a list of regions where the field values will be extracted. If not specified, the command uses the entire plot.
-plot <stringValue>	Name of the plot. If not specified, the command uses the selected plot.
-points <pointList>	Specifies the coordinates of each vertex of the polygon to extrude. The list of coordinates must be ordered since each point will be interpolated.
-start {<x> <y> <z>} -end {<x> <y> <z>}	Specifies the start point and the end point of the 3D path to be extracted. Available for 3D plots only.
-top <doubleValue>	Sets the axis coordinate of the top face of the extruded polygon.

Appendix A: Tcl Commands

extract_path

Argument	Description
<code>-type x y z</code>	Selects the axis where the polygon specified by <code>-points</code> should be extruded.
<code>-window {<x1> <y1> [<z1>] <x2> <y2> [<z2>]}</code>	Specifies a window defined by <code>x1, y1, [z1]</code> and <code>x2, y2, [z2]</code> . These values must be specified in Cartesian coordinates. If not specified, the command uses the entire plot.

Returns

String.

Example

```
extract_path ElectrostaticPotential -plot Plot_2D -geom 2D \
    -materials {Oxide Silicon} -window {-10.32 0 10 10} -max
#-> PathGeometry_2D
```

Appendix A: Tcl Commands

extract_streamlines

extract_streamlines

Extracts the fields and coordinates data from one or more streamlines created in 2D or 3D plots.

Note:

This command applies to 2D and 3D plots only.

Syntax

```
extract_streamlines <stringList> [-plot <stringValue>]
```

Argument	Description
<stringList>	Specifies a list of streamlines from which to extract data.
-plot <stringValue>	Name of the plot. If not specified, the command uses the selected plot.

Returns

List (names of the 1D datasets created).

Example

```
extract_streamlines {Streamline} -plot Plot_2D  
#-> {Streamline(Plot_2D)}  
extract_streamlines {Streamline_1 Streamline_2 Streamline_3} \  
-plot Plot_2D  
#-> {Streamline_1(Plot_2D) Streamline_1(Plot_2D) Streamline_1(Plot_2D)}
```

Appendix A: Tcl Commands

extract_value

extract_value

Calculates the minimum and maximum field value of a defined polyhedron and shows (with a marker) the location of these values.

Note:

This command applies to 3D plots only.

Syntax

```
extract_value -min | -max
[   -bottom <doubleValue>] [-dataset <stringValue>]
[   -field <stringValue>] [-geom <stringValue>]
[   -materials <stringList> | -regions <stringList>]
[   -plot <stringValue>] [-points <pointList>]
[   -top <doubleValue>] [-type x | y | z]
```

Argument	Description
-min -max	Selects whether the point returned is the minimum or maximum value.
-bottom <doubleValue>	Sets the value of the height bottom of the vertical axis of the polyhedron.
-dataset <stringValue>	Selects the dataset (for batch mode only) from which the data should be extracted.
-field <stringValue>	Name of the field from which the data is obtained. If not specified, the command uses the current contour-band field displayed in the plot.
-geom <stringValue>	Selects the geometry from which the data is extracted.
-materials <stringValue> -regions <stringList>	Sets a list of materials or regions that will be used to find the minimum or maximum value.
-plot <stringValue>	Selects the plot from which the data should be extracted.
-points <pointList>	Selects the points in the structure defining the polygon to construct the polyhedron. Points must be defined as {"x0 y0 z0" "x1 y1 z1" ...}.
-top <doubleValue>	Sets the value of the height top of the vertical axis of the polyhedron.
-type x y z	Specifies the extrusion axis for the polyhedron.

Appendix A: Tcl Commands

extract_value

Returns

Double and a list of coordinate values.

Example

```
extract_value -plot Plot_n10_des -field Abs(TotalCurrentDensity-V)
  -min -type z -top 0.05 -bottom 0
  -points {{0 -0.04 0.05} {0 0.04 0.05} {0.1 0.04 0.05}
            {0.1 -0.04 0.05}}
# position:
# min: 0.00781 0.01 0.00625 value -6.41e+18
# max: 1.92e-5 0.03 0.05 value 6e+20
#-> -6.41e+18 {0.00781 0.01 0.00625}
```

Appendix A: Tcl Commands

find_values

find_values

Returns the positions of all points with the specified value in a scalar field.

Note:

This command applies to 2D and 3D plots only.

Syntax

```
find_values <doubleValue> -field <stringValue>
    [-geom <stringValue>] [-plot <stringValue>]
    [-window {<x1> <y1> [<z1>] <x2> <y2> [<z2>]}]
```

Argument	Description
<doubleValue>	Specifies a value that must be found in the field (floating-point number).
-field <stringValue>	Name of a scalar field.
-geom <stringValue>	Specifies the dataset (or geometry) where the command searches for the scalar field. If not specified, the command uses the main one from the active plot.
-plot <stringValue>	Name of the plot. If not specified, the command uses the selected plot.
-window {<x1> <y1> [<z1>] <x2> <y2> [<z2>]}	Specifies a window whose values must be specified in Cartesian coordinates. If not specified, the command uses the entire geometry.

Returns

Positions of all points in the specified field with the specified value.

Example

```
find_values 1.0 -field ElectrostaticPotential
#-> {3.53079 -0.0263978 0.392817} {6.41207 -1.7982327 3.289376}
{2.76445 -9.8237643 9.238764}
```

Appendix A: Tcl Commands

get_axis_prop

get_axis_prop

Returns axis properties.

Note:

The command returns only one property at a time for the specified axis.

Syntax

```
get_axis_prop -axis x | y | z | y2
(
    -anchor | -auto_padding | -auto_precision | -auto_spacing |
    -inverted |
    -label_angle | -label_font_att | -label_font_color |
    -label_font_factor | -label_font_family | -label_font_size |
    -label_format | -label_padding | -label_precision |
    -major_ticks_length | -major_ticks_width |
    -max | -max_fixed | -min | -min_fixed |
    -min | -min_fixed | -minor_ticks_length | -minor_ticks_position |
    -minor_ticks_width | -nof_minor_ticks | -padding | -range |
    -show | -show_minor_ticks | -show_label | -show_ticks |
    -show_title | -spacing | -ticks_position | -title |
    -title_font_att | -title_font_color | -title_font_factor |
    -title_font_family | -title_font_size | -type
)
    [-plot <stringValue>]
```

Argument	Description
-axis x y z y2	Specifies from which axis the properties will be returned. The axis identifier <code>y2</code> is valid only for xy plots.
-anchor	Starting point where ticks are computed.
-auto_padding	Shows whether automatic padding of the axis is active (applies to xy plots only).
-auto_precision	Returns the property if the precision of the axis is automatic.
-auto_spacing	Shows if automatic ticks spacing is used.
-inverted	Shows inverted axis.
-label_angle	Rotation angle applied to the labels of major ticks (applies to xy plots only).
-label_font_att	Font attributes of the axis.

Appendix A: Tcl Commands

get_axis_prop

Argument	Description
-label_font_color	Color of the axis.
-label_font_factor	Size factor of the axis (applies to 2D and 3D plots only).
-label_font_family	Font of the axis.
-label_font_size	Font size of the axis (applies to xy plots only).
-label_format	Numeric format of the axis.
-label_padding	Padding of the axis.
-label_precision	Precision of the axis.
-major_ticks_length	Length of the major ticks.
-major_ticks_width	Width of the major ticks (applies to xy plots only).
-max	Returns the maximum value of the axis.
-max_fixed	Shows whether the maximum value of the axis is fixed.
-min	Returns the minimum value of the axis.
-min_fixed	Shows whether the minimum value of the axis is fixed.
-minor_ticks_length	Length of the minor ticks.
-minor_ticks_position	Shows the position of minor ticks (in, out, or center) (applies to 2D plots only).
-minor_ticks_width	Width of the minor ticks (applies to xy plots only).
-nof_minor_ticks	Number of minor ticks on the selected axis.
-padding	Padding value of the axis scale, in pixels (applies to xy plots only).
-range	Range covered on the axis.
-show	Shows axis.
-show_minor_ticks	Shows minor ticks.
-show_label	Shows labels if values for the major ticks are displayed.

Appendix A: Tcl Commands

get_axis_prop

Argument	Description
-show_ticks	Shows major ticks.
-show_title	Shows title.
-spacing	Shows the spacing between two major ticks.
-ticks_position	Shows the position of major ticks (in, out, or center).
-title	Axis label.
-title_font_att	Font attributes of the axis label.
-title_font_color	Color of the axis label.
-title_font_factor	Font size factor of the axis label (applies to 2D and 3D plots only).
-title_font_family	Font of the axis title.
-title_font_size	Font size of the axis label (applies to xy plots only).
-type	Scale type: linear or logarithmic.
-plot <stringValue>	Name of the plot from where the axis properties will be returned.

Returns

The value of the queried property.

Example

```
get_axis_prop -axis x -type  
#-> linear
```

Appendix A: Tcl Commands

get_camera_prop

get_camera_prop

Returns camera properties.

Note:

The command returns only one property at a time and applies to 3D plots only.

Syntax

```
get_camera_prop
(
    -dolly_zoom_on | -focal_point | -parallel | -position | -rot_color |
    -rot_size | -rot_width | -rotation_point | -setup |
    -show_rotation_point | -view_angle | -view_up
)
    [-plot <stringValue>]
```

Argument	Description
-dolly_zoom_on	Returns whether dolly zoom is switched on.
-focal_point	Returns position of the focal point.
-parallel	Returns whether parallel projection mode is active.
-position	Returns position of the camera.
-rot_color	Returns color of the rotation point.
-rot_size	Returns size of the rotation point.
-rot_width	Returns width of the lines of the rotation point.
-rotation_point	Returns position of the rotation point.
-setup	Returns the current setup of the camera.
-show_rotation_point	Shows rotation point.
-view_angle	Shows the actual view angle of the camera in degrees.
-view_up	Returns the view-up vector of the camera.
-plot <stringValue>	Name of the plot from which to obtain the required property. If not specified, the command uses the selected plot.

Appendix A: Tcl Commands

get_contour_labels_prop

Returns

The value of the queried property.

Example

```
get_camera_prop -position  
#-> 3.53079 -0.0263978 0.392817
```

get_contour_labels_prop

Returns properties of the contour labels.

Note:

The command returns only one property at a time.

Syntax

```
get_contour_labels_prop <stringValue>  
(  
    -anchor | -color_bg | -font_att | -font_color | -font_factor |  
    -font_family | -format | -level_skip | -precision | -show_bg |  
    -show_index | -spacing  
)  
    [-geom <stringValue>] [-plot <stringValue>]
```

Argument	Description
<stringValue>	Name of the field. If not specified, then the command uses the active field.
-anchor	Anchor of the contour labels.
-color_bg	Background color of the contour labels.
-font_att	Font attribute of the contour labels.
-font_color	Font color of the contour labels.
-font_factor	Font multiplier of the contour labels.
-font_family	Font type of the contour labels.
-format	Numeric format of the contour labels.
-level_skip	Returns level skip of the contour labels. The value 0 means every level is labeled.

Appendix A: Tcl Commands

get_contour_labels_prop

Argument	Description
-precision	Precision of the contour labels.
-show_bg	Shows whether contour labels background is visible.
-show_index	Shows whether contour values are shown as indexes.
-spacing	Spacing between contour labels in pixels.
-geom <stringValue>	Name of the dataset (or geometry). If not specified, then the command uses the main one from the active plot.
-plot <stringValue>	Name of the plot. If not specified, then the command uses the active plot.

Returns

The value of the queried property.

Example

```
get_contour_labels_prop DopingConcentration -format  
#-> engineering
```

Appendix A: Tcl Commands

get_curve_data

get_curve_data

Returns data from the specified curve axis.

Syntax

```
get_curve_data <stringValue> -axisX | -axisY | -scalar  
[-plot <stringValue>]
```

Argument	Description
<stringValue>	Name of the curve from where data is retrieved.
-axisX -axisY -scalar	Curve axis or scalar field from where data is retrieved.
-plot <stringValue>	Name of plot where curve is displayed.

Returns

The value of the queried property.

Example

```
get_curve_data Curve_1 -axisX  
#-> 0 0.05 0.1 0.15 0.2 0.25 0.3 0.35 0.4 0.45 0.5
```

Appendix A: Tcl Commands

get_curve_prop

get_curve_prop

Returns curve properties.

Note:

The command returns only one property at a time and applies to xy plots only.

Syntax

```
get_curve_prop <stringValue>
(
    -axis | -color | -deriv | -function | -integ | -label | -line_style |
    -line_width | -markers_size | -markers_type | -selected | -show |
    -show_legend | -show_line | -show_markers | -xScale | -xShift |
    -yScale | -yShift
)
    [-plot <stringValue>]
```

Argument	Description
<stringValue>	Identifier of the curve.
-axis	Shows axis alignment of selected curve (left or right aligned).
-color	Shows the selected curve line color.
-deriv	Shows the order of the derivative applied to the curve.
-function	Shows a function applied to the curve.
-integ	Shows the integration applied to the curve.
-label	Shows the selected curve label.
-line_style	Shows the line style of the curve.
-line_width	Shows the selected curve line width.
-markers_size	Shows the selected curve markers size.
-markers_type	Shows the selected curve markers type.
-selected	Shows whether the curve is selected.
-show	Shows curve.
-show_legend	Shows whether the legend is visible.

Appendix A: Tcl Commands

get_curve_prop

Argument	Description
-show_line	Shows curve line.
-show_markers	Shows activated markers.
-xScale	Shows the scale of the curve in the x-axis.
-xShift	Shows the shift of the curve in the x-axis.
-yScale	Shows the scale of the curve in the y-axis.
-yShift	Shows the shift of the curve in the y-axis.
-plot <stringValue>	Name of the plot. If not specified, the command uses the selected plot.

Returns

The value of the queried property.

Example

```
get_curve_prop Curve_1 -deriv  
#-> 2
```

Appendix A: Tcl Commands

get_cutline_prop

get_cutline_prop

Returns the properties of cutlines.

Note:

This command returns only one property at a time. It applies to 2D and 3D plots only.

Syntax

```
get_cutline_prop <stringValue> -plot <stringValue>
(
    -handles_color | -hide_handles | -label_op | -label_pos |
    -label_size | -line_color | -line_style | -line_width |
    -pos1 | -pos2 | -show_handles | -show_label
)
```

Argument	Description
<stringValue>	Name of the cutline from which the property will be returned.
-plot <stringValue>	Name of the plot in which the cutline is located.
-handles_color	Color of the handles.
-hide_handles	Shows whether the handles are hidden.
-label_op	The side of the cutline where the label is displayed. It returns false (0) if it is the “normal” position or true (1) if it is the opposite position.
-label_pos	Position of the label.
-label_size	Size of the label.
-line_color	Color of the line.
-line_style	Style of the line.
-line_width	Width of the line.
-pos1	Position of the first point.
-pos2	Position of the second point.
-show_handles	The status of the visibility of the handles.

Appendix A: Tcl Commands

get_cutline_prop

Argument	Description
<code>-show_label</code>	The status of the visibility of the label.
<code>-handles_color</code>	Color of the handles.

Returns

The value of the queried property.

Example

```
get_cutline_prop C1 -plot Plot_2D -pos1  
#-> 0.1 2.7891 0
```

Appendix A: Tcl Commands

get_cutplane_prop

get_cutplane_prop

Returns cutplane properties.

Note:

This command applies to 3D plots only.

Syntax

```
get_cutplane_prop <stringValue> -plot <stringValue>
(
    -at | -hide_label | -label_position | -label_size | -normal |
    -origin | -show_label
)
```

Argument	Description
<stringValue>	Name of the cutplane from which the property is returned.
-plot <stringValue>	Name of the plot in which the cutplane is located.
-at	Position value of an -at type cutplane.
-hide_label	Shows whether the label in the cutplane is hidden.
-label_position	Position of the label.
-label_size	Size of the label.
-normal	The normal vector of a free-type cutplane.
-origin	The origin vector of a free-type cutplane.
-show_label	The status of the visibility of the label.

Returns

The value of the queried property.

Example

```
get_cutplane_prop C1 -plot Plot_3D -at
#-> 0.483
```

Appendix A: Tcl Commands

get_ellipse_prop

get_ellipse_prop

Returns ellipse properties.

Note:

This command returns only one property at a time and applies only to xy plots.

Syntax

```
get_ellipse_prop <stringValue>
  ( -fill_color | -line_color | -line_style | -line_width | -p1 | -p2 )
  [-plot <stringValue>]
```

Argument	Description
<stringValue>	Name of the ellipse object from which the property will be returned.
-fill_color	Returns the color inside of the ellipse.
-line_color	Returns the line color.
-line_style	Returns the line pattern.
-line_width	Returns the line width.
-p1	Returns a list of double values representing the start point of the ellipse (upper-left corner).
-p2	Returns a list of double values representing the end point of the ellipse (lower-right corner).
-plot <stringValue>	Name of the plot. If not specified, the command uses the selected plot.

Returns

The value of the queried property.

Example

```
get_ellipse_prop Ellipse_1 -plot Plot_1 -line_style
#-> dashdotdot
```

Appendix A: Tcl Commands

get_field_prop

get_field_prop

Returns field properties.

Note:

The command returns only one property at a time and applies only to 2D and 3D plots.

Syntax

```
get_field_prop [<stringValue>]
(
    -contour_labels_on | -custom_levels | -interpolated_values | -label |
    -levels | -line_color | -line_width | -max | -max_fixed | -min |
    -min_fixed | -range | -scale | -show | -show_bands
)
    [-geom <stringValue>] [-plot <stringValue>]
```

Argument	Description
<stringValue>	Name of the field. If not specified, the command uses the active field.
-contour_labels_on	Shows contour labels of the selected field.
-custom_levels	Shows custom levels defined for the selected field.
-interpolated_values	Returns whether the interpolated values on its vertices are used for visualization (this property is only valid for fields defined on cells).
-label	Label of the field.
-levels	Levels of the selected field.
-line_color	Color of the contour lines.
-line_width	Width of the contour lines.
-max	Maximum value of the field.
-max_fixed	Shows whether the maximum value of the field is fixed.
-min	Minimum value of the field.
-min_fixed	Shows whether the minimum value of the field is fixed.
-range	Range of the selected field.

Appendix A: Tcl Commands

get_field_prop

Argument	Description
-scale	Scale of the selected field.
-show	Shows the selected field on the plot.
-show_bands	Shows bands.
-geom <stringValue>	Name of the dataset (or geometry). If not specified, the command uses the main one from the active plot.
-plot <stringValue>	Name of the plot. If not specified, the command uses the active plot.

Returns

The value of the queried property.

Example

```
get_field_prop -range  
#-> {2.36618e-05 4.36902e+06}
```

Appendix A: Tcl Commands

get_grid_prop

get_grid_prop

Returns grid properties of a plot.

Note:

The command applies to xy and 2D plots only.

Syntax

```
get_grid_prop
(
    -align | -line1_color | -line1_style | -line1_width | -line2_color |
    -line2_style | -line2_width | -show | -show_minor_lines
)
[-plot <stringValue>]
```

Argument	Description
-align	Shows plot alignment (applies to xy plots only).
-line1_color	Shows color of major lines.
-line1_style	Shows style of major lines (applies to xy plots only).
-line1_width	Shows width of major lines.
-line2_color	Shows color of minor lines.
-line2_style	Shows style of minor lines (applies to xy plots only).
-line2_width	Shows width of minor lines.
-show	Shows major lines.
-show_minor_lines	Shows minor lines.
-plot <stringValue>	Name of the plot. If not specified, the command uses the selected plot.

Returns

The value of the queried property.

Example

```
get_grid_prop -line1_color
#-> #ff0000
```

Appendix A: Tcl Commands

get_input_data

get_input_data

Displays a customizable dialog box that requires user input.

Note:

This command works only in interactive mode. Otherwise, it fails unless a default value is provided.

Syntax

```
get_input_data [-default <stringValue>] [-desc <stringValue>]
               [-double | -file | -int | -list | -string | -yesno]
               [-elements <stringList>]
```

Argument	Description
-default <stringValue>	Sets the default value of a variable.
-desc <stringValue>	Text that describes the input.
-double -file -int -list -string -yesno	Sets the type of input to be expected. Options are: <ul style="list-style-type: none">• -double: Sets the input to expect a double number.• -file: Sets the input to be selected from the file dialog box.• -int: Sets the input to expect an integer.• -list: Sets the input to be selected from a list.• -string: Sets the input to expect a string. This is the default if no type input is specified.• -yesno: Sets the input to expect a Yes or No value. It returns 1 for Yes and 0 for No.
-elements <stringList>	List of possible elements.

Returns

The result of the input as a string.

If you cancel the input, the result is equal to the default value. If a default value is not specified, the command generates an error message.

Example

```
get_input_data -desc "Variable that defines the cut point." \
               -default 0.0 -double
#-> 1.5
```

Appendix A: Tcl Commands

get_legend_prop

get_legend_prop

Returns legend properties.

Note:

The command returns only one property at a time.

Syntax

```
get_legend_prop
(
    -color_bg | -color_fg | -label_font_att | -label_font_color |
    -label_font_factor | -label_font_family | -label_font_size |
    -label_format | -location | -margins | -nof_labels | -orientation |
    -position | -precision | -show_background | -size | -title_font_att |
    -title_font_color | -title_font_factor | -title_font_family
)
[-plot <stringValue>]
```

Argument	Description
-color_bg	Background color of the legend.
-color_fg	Foreground color of the legend.
-label_font_att	Font attribute of the legend labels.
-label_font_color	Font color of the legend labels.
-label_font_factor	Font multiplier of the legend labels.
-label_font_family	Font type of the legend labels.
-label_font_size	Font size of the legend labels.
-label_format	Numeric format of the legend labels.
-location	Location in the plot area where the legend is displayed (only for xy plots).
-margins	Margins of the legend box.
-nof_labels	Number of labels used in the legend.
-orientation	Orientation of the legend.
-position	Position of the legend that is normalized to the window coordinates between 0 and 1.

Appendix A: Tcl Commands

get_legend_prop

Argument	Description
-precision	Precision of the legend labels.
-show_background	Shows whether legend is transparent.
-size	Legend size is normalized to window coordinates.
-title_font_att	Font attribute of the legend title.
-title_font_color	Font color of the legend title.
-title_font_factor	Font multiplier of the legend title.
-title_font_family	Font type of the legend title.
-plot <stringValue>	Name of the plot. If not specified, the command uses the selected plot.

Returns

The value of the queried property.

Example

```
get_legend_prop -orientation  
#-> vertical
```

Appendix A: Tcl Commands

get_line_prop

get_line_prop

Returns line properties.

Note:

This command returns only one property at a time and applies only to xy and 2D plots.

Syntax

```
get_line_prop <stringValue>
  (-line_color | -line_style | -line_width | -p1 | -p2 )
  [-plot <stringValue>]
```

Argument	Description
<stringValue>	Name of the line object from which the property is returned.
-line_color	Returns the line color.
-line_style	Returns the line pattern (applies only to xy plots).
-line_width	Returns the line width in pixels.
-p1	Returns a list of double values representing the start point of the line.
-p2	Returns a list of double values representing the end point of the line.
-plot <stringValue>	Name of the plot. If not specified, the command uses the selected plot.

Returns

The value of the queried property.

Example

```
get_line_prop Line_1 -plot Plot_1 -line_width
#-> 4
```

Appendix A: Tcl Commands

get_material_prop

get_material_prop

Returns material properties.

Note:

The command returns only one property at a time and applies only to 2D and 3D plots.

Syntax

```
get_material_prop <stringValue>
(
    -border_color | -border_width | -color | -mesh_color | -mesh_width |
    -on | -particles_size | -show_all | -show_border |
    -show_bulk | -show_field | -show_mesh | -show_vector |
    -translucency_level | -translucency_on
)
    [-geom <stringValue>] [-plot <stringValue>]
```

Argument	Description
<stringValue>	Name of the material.
-border_color	Color of material border.
-border_width	Width of material border in pixels.
-color	Material bulk color.
-mesh_color	Color of the material mesh.
-mesh_width	Width of the material mesh in pixels.
-on	Status of the visibility of the material.
-particles_size	Size of the particles.
-show_all	Shows the visibility status of the material.
-show_border	Shows the visibility status of the material border.
-show_bulk	Shows the visibility status of the material bulk.
-show_field	Shows the scalar field visibility status of the material.
-show_mesh	Shows the visibility status of the material mesh.

Appendix A: Tcl Commands

get_material_prop

Argument	Description
-show_vector	Shows the status of the vector field visibility of the material.
-translucency_level	Translucency level of the material.
-translucency_on	Shows whether translucency is activated.
-geom <stringValue>	Specifies the geometry from where the material property will be requested. If not specified, the command uses first geometry of given -plot.
-plot <stringValue>	Name of the plot from where the material property will be requested. If not specified, the command uses selected plot.

Returns

The value of the queried property.

Example

```
get_material_prop Silicon -plot Plot_2D -show_mesh  
#-> false
```

Appendix A: Tcl Commands

get_plot_prop

get_plot_prop

Returns plot properties.

Note:

The command returns only one property at a time.

Syntax

```
get_plot_prop
(
    -axes_interchanged | -bg_gradient | -caption_font_size |
    -caption_leader_on | -caption_material | -color_bg | -color_fg |
    -color_map | -enable_path_limit | -frame_width |
    -gradient_colors | -keep_aspect_ratio | -material_colors |
    -path_depth | -ratio_xtoy | -show | -show_axes |
    -show_axes_label | -show_axes_title | -show_cube_axes |
    -show_curve_lines | -show_curve_markers | -show_grid |
    -show_legend | -show_major_ticks |
    -show_max_marker | -show_min_marker | -show_minor_ticks |
    -show_path | -show_regions_bg | -show_title |
    -tdr_state | -tdr_state_index | -title |
    -title_font_att | -title_font_color | -title_font_factor |
    -title_font_family | -title_font_size | -transformation
)
[-plot <stringValue>]
```

Argument	Description
-axes_interchanged	Returns a value indicating whether the axes are interchanged (only for 2D plots).
-bg_gradient	Returns a value indicating whether the background displays a gradient (2D and 3D plots only).
-caption_font_size	Returns the size of the caption font (3D plots only).
-caption_leader_on	Returns whether the leaders of the captions are displayed (3D plots only).
-caption_material	Returns whether material names or region names are displayed in the caption.
-color_bg	Shows the background color used when a solid background is active.
-color_fg	Shows the foreground color used.

Appendix A: Tcl Commands

get_plot_prop

Argument	Description
-color_map	Returns a value indicating whether the color map is in default mode (full palette) or grayscale mode.
-enable_path_limit	Returns a value indicating whether the path is limited.
-frame_width	Returns a value that is a positive integer less than 8, denoting the frame width in pixels.
-gradient_colors	Returns a list of colors used when a gradient background is active (2D and 3D plots only).
-keep_aspect_ratio	Returns a value indicating whether the aspect ratio is maintained (only for 2D plots).
-material_colors	Returns a value indicating whether the color scheme is Classic or Vivid.
-path_depth	Returns an integer with the number of path directory names to be displayed in the plot title.
-ratio_xtoy	Returns the transformation ratio between the x-axis and y-axis (only for 2D plots).
-show	Returns a value indicating whether the plot is displayed.
-show_axes	Returns a value indicating whether the axes are present.
-show_axes_label	Returns a value indicating whether axes labels are present (only for xy plots).
-show_axes_title	Returns a value indicating whether the axes titles are displayed (only for xy plots).
-show_cube_axes	Returns a value indicating whether cube axes are displayed (only for 3D plots).
-show_curve_lines	Returns a value indicating whether the curve lines are displayed (only for xy plots).
-show_curve_markers	Returns a value indicating whether the markers are enabled (only for xy plots).
-show_grid	Returns a value indicating whether the grid is displayed (only for xy and 2D plots).
-show_legend	Returns a value indicating whether legend is displayed.

Appendix A: Tcl Commands

get_plot_prop

Argument	Description
-show_major_ticks	Returns a value indicating whether the major ticks are displayed (only for 3D plots).
-show_max_marker	Returns a value indicating whether the maximum marker is displayed (only for 2D and 3D plots).
-show_min_marker	Returns a value indicating whether the minimum marker is displayed (only for 2D and 3D plots).
-show_minor_ticks	Returns a value indicating whether the minor ticks are displayed (only for 3D plots).
-show_path	Returns a value indicating whether the path is displayed.
-show_regions_bg	Returns a value indicating whether the background color of regions is displayed.
-show_title	Returns a value indicating whether title is displayed.
-tdr_state	Returns the name of the TDR state currently displayed in the plot.
-tdr_state_index	Returns the index of the TDR state currently displayed in the plot.
-title	Shows the plot label.
-title_font_att	Shows the plot label attribute (normal, bold, italic, underline or strikeout).
-title_font_color	Shows the plot label color used.
-title_font_factor	Returns a value indicating the font factor size (only for 2D and 3D plots).
-title_font_family	Shows the label font family used (arial, courier or times).
-title_font_size	Returns a value indicating the label font size (only for xy plots).
-transformation	Shows the transformation used (only for 3D plots).
-plot <stringValue>	Name of the plot. If not specified, the command uses the selected plot.

Returns

The value of the queried property.

Appendix A: Tcl Commands

get_rectangle_prop

Example

```
get_plot_prop -title  
#-> Test Plot
```

get_rectangle_prop

Returns rectangle properties.

Note:

This command returns only one property at a time and applies only to xy plots and 2D plots.

Syntax

```
get_rectangle_prop <stringValue>  
(  
    -fill_color | -line_color | -line_style | -line_width | -p1 | -p2  
)  
    [-plot <stringValue>]
```

Argument	Description
<stringValue>	Name of the rectangle object from which the property is returned.
-fill_color	Returns the color inside of the rectangle.
-line_color	Returns the line color.
-line_style	Returns the line pattern.
-line_width	Returns the line width.
-p1	Returns a list of double values representing the start point of the rectangle (lower-left corner in 2D plots or upper-left corner in xy plots).
-p2	Returns a list of double values representing the end point of the rectangle (upper-right corner in 2D plots or lower-right corner in xy plots).
-plot <stringValue>	Name of the plot. If not specified, the command uses the selected plot.

Returns

The value of the queried property.

Appendix A: Tcl Commands

get_region_prop

Example

```
get_rectangle_prop Rectangle_1 -plot Plot_1 -fill_color  
#-> #00FF3B
```

get_region_prop

Returns region properties.

Note:

The command returns only one property at a time and applies only to 2D and 3D plots.

Syntax

```
get_region_prop <stringValue>  
(  
    -border_color | -border_width | -color | -mesh_color | -mesh_width |  
    -on | -particles_size |  
    -show_all | -show_border | -show_bulk | -show_caption |  
    -show_field | -show_mesh | -show_vector |  
    -translucency_level | -translucency_on  
)  
    [-geom <stringValue>] [-plot <stringValue>]
```

Argument	Description
<stringValue>	Name of the region.
-border_color	Color of region border.
-border_width	Width of region border in pixels.
-color	Region bulk color.
-mesh_color	Color of the region mesh.
-mesh_width	Width of the region mesh in pixels.
-on	Status of the visibility of the material.
-particles_size	Current particles size. Applies only to particle (kinetic Monte Carlo) regions.
-show_all	Shows the visibility status of the region.
-show_border	Shows the visibility status of the region border.

Appendix A: Tcl Commands

get_region_prop

Argument	Description
-show_bulk	Shows the visibility status of the region bulk.
-show_caption	Shows the visibility status of the region caption.
-show_field	Shows the status of the scalar field visibility of the region.
-show_mesh	Shows the visibility status of the region mesh.
-show_vector	Shows the status of the vector field visibility of the region.
-translucency_level	Translucency level of the region.
-translucency_on	Shows whether translucency is activated.
-geom <stringValue>	Specifies the geometry from where the region property will be requested. If not specified, the command uses the first geometry of the given -plot.
-plot <stringValue>	Name of the plot from where the region property will be requested. If not specified, the command uses the selected plot.

Returns

The value of the queried property.

Example

```
get_region_prop RGate -plot Plot_2D -border_color  
#->#0000FF
```

Appendix A: Tcl Commands

get_ruler_prop

get_ruler_prop

Returns ruler properties.

Note:

This command returns only one property at a time.

Syntax

```
get_ruler_prop -color | -pos1 | -pos2 | -precision | -show_label |  
    -snap_on | -width  
    [-id <intValue>] [-plot <stringValue>]
```

Argument	Description
-color	Color of the ruler.
-pos1	First point of the ruler.
-pos2	Second point of the ruler.
-precision	Decimal precision of the measurements.
-show_label	Status of the ruler label.
-snap_on	Status of the snap-to-mesh feature.
-width	Width of the ruler in pixels.
-id <intValue>	The ID of the ruler where the command will search for properties. If not specified, the command uses the last selected ruler.
-plot <stringValue>	Name of the plot where the command will search for properties. If not specified, the command uses the selected plot.

Returns

The value of the queried property.

Example

```
get_ruler_prop -width  
#-> 4
```

Appendix A: Tcl Commands

get_streamline_prop

get_streamline_prop

Returns the specified property of a streamline.

Note:

This command applies to 2D and 3D plots only.

Syntax

```
get_streamline_prop <stringValue> -plot <stringValue>
(
    -arrow_angle | -arrow_color | -arrow_size | -arrow_step |
    -arrow_style | -arrow_width | -constant_arrow |
    -line_color | -line_resolution | -line_style |
    -line_width | -positive_direction | -show_arrows | -show_line
)
```

Argument	Description
<stringValue>	Name of the streamline from which the specified property will be returned.
-plot <stringValue>	Name of the plot from where the streamline property will be requested. If not specified, the command uses the selected plot.
-arrow_angle	Arrowhead angle.
-arrow_color	Color of the arrows.
-arrow_size	Size of the arrows.
-arrow_step	Step between arrows.
-arrow_style	Returns the style of the streamline arrowheads.
-arrow_width	Width of the arrowheads.
-constant_arrow	Returns whether the zoom for the streamline arrowheads is constant in the plot area.
-line_color	Color of the line.
-line_resolution	Distance between the points that conform the line.
-line_style	Line style.
-line_width	Width of the line.

Appendix A: Tcl Commands

get_streamline_prop

Argument	Description
-positive_direction	Shows whether the arrow is displayed in the normal view or inverted view.
-show_arrows	Shows whether arrows are visible.
-show_line	Shows whether the line is visible.

Returns

The value of the queried property.

Example

```
get_streamline_prop Streamline_1 -plot Plot_2D -arrow_angle  
#-> 30
```

Appendix A: Tcl Commands

get_textbox_prop

get_textbox_prop

Returns the specified property of a text box.

Note:

This command applies to xy and 2D plots only.

Syntax

```
get_textbox_prop <stringValue>
(
    -anchor_pos | -arrow_size | -font_att | -font_color | -font_factor |
    -font_family | -font_size | -line_color | -line_style | -line_width |
    -pos | -rotation | -show_anchor | -show_border | -text
)
    [-plot <stringValue>]
```

Argument	Description
<stringValue>	Name of the text box.
-anchor_pos	Returns anchor position using the world coordinate system (only for 2D plots).
-arrow_size	Returns arrow size (only for 2D plots).
-font_att	Returns font attribute of text box.
-font_color	Returns color of text font.
-font_factor	Returns multiplier for text font (only for 2D plots).
-font_family	Returns font family of text.
-font_size	Returns font size (only for xy plots).
-line_color	Returns line and arrow color (only for 2D plots).
-line_style	Returns the representation style of the text box line (only for 2D plots).
-line_width	Returns width of text box and anchor line (only for 2D plots).

Appendix A: Tcl Commands

get_textbox_prop

Argument	Description
-pos	Returns the lower-left corner position of the text box. For xy plots, it returns a point in the world coordinate system {x, y} (see Inserting Text on page 59 for a description of the world coordinate system). For 2D plots, it returns a relative normalized screen coordinates pair (from 0.0 to 1.0).
-rotation	Returns the rotation of the text box in degrees (only for xy plots).
-show_anchor	Returns true if the text box anchor is shown (only for 2D plots).
-show_border	Returns true if the text box border is shown (only for 2D plots).
-text	Returns text in text box.
-plot <stringValue>	Name of the plot where the command will search for the text box. If not specified, the command uses the selected plot.

Returns

The value of the queried property.

Example

```
get_textbox_prop Textbox_1 -text  
#-> Hello World
```

Appendix A: Tcl Commands

get_variable_data

get_variable_data

Returns a list of variable values.

Note:

This command applies to xy plots only.

Syntax

```
get_variable_data <stringValue> -dataset <stringValue>
```

Argument	Description
<stringValue>	Name of the variable.
-dataset <stringValue>	Name of the dataset.

Returns

List.

Example

```
get_variable_data X -dataset plotXY  
#-> 1 2 3 4 5
```

Appendix A: Tcl Commands

get_vector_prop

get_vector_prop

Returns the specified property of a vector field.

Note:

This command applies to 2D and 3D plots only.

Syntax

```
get_vector_prop <stringValue>
(
    -constant_heads | -fill_color | -head_angle | -head_shape |
    -head_size | -line_color | -line_pattern | -line_width |
    -scale | -scale_factor_grid | -scale_factor_uniform | -show
)
    [-geom <stringValue>] [-plot <stringValue>]
```

Argument	Description
<stringValue>	Name of the vector field.
-constant_heads	Shows whether the arrows are constant to the plot area regardless of the vector magnitude or proportional (normal) to the vector magnitude.
-fill_color	Color of a solid arrowhead.
-head_angle	Arrowhead angle.
-head_shape	Shape of the arrows.
-head_size	Size of the arrows.
-line_color	Color of the arrows.
-line_pattern	Line pattern of the arrows.
-line_width	Width of the arrows.
-scale	Scale for displaying the arrows, either uniform size or a grid display.
-scale_factor_grid	Grid factor for displaying the arrows.
-scale_factor_uniform	Uniform factor for displaying the arrows.
-show	Shows whether arrows are visible.
-geom <stringValue>	Specifies the geometry where the command will search for the vector.

Appendix A: Tcl Commands

get_vector_prop

Argument	Description
-plot <stringValue>	Name of the plot where the command will search for the geometry. If not specified, the command uses the selected plot.

Returns

The value of the queried property.

Example

```
get_vector_prop ElectricField-V -plot Plot_2D -geom 2D -scale  
#-> uniform
```

Appendix A: Tcl Commands

get_vertical_lines_prop

get_vertical_lines_prop

Returns the properties of vertical lines.

Note:

This command applies to xy plots only.

Syntax

```
get_vertical_lines_prop <stringValue>
  (-line_color | -line_style | -line_width | -show | -values )
  [-plot <stringValue>]
```

Argument	Description
<stringValue>	Name of a vertical line.
-line_color	Returns the color of the vertical line.
-line_style	Returns the style of the vertical line.
-line_width	Returns the width of the vertical line in pixels.
-show	Indicates whether the vertical line is visible.
-values	Returns a list of double values representing the x-values of the vertical line.
-plot <stringValue>	Name of the plot where the command searches for the vertical line. If not specified, the command uses the selected plot.

Returns

The value of the queried property.

Example

```
get_vertical_lines_prop VerticalLine_1 -values
#-> 0.001 0.0156
```

Appendix A: Tcl Commands

help

help

Displays information about Sentaurus Visual Tcl commands.

Without any arguments, the command returns a complete list of the available Tcl commands.

Syntax

```
help [<stringValue>]
```

Argument	Description
<stringValue>	Sentaurus Visual Tcl command.

Returns

String.

Example

```
help import_settings  
#-> import_settings <stringValue>
```

Appendix A: Tcl Commands

import_settings

import_settings

Imports Sentaurus Visual settings from a previously saved configuration file.

Syntax

```
import_settings <stringValue>
```

Argument	Description
<stringValue>	Name of the configuration file.

Returns

Integer.

Example

```
import_settings /path/to/settings.conf  
#-> 0
```

Appendix A: Tcl Commands

integrate_field

integrate_field

Integrates a field over a complete 2D or 3D plot, or in specific regions.

Note:

The command applies to 2D and 3D plots only.

Syntax

```
integrate_field
  [-bottom <doubleValue>] [-top <doubleValue>]
  [-dataset <stringValue> | -plot <stringValue>]
  [-field <stringValue>]
  [-geom <stringValue>]
  [-materials <stringList> | -regions <stringList>]
  [-points <pointList>] [-returnndomain]
  [-type x | y | z]
  [-window {<xmin> <ymin> [<zmin>] <xmax> <ymax> [<zmax>]}]
```

Argument	Description
-bottom <doubleValue>	Sets the lower bound of the polyhedron domain in the extrusion axis.
-top <doubleValue>	Sets the upper bound of the polyhedron domain in the extrusion axis.
-dataset <stringValue> -plot <stringValue>	Name of the plot or dataset. If not specified, uses the selected plot.
-field <stringValue>	Specifies the field to integrate. If not specified, uses the current active field.
-geom <stringValue>	Specifies the name of the dataset.
-materials <stringList> -regions <stringList>	List of regions or materials to integrate. If not specified, integrates all regions.
-points <pointList>	Sets the vertices for the polyline to be extruded along the chosen axis to create a polyhedron domain. Points must be defined as: { {x0 y0 z0} {x1 y1 z1} ... }
-returnndomain	Sets the function to return the integrated domain value (an integrated volume when used in three dimensions, or an integrated surface when used in two dimensions).
-type x y z	Selects the extrusion axis for the polyline defined by the points.

Appendix A: Tcl Commands

integrate_field

Argument	Description
<code>-window {<xmin> <ymin> [<zmin>] <xmax> <ymax> [<zmax>]}</code>	<p>Defines a specific window to integrate.</p> <p>Note: When the plot to be integrated is a 2D plot with x- and z-axes, the parameters change to {<xmin> <zmin> <xmax> <zmax>}. In the same way, when the plot has y- and z-axes, the parameters change to {<ymin> <zmin> <ymax> <zmax>}.</p>

Returns

Double.

Example

```
integrate_field -field Abs(ElectricField) -regions {RGate}
# Dataset: 3D
# Field: Abs(ElectricField)
# =====
# Regions of Dimension 3
# -----
# 1. RGate (PolySi)
#   Integral: 3.697957e-04 [V*um^2]
#   Domain: 9.211180e-04 [um^3]
# -----
# Total Integral: 3.697957e-04 [V*um^2]
# Total Domain: 9.211180e-04 [um^3]
# =====
#-> 0.000369796
```

Appendix A: Tcl Commands

link_plots

link_plots

Links plot properties of two or more plots.

Syntax

```
link_plots <stringList>
[-id <intValue>]
[-special {axes_prop axis_x axis_y axis_y2 blanking curve_prop cuts
deformation field_prop field_sel grid_prop legend_prop matreg move
plot_mode plot_prop streamlines}]
[-unlink]
```

Argument	Description
<stringList>	List of all the plots to be linked.
-id <intValue>	Integer identifier for the linked plots.
-special {<properties>}	Links only the specified properties:
axes_prop	Links axes properties (only for 2D plots).
axis_x, axis_y, axis_y2	Links properties of the x-, y-, and y2-axis, respectively (only for xy plots).
blanking	Links blanking operations (only for 2D and 3D plots).
curve_prop	Links curve properties (only for xy plots).
cuts	Links cuts such as cutlines and cutplanes (only for 2D and 3D plots).
deformation	Links deformation operations (only for 2D and 3D plots).
field_prop	Links the properties of fields such as the range, scale, and number of levels (only for 2D and 3D plots).
field_sel	Links the selection of fields (on and off for contour band and line fields) (only for 2D and 3D plots).
grid_prop	Links grid properties (only for xy plots).
legend_prop	Links legend properties.
matreg	Links material and region properties.

Appendix A: Tcl Commands

link_plots

Argument	Description
move	Links movements such as zoom, pan, and rotation (only for 3D plots).
plot_mode	Links the plot mode, such as select, zoom window, and probe.
plot_prop	Links plot properties, except the text of the plot title.
streamlines	Links operations on streamlines. Deactivated by default (only for 2D and 3D plots).
-unlink	Removes linking.

Returns

Integer.

Example

```
link_plots {Plot_1 Plot_2} -id 10
#-> 10
```

Appendix A: Tcl Commands

list_curves

list_curves

Returns a list of curve names.

Note:

The command applies to xy plots only.

Syntax

```
list_curves [<stringValue>] [-plot <stringValue>]  
[-selected | -not_selected] [-show | -hide]
```

Argument	Description
<stringValue>	Specifies the search pattern to use.
-plot <stringValue>	Searches on a specific plot. If not specified, the command searches on the selected plot.
-selected -not_selected	Returns a list of curves that are selected or are not selected.
-show -hide	Shows or hides the results.

Returns

List.

Example

```
list_curves _1  
#-> Curve_1 Curve_12
```

Appendix A: Tcl Commands

list_custom_buttons

list_custom_buttons

Returns a list of custom buttons.

Syntax

```
list_custom_buttons [<stringValue>]
```

Argument	Description
<stringValue>	Specifies the search pattern to use. If not specified, the command lists all custom buttons.

Returns

List of all custom buttons and separators that match the pattern. If no search pattern is specified, the command returns all custom buttons.

Example

```
list_custom_buttons Buttons  
#-> Buttons1 Custom_Button2 MyButton
```

Appendix A: Tcl Commands

list_cutlines

list_cutlines

Returns a list of cutlines. If no arguments are specified, the command returns all cutlines.

Note:

The command applies to 2D plots only.

Syntax

```
list_cutlines [<stringValue>] [-plot <stringValue>]  
[-type x | y | z | free]
```

Argument	Description
<stringValue>	Specifies the search pattern to use.
-plot <stringValue>	Searches on a specific plot. If not specified, the command searches on the selected plot.
-type x y z free	Specifies the type of cutline to search for.

Returns

List.

Example

```
list_cutlines -type y  
#-> C1 C2
```

Appendix A: Tcl Commands

list_cutplanes

list_cutplanes

Returns a list of cutplanes. If no arguments are specified, the command returns all cutplanes.

Note:

The command applies to 3D plots only.

Syntax

```
list_cutplanes [<stringValue>] [-plot <stringValue>]  
[-type x | y | z | free]
```

Argument	Description
<stringValue>	Specifies the search pattern to use.
-plot <stringValue>	Searches on a specific plot. If not specified, the command searches on the selected plot.
-type x y z free	Specifies the type of cutplane to search for.

Returns

List.

Example

```
list_cutplanes -type y  
#-> C3
```

list_datasets

Returns a list of dataset names according to the given pattern. If no pattern is specified, then all datasets are returned. You can filter the returned datasets by dimension, a given plot, or the selected plot. The dimension filter can be combined with the given plot or selected plot filter.

Syntax

```
list_datasets [<stringValue>] [-dim <intValue>]  
[-plot <stringValue> | -selected]
```

Argument	Description
<stringValue>	Specifies the search pattern to use.
-dim <intValue>	Dimension of datasets, which can be 1, 2, or 3.
-plot <stringValue> -selected	Specifies whether to filter by plot name or by the selected plot. If you specify one of these arguments, then the list of returned datasets is restricted to those contained in the named plot, for the -plot argument, or those contained in the selected plot, for the -selected argument.

Returns

List.

Example

```
list_datasets -dim 3  
#-> 3D_3
```

Appendix A: Tcl Commands

list_ellipses

list_ellipses

Returns a list of names of ellipses.

Note:

This command applies to xy plots only.

Syntax

```
list_ellipses [<stringValue>] [-plot <stringValue>]
```

Argument	Description
<stringValue>	Specifies the search pattern to use. If no search pattern is specified, the names of all ellipses are returned.
-plot <stringValue>	Name of a plot to search. If not specified, the command uses the selected plot.

Returns

List of names of ellipses matching the search pattern.

Example

```
list_ellipses -plot Plot_1  
#-> Ellipse_1 Ellipse_2
```

Appendix A: Tcl Commands

list_fields

list_fields

Returns a list of field names.

Note:

This command applies to 2D and 3D plots only.

Syntax

```
list_fields [<stringValue>]
    [-dataset <stringValue> | -geom <stringValue> | -plot <stringValue>]
    [-show | -hide] [-show_bands | -hide_bands]
```

Argument	Description
<stringValue>	Specifies the search pattern to use.
-dataset <stringValue> -geom <stringValue> -plot <stringValue>	Searches a specific plot, dataset, or geometry.
-show -hide	Shows or hides contour bands.
-show_bands -hide_bands	Searches fields with contour bands shown or hidden.

Returns

List.

Example

```
list_fields ElectricField -plot Plot_3D
#-> Abs(ElectricField) ElectricField-X ElectricField-Y ElectricField-Z
```

Appendix A: Tcl Commands

list_files

list_files

Returns a list of opened files according to the given pattern. If no pattern is specified, all files are returned. You can filter the returned files by only those contained in the selected plot.

Syntax

```
list_files [<stringValue>] [-fullpath] [-selected]
```

Argument	Description
<stringValue>	Specifies the search pattern to use.
-fullpath	Specifies the full path to where the directory resides that contains the opened files. If not specified, uses the current work directory.
-selected	Specifies to retrieve only the files contained in the selected plot.

Returns

List.

Example

```
list_files  
#-> 2D_file.tdr 3D_file.tdr
```

Appendix A: Tcl Commands

list_lines

list_lines

Returns a list of line names.

Note:

This command applies to xy and 2D plots only.

Syntax

```
list_lines [<stringValue>] [-plot <stringValue>]
```

Argument	Description
<stringValue>	Specifies the search pattern to use. If no search pattern is specified, the names of all lines are returned.
-plot <stringValue>	Name of the plot. If not specified, the command uses the selected plot.

Returns

List of names of lines matching the search pattern.

Example

```
list_lines -plot Plot_1  
#-> Line_1 Line_2
```

Appendix A: Tcl Commands

list_materials

list_materials

Returns a list of material names.

Note:

The command applies to 2D and 3D plots only.

Syntax

```
list_materials [<stringValue>]
    [-dataset <stringValue> | -geom <stringValue> | -plot <stringValue>]
    [-show_all | -hide_all] [-show_border | -hide_border]
    [-show_bulk | -hide_bulk] [-show_field | -hide_field]
    [-show_mesh | -hide_mesh]
```

Argument	Description
<stringValue>	Specifies the search pattern to use.
-dataset <stringValue> -geom <stringValue> -plot <stringValue>	Searches a specific plot, dataset, or geometry.
-show_all -hide_all	Shows or hides materials completely.
-show_border -hide_border	Shows or hides borders of materials.
-show_bulk -hide_bulk	Shows or hides bulk of materials.
-show_field -hide_field	Shows or hides fields of materials.
-show_mesh -hide_mesh	Shows or hides mesh of materials.

Returns

List.

Example

```
list_materials -plot Plot_3D
#-> Contact DepletionRegion JunctionLine nitride Oxide PolySi Silicon
```

Appendix A: Tcl Commands

list_movie_frames

list_movie_frames

Returns the list of frames in the frame buffer.

Syntax

```
list_movie_frames
```

Returns

List.

Example

```
list_movie_frames  
# Frame0001 Frame0002 Frame0003 Frame0004
```

list_plots

Returns a list of plot names according to the given pattern. If no pattern is specified, all plots are returned.

Syntax

```
list_plots [<stringValue>] [-dim <intValue>] [-link <intValue>]  
[-selected] [-show | -hide]
```

Argument	Description
<stringValue>	Specifies the search pattern to use.
-dim <intValue>	Dimension of plots, which can be 1, 2, or 3.
-link <intValue>	Returns only linked plots with the ID link equal to <intValue>.
-selected	Returns the selected plot.
-show -hide	Specifies whether only visible plots (-show) or only hidden plots (-hide) will be listed.

Returns

List.

Example

```
list_plots -dim 3  
#-> 3D
```

Appendix A: Tcl Commands

list_rectangles

list_rectangles

Returns a list of rectangle names.

Note:

This command applies to xy and 2D plots only.

Syntax

```
list_rectangles [<stringValue>] [-plot <stringValue>]
```

Argument	Description
<stringValue>	Specifies the search pattern to use. If no search pattern is specified, the names of all rectangles are returned.
-plot <stringValue>	Name of the plot. If not specified, the command uses the selected plot.

Returns

List of names of rectangles matching the search pattern.

Example

```
list_rectangles -plot Plot_1  
#-> Rectangle_1 Rectangle_2 Rectangle_3
```

Appendix A: Tcl Commands

list_regions

list_regions

Returns a list of region names.

Note:

The command applies to 2D and 3D plots only.

Syntax

```
list_regions [<stringValue>]
    [-dataset <stringValue> | -geom <stringValue> | -plot <stringValue>]
    [-material <stringValue>] [-parts]
    [-show_all | -hide_all] [-show_border | -hide_border]
    [-show_bulk | -hide_bulk] [-show_field | -hide_field]
    [-show_mesh | -hide_mesh]
```

Argument	Description
<stringValue>	Specifies the search pattern to use.
-dataset <stringValue> -geom <stringValue> -plot <stringValue>	Returns the regions of the specified plot, dataset, or geometry. If not specified, the command returns the regions of the selected plot.
-material <stringValue>	Returns the regions present in the specified material.
-parts	Returns a list of regions that have parts.
-show_all -hide_all	Filters by whether regions are completely shown or hidden.
-show_border -hide_border	Filters by whether materials have their border shown or hidden.
-show_bulk -hide_bulk	Filters by whether materials have their bulk shown or hidden.
-show_field -hide_field	Filters by whether materials have their field shown or hidden.
-show_mesh -hide_mesh	Filters by whether materials have their mesh shown or hidden.

Returns

List.

Example

```
list_regions -dataset 3D
#-> bulk gate drain DepletionRegion JunctionLine source
```

Appendix A: Tcl Commands

list_streamlines

list_streamlines

Returns a list of streamlines created on the plot.

Note:

The command applies to 2D and 3D plots only.

Syntax

```
list_streamlines [<stringValue>] [-plot <stringValue>]
```

Argument	Description
<stringValue>	Specifies the search pattern to use.
-plot <stringValue>	Returns the streamlines of the specified plot. If not specified, the command returns the streamlines of the selected plot.

Returns

List.

Example

```
list_streamlines  
#-> Streamline_1 Streamline_2 Streamline_3
```

Appendix A: Tcl Commands

list_tdr_states

list_tdr_states

Returns a list of state names for the geometry visualized in a plot.

Syntax

```
list_tdr_states [<stringValue>] [-plot <stringValue>]
```

Argument	Description
<stringValue>	Specifies the search pattern that the state name must match. If not specified, all state names are returned.
-plot <stringValue>	Specifies the name of the plot where the geometry of interest is visualized. If not specified, the command uses the selected plot.

Returns

List.

Example

```
list_tdr_states state_1* -plot Plot_2D  
#-> state_1 state_10 state_100
```

Appendix A: Tcl Commands

list_textboxes

list_textboxes

Returns a list of text box names.

Note:

This command applies to xy and 2D plots only.

Syntax

```
list_textboxes [<stringValue>] [-plot <stringValue>]
```

Argument	Description
<stringValue>	Specifies the search pattern to use. If no search pattern is specified, the names of all text boxes are returned.
-plot <stringValue>	Name of the plot. If not specified, the command uses the selected plot.

Returns

List of names of text boxes matching the search pattern.

Example

```
list_textboxes -plot Plot_1  
#-> TextBox_1 TextBox_2 TextBox_3
```

Appendix A: Tcl Commands

list_variables

list_variables

Returns a list of variable names according to the given pattern. If no pattern is specified, all variables are returned.

Note:

The command applies to xy plots only.

Syntax

```
list_variables -dataset <stringValue> [<stringValue>]
```

Argument	Description
-dataset <stringValue>	Specifies the dataset to use.
<stringValue>	Specifies the search pattern to use.

Returns

List.

Example

```
list_variables -dataset testDataset  
#-> drainCurrent gateToSourceVoltage time
```

Appendix A: Tcl Commands

list_vertical_lines

list_vertical_lines

Returns a list of the names of vertical lines.

Note:

This command applies to xy plots only.

Syntax

```
list_vertical_lines [<stringValue>] [-plot <stringValue>]
```

Argument	Description
<stringValue>	Specifies the search pattern to use. If no search pattern is specified, the names of all vertical lines are returned.
-plot <stringValue>	Name of the plot. If not specified, the command uses the selected plot.

Returns

List of names of vertical lines matching the search pattern.

Example

```
list_vertical_lines -plot Plot_1  
#-> DE(C1(n_10)) DE(C2(n_10))
```

Appendix A: Tcl Commands

load_file

load_file

Loads the specified file, and returns a string with the dataset name associated with the file.

Syntax

```
load_file <stringValue>
    [-alldata] [-fod] [-geoms <intList>] [-name <stringValue>] [-parts]
```

Argument	Description
<stringValue>	Name of the file to load.
-alldata	Loads all data from the file regardless of the user preference settings.
-fod	Loads field data on demand.
-geoms <intList>	Specifies the geometry indices to load.
-name <stringValue>	Specifies a custom dataset name. See Object Names: -name Argument on page 213 .
-parts	This option specifies that, if regions have parts, the parts are displayed and controlled independently.

Returns

String.

Example

```
load_file /pathTo/Structure.tdr -geoms {0 2}
#-> Structure_geometry_0
```

Appendix A: Tcl Commands

load_file_datasets

load_file_datasets

Loads datasets from the specified file.

Syntax

```
load_file_datasets <stringValue> [-alldata] [-geoms <intList>] [-parts]
```

Argument	Description
<stringValue>	Name of the file.
-alldata	Loads all data from the file regardless of the user preference settings.
-geoms <intList>	Specifies the geometry indices to load.
-parts	This option specifies that, if regions have parts, the parts are displayed and controlled independently.

Returns

List.

Example

```
load_file_datasets /pathTo/IdVg.tdr
#-> IdVg
```

Appendix A: Tcl Commands

load_library

load_library

Loads a Sentaurus Visual library. It can load either all libraries located at the default paths, or only libraries located at the given path, or only the library given by the specified path and the library name.

Syntax

```
load_library  
  -all |  
  -path <stringValue> [<stringValue>]
```

Argument	Description
-all	Loads all libraries located at the default path.
-path <stringValue> [<stringValue>]	Path to where libraries for loading are located. Optionally, you can specify the name of a particular library located at the specified path.

Returns

Integer.

Example

```
load_library -path ~/mySVLibs myLibrary1  
#-> 0
```

Appendix A: Tcl Commands

load_script_file

load_script_file

Loads a Tcl script or Inspect script.

Syntax

```
load_script_file <stringValue> [-inspect]
```

Argument	Description
<stringValue>	Name of the Tcl script to load.
-inspect	Forces Sentaurus Visual to execute the script as an Inspect script.

Returns

Integer.

Example

```
load_script_file testScript.tcl  
#-> 0
```

Appendix A: Tcl Commands

move_plot

move_plot

Moves the selected plot.

Syntax

```
move_plot -position <doubleList>
           [-absolute] [-plot <stringValue>]
```

Argument	Description
-position <doubleList>	Sets the new position of the plot. Arguments of type <i>Double</i> are expected.
-absolute	Moves plot to an absolute position if specified.
-plot <stringValue>	Name of the plot to be moved. If not specified, the current active plot is used.

Returns

Integer.

Example

```
move_plot -position {1.5 0.5} -absolute
#-> 0
```

Appendix A: Tcl Commands

overlay_plots

overlay_plots

Overlays two or more plots. Creates a new plot with the specified name.

Note:

The command applies to 2D and 3D plots only.

Syntax

```
overlay_plots <stringList>
    [-datasets <stringList>] [-name <stringValue>]
```

Argument	Description
<stringList>	List of plots to be overlaid onto the first plot. If only one plot is given, the command overlays the plot onto the list of datasets.
-datasets <stringList>	Overlays this list of datasets to the list of plots specified.
-name <stringValue>	Name of the new plot to be created. If not specified, creates a new plot with a generic name. See Object Names: -name Argument on page 213 .

Returns

String.

Example

```
overlay_plots {Plot_3D Plot_3D_2} -name Plot_Overlay
#-> Plot_Overlay
```

Appendix A: Tcl Commands

probe_curve

probe_curve

Probes a curve using the interpolation that matches the axis (linear if the axis is in normal mode or log if the axis is in log mode).

Note:

The command applies to xy plots only. For 2D and 3D plots, use the command `probe_field` (see [probe_field on page 325](#)).

Syntax

```
probe_curve <stringValue>
    -valueX <doubleValue> | -valueY <doubleValue> | -valueY2 <doubleValue>
    [-extrapolate | -snap] [-plot <stringValue>] [-position | -scalar]
```

Argument	Description
<stringValue>	Identifier associated with the curve to be probed.
-valueX <doubleValue> -valueY <doubleValue> -valueY2 <doubleValue>	Specifies the point to probe on the curve.
-extrapolate -snap	Specifies either that the probe can extrapolate values or that the probe returns the curve closest point.
-plot <stringValue>	Name of the plot with the curve to be probed. If not specified, the command uses the selected plot.
-position -scalar	Returns either the position of the curve or the scalar value of the curve.

Returns

Double.

Example

```
probe_curve idvg_1_des -valueX 0.85
#-> 0.5433e-6
```

Appendix A: Tcl Commands

probe_field

probe_field

Probes a point on a plot, and returns the values of the defined field on that point.

Note:

This command applies to 2D and 3D plots only. For xy plots, use the command probe_curve.

Syntax

```
probe_field
    -coord {<x> <y> [<z>]} |
    -point <intValue> -region <stringValue>
    [-field <stringValue>] [-geom <stringValue> | -plot <stringValue>]
    [-position] [-snap]
```

Argument	Description
-coord {<x> <y> [<z>]}	Specifies the point to be probed. In 2D plots, the <i>z</i> value must be 0 or must be left undefined.
-point <intValue>	Specifies the vertex ID relative to the region set to be probed. Use with the <i>-region</i> option.
-region <stringValue>	Specifies the region where the field will be probed. Use with the <i>-point</i> option.
-field <stringValue>	Name of the field to probe in the plot. If not specified, probes the active field.
-geom <stringValue> -plot <stringValue>	Name of the plot or geometry to be probed. If not specified, the command probes the selected plot.
-position	Returns a vector with the coordinates of the probe. If <i>-snap</i> is specified, returns the coordinates of the closest node. If <i>-snap</i> is not specified, returns <i>-coord</i> .
-snap	Probes the field at the closest node if specified.

Returns

Double.

Example

```
probe_field -field DopingConcentration -coord {0.2 0.3 -0.2}
#-> -2e18
```

Appendix A: Tcl Commands

reload_datasets

reload_datasets

Reloads all the specified datasets.

Syntax

```
reload_datasets <stringList> [-alldata]
```

Argument	Description
<stringList>	List of datasets to reload.
-alldata	Reloads all data from the files regardless of the user preference settings.

Returns

Integer.

Example

```
reload_datasets {3D 3D_2}  
#-> 0
```

reload_files

Reloads the specified files.

Syntax

```
reload_files <stringList> [-alldata]
```

Argument	Description
<stringList>	List of files to reload.
-alldata	Reloads all data from the files regardless of the user preference settings.

Returns

Integer.

Example

```
reload_files {2D.tdr 3D.tdr}  
#-> 0
```

Appendix A: Tcl Commands

remove_curves

remove_curves

Removes curves from an xy plot.

Syntax

```
remove_curves <stringList> [-plot <stringValue>]
```

Argument	Description
<stringList>	List of curve names.
-plot <stringValue>	Name of the plot from where curves will be removed. If not specified, the command uses the active plot.

Returns

Integer.

Example

```
remove_curves {IdVg_1 IdVg_2}  
#-> 0
```

remove_custom_buttons

Removes custom buttons.

Syntax

```
remove_custom_buttons <stringList>
```

Argument	Description
<stringList>	List of names of custom buttons.

Returns

List of all custom buttons and separators removed.

Example

```
remove_custom_buttons {Buttons1 MyButton}  
#-> Buttons1 MyButton
```

Appendix A: Tcl Commands

remove_cutlines

remove_cutlines

Removes the specified cutlines.

Syntax

```
remove_cutlines <stringList> [-plot <stringValue>]
```

Argument	Description
<stringList>	List of cutline names.
-plot <stringValue>	Name of plot from where the cutlines will be removed. If not specified, the command uses the active plot.

Returns

List.

Example

```
remove_cutlines {C1 C2}
#-> C1 C2
```

remove_cutplanes

Removes the specified cutplanes.

Syntax

```
remove_cutplanes <stringList> [-plot <stringValue>]
```

Argument	Description
<stringList>	List of cutplane names.
-plot <stringValue>	Name of plot from where the cutplanes will be removed.

Returns

List.

Example

```
remove_cutplanes {C1 C2}
#-> C1 C2
```

Appendix A: Tcl Commands

remove_datasets

remove_datasets

Removes the specified datasets.

Syntax

```
remove_datasets <stringList>
```

Argument	Description
<stringList>	List of dataset names.

Returns

Integer.

Example

```
remove_datasets {dataSet1 dataSet2 dataSet3}  
#-> 0
```

remove_ellipses

Removes ellipses from a plot.

Note:

This command applies to xy plots only.

Syntax

```
remove_ellipses <stringList> [-plot <stringValue>]
```

Argument	Description
<stringList>	List of ellipse names.
-plot <stringValue>	Name of the plot. If not specified, the command uses the selected plot.

Returns

List of deleted ellipses.

Appendix A: Tcl Commands

remove_lines

Example

```
remove_ellipses {Ellipse_1 Ellipse_2} -plot Plot_1  
#-> Ellipse_1 Ellipse_2
```

remove_lines

Removes lines from a plot.

Note:

This command applies to xy and 2D plots only.

Syntax

```
remove_lines <stringList> [-plot <stringValue>]
```

Argument	Description
<stringList>	List of line names.
-plot <stringValue>	Name of the plot. If not specified, the command uses the selected plot.

Returns

List of deleted lines.

Example

```
remove_lines Line_1 -plot Plot_n60_des  
#-> Line_1
```

Appendix A: Tcl Commands

remove_plots

remove_plots

Removes the specified plots.

Syntax

```
remove_plots <stringList>
```

Argument	Description
<stringList>	List of plot names.

Returns

Integer.

Example

```
remove_plots {plotXY plot2D plot3D}  
#-> 0
```

remove_rectangles

Removes rectangles from a plot.

Note:

This command applies to xy and 2D plots only.

Syntax

```
remove_rectangles <stringList> [-plot <stringValue>]
```

Argument	Description
<stringList>	List of rectangle names.
-plot <stringValue>	Name of the plot. If not specified, the command uses the selected plot.

Returns

List of deleted rectangles.

Appendix A: Tcl Commands

remove_rulers

Example

```
remove_rectangles {Rectangle_1 Rectangle_2} -plot Plot_n60_des
#-> Rectangle_1 Rectangle_2
```

remove_rulers

Removes the specified rulers from a plot.

Syntax

```
remove_rulers <intList> [-plot <stringValue>]
```

Argument	Description
<intList>	List of ruler IDs.
-plot <stringValue>	Name of the plot from where the rulers will be removed. If not specified, the command uses the selected plot.

Returns

Integer.

Example

```
remove_rulers {1 2 3}
#-> 0
```

Appendix A: Tcl Commands

remove_streamlines

remove_streamlines

Removes the specified streamlines.

Syntax

```
remove_streamlines <stringList> [-plot <stringValue>]
```

Argument	Description
<stringList>	List of streamline names.
-plot <stringValue>	Name of the plot from where the streamlines will be removed. If not specified, the command uses the selected plot.

Returns

List.

Example

```
remove_streamlines {Streamline_1 Streamline_2}
#-> Streamline_1 Streamline_2
```

Appendix A: Tcl Commands

remove_textboxes

remove_textboxes

Removes text boxes from a plot.

Note:

This command applies to xy and 2D plots only.

Syntax

```
remove_textboxes <stringList> [-plot <stringValue>]
```

Argument	Description
<stringList>	List of names of text boxes.
-plot <stringValue>	Name of the plot. If not specified, the command uses the selected plot.

Returns

List of deleted text boxes.

Example

```
remove_textboxes {TextBox_1 TextBox_2 TextBox_3} -plot Plot_1  
#-> TextBox_1 TextBox_2 TextBox_3
```

Appendix A: Tcl Commands

render_mode

render_mode

Updates the rendering status when plots are loaded. If rendering is switched off, you must switch it on when plots are finished loading; otherwise, no plots are displayed.

Note:

This command applies to 2D and 3D plots only.

Syntax

```
render_mode [-on | -off]
```

Argument	Description
-on -off	Switches on or off rendering when plots are loaded. If no option is specified, the command displays the current status of rendering

Returns

Current status of rendering mode.

Example

```
render_mode -on  
#-> on
```

reset_settings

Reset Sentaurus Visual settings to their default values.

Syntax

```
reset_settings
```

Returns

None.

Example

```
reset_settings
```

Appendix A: Tcl Commands

rotate_plot

rotate_plot

Rotates a 3D plot by specifying either axes, or angles, or a direction, or a plane. Different axes can be rotated simultaneously. The axis and angles used are explained in [Figure 67 on page 121](#).

Note:

This command applies to 3D plots only and is supported in batch mode in Sentaurus Visual.

Syntax

```
rotate_plot
    -alpha <doubleValue> -psi <doubleValue> -theta <doubleValue> |
    -angle <doubleValue> -direction (up | down | left | right) |
    -plane (xy | yz | xz) |
    -x <doubleValue> -y <doubleValue> -z <doubleValue> |
    [-absolute] [-plot <stringValue>]
```

Argument	Description
-alpha <doubleValue>	Rotates the plot using alpha spherical coordinate (in degrees).
-psi <doubleValue>	Rotates the plot using psi spherical coordinate (in degrees).
-theta <doubleValue>	Rotates the plot using theta spherical coordinate (in degrees).
-angle <doubleValue>	Rotates the plot in the specified direction by the angle specified by -angle.
-direction up down left right	Rotates the plot by the specified angle (in degrees) in the direction defined by -direction.
-plane xy yz xz	Rotates the plot in the specified plane.
-x <doubleValue>	Rotates the plot around the x-axis (in degrees).
-y <doubleValue>	Rotates the plot around the y-axis (in degrees).
-z <doubleValue>	Rotates the plot around the z-axis (in degrees).
-absolute	Rotates the plot to an absolute position if specified.
-plot <stringValue>	Name of the plot. If not specified, the command uses the selected plot.

Returns

Integer.

Appendix A: Tcl Commands

save_plot_to_script

Example

```
rotate_plot -x 10.5 -y 20  
#-> 0  
rotate_plot -plane xz  
#-> 0  
rotate_plot -direction up -angle 90  
#-> 0
```

save_plot_to_script

Exports the plot properties and curve data of the current plot to a Tcl file.

Note:

This command applies to xy plots only.

Syntax

```
save_plot_to_script <stringValue>  
[-overwrite] [-plot <stringValue>]
```

Argument	Description
<stringValue>	Specifies the path (either absolute or relative) where the resulting Tcl file will be located.
-overwrite	Specifies whether the target Tcl file should be overwritten if it already exists.
-plot <stringValue>	Name of the plot to be exported. If not specified, the command uses the selected plot.

Returns

Integer.

Example

```
save_plot_to_script testFile.tcl -plot Plot_1 -overwrite  
#-> 0
```

Appendix A: Tcl Commands

select_plots

select_plots

Selects the plots.

Syntax

```
select_plots <stringList>
```

Argument	Description
<stringList>	List of plot names to be selected. Passing an empty list (<code>select_plots {}</code>) deselects all plots.

Returns

List.

Example

```
select_plots {plot2D anotherPlot2D}
#-> plot2D anotherPlot2D
```

Appendix A: Tcl Commands

set_axis_prop

set_axis_prop

Sets axis properties.

If `-axis` is not specified, the properties are set for all axes.

Note:

The command applies to xy and 2D plots only.

Syntax

```
set_axis_prop
  [-anchor <doubleValue>]
  [-auto_padding | -manual_padding]
  [-auto_precision | -manual_precision]
  [-auto_spacing | -manual_spacing]
  [-axis x | y | z | y2]
  [-inverted | -not_inverted]
  [-label_angle <doubleValue>]
  [-label_font_att normal | bold | italic | underline | strikeout]
  [-label_font_color <#rrggbb>]
  [-label_font_factor <doubleValue>]
  [-label_font_family arial | courier | times]
  [-label_font_size <intValue>]
  [-label_format preferred | scientific | engineering | fixed]
  [-label_padding <intValue>] [-label_precision <intValue>]
  [-major_ticks_length <intValue>]
  [-major_ticks_width <intValue>]
  [-max <doubleValue>] [-max_auto | -max_fixed]
  [-min <doubleValue>] [-min_auto | -min_fixed]
  [-minor_ticks_length <intValue>]
  [-minor_ticks_position center | in | out]
  [-minor_ticks_width <intValue>]
  [-nof_minor_ticks <intValue>]
  [-padding <intValue>]
  [-plot <stringValue>]
  [-range {<x1> <x2>}] [-reset]
  [-show | -hide] [-show_minor_ticks | -hide_minor_ticks]
  [-show_label | -hide_label] [-show_ticks | -hide_ticks]
  [-show_title | -hide_title] [-spacing <doubleValue>]
  [-ticks_position out | in | center] [-title <stringValue>]
  [-title_font_att normal | bold | italic | underline | strikeout]
  [-title_font_color <#rrggbb>]
  [-title_font_factor <doubleValue>]
  [-title_font_family arial | courier | times]
  [-title_font_size <intValue>]
  [-type linear | log]
```

Appendix A: Tcl Commands

set_axis_prop

Argument	Description
-anchor <doubleValue>	Sets the anchor of ticks.
-auto_padding -manual_padding	Specifies either automatic padding or manual padding of the axis (applies to xy plots only).
-auto_precision -manual_precision	Sets the automatic or manual precision of the axis.
-auto_spacing -manual_spacing	Sets the spacing mode of ticks.
-axis x y z y2	Axis to apply the settings. If not specified, the command applies the attributes to all axes.
-inverted -not_inverted	Inverts the axis values.
-label_angle <doubleValue>	Sets the rotation angle applied to the labels of major ticks (applies to xy plots only). Only values between –90 and 90, and including –90 and 90, are valid.
-label_font_att normal bold italic underline strikeout	Sets the font attributes of the axis.
-label_font_color <#rrggbb>	Sets the axis color.
-label_font_factor <doubleValue>	Sets the size factor of the axis (applies to 2D and 3D plots only).
-label_font_family arial courier times	Sets the axis font.
-label_font_size <intValue>	Sets the font size of the axis (applies to xy plots only).
-label_format preferred scientific engineering fixed	Sets the axis numeric format.
-label_padding <intValue>	Sets the padding of the axis values.
-label_precision <intValue>	Sets the numeric precision of the axis.
-major_ticks_length <intValue>	Sets the length of major ticks.

Appendix A: Tcl Commands

set_axis_prop

Argument	Description
-major_ticks_width <intValue>	Sets the width of major ticks (applies to xy plots only).
-max <doubleValue>	Sets the maximum value of the axis.
-max_auto -max_fixed	Sets the maximum of the axis automatically, or fixes the maximum of the axis to a user-defined value (applies to xy plots only). If -max_fixed is specified, any change to the data will not update the range.
-min <doubleValue>	Sets the minimum value of the axis.
-min_auto -min_fixed	Sets the minimum of the axis automatically, or fixes the minimum of the axis to a user-defined value (applies to xy plots only). If -min_fixed is specified, any change to the data will not update the range.
-minor_ticks_length <intValue>	Sets the length of minor ticks.
-minor_ticks_position center in out	Sets the position of minor ticks (applies to 2D plots only).
-minor_ticks_width <intValue>	Sets the width of minor ticks (applies to xy plots only).
-nof_minor_ticks <intValue>	Sets the number of minor ticks.
-padding <intValue>	Sets the padding of the axis in pixels (applies to xy plots only). If -auto_padding is specified, -padding has no effect.
-plot <stringValue>	Name of the plot. If not specified, the command applies the attributes to the selected plot.
-range {<x1> <x2>}	Sets the axis range.
-reset	Resets axis parameters to default values (applies to xy plots only).
-show -hide	Shows or hides the axis.
-show_minor_ticks -hide_minor_ticks	Shows or hides the minor ticks (applies to xy plots only).

Appendix A: Tcl Commands

set_axis_prop

Argument	Description
-show_label -hide_label	Shows or hides the label.
-show_ticks -hide_ticks	Shows or hides the major ticks.
-show_title -hide_title	Shows or hides the axis label.
-spacing <doubleValue>	Sets the spacing between ticks.
-ticks_position out in center	Sets the position of the ticks.
-title <stringValue>	Sets the axis label.
-title_font_att normal bold italic underline strikeout	Sets font attributes of the axis label.
-title_font_color <#rrggbb>	Sets the axis label color.
-title_font_factor <doubleValue>	Sets the font size factor (2D and 3D plots only).
-title_font_family arial courier times	Sets the axis label font.
-title_font_size <intValue>	Sets the font size of the axis label font size (xy plots only).
-type linear log	Sets the axis scale (applies to xy plots only).

Returns

String.

Example

```
set_axis_prop -axis y1 -title "Drain Current"  
#-> 0
```

Appendix A: Tcl Commands

set_band_diagram

set_band_diagram

Creates a band diagram. For details, see [Plotting Band Diagrams on page 91](#).

Syntax

```
set_band_diagram [<stringList>]
```

Argument	Description
<stringList>	List of plots created with the cutline function.

Returns

Integer.

Example

```
set_band_diagram Plot_1  
#-> 0
```

set_best_look

Automatically configures various plot parameters. For details, see [Best Look Option on page 89](#).

Syntax

```
set_best_look [<stringList>]
```

Argument	Description
<stringList>	List of plots to apply best look settings.

Returns

Integer.

Example

```
set_best_look {Plot_1 Plot_2}  
#-> 0
```

Appendix A: Tcl Commands

set_camera_prop

set_camera_prop

Sets camera properties.

Note:

The command applies to 3D plots only.

Syntax

```
set_camera_prop
  [-dolly_zoom_on | -dolly_zoom_off]
  [-focal_point {<x> <y> <z>}]
  [-parallel | -perspective]
  [-plot <stringValue>]
  [-position {<x> <y> <z>}] [-reset]
  [-rot_color <#rrggbb>] [-rot_size <intValue>]
  [-rot_width <intValue>]
  [-rotation_point {<x> <y> <z>}]
  [-setup <listdoubleList>]
  [-show_rotation_point | -hide_rotation_point]
  [-view_angle <doubleValue>] [-view_up <doubleList>]
```

Argument	Description
-dolly_zoom_on -dolly_zoom_off	Sets to use dolly zoom instead of modifying the view angle when zooming.
-focal_point {<x> <y> <z>}	Sets the focal point of the camera.
-parallel -perspective	Sets the projection mode in which to display the plot; either parallel or perspective projection.
-plot <plotName>	Name of the plot. If not specified, the command uses the selected plot.
-position {<x> <y> <z>}	Sets the position of the camera.
-reset	Resets camera settings to their default values.
-rot_color <#rrggbb>	Sets the color of the rotation point.
-rot_size <intValue>	Sets the size of the rotation point.
-rot_width <intValue>	Sets the width of the rotation point.
-rotation_point {<x> <y> <z>}	Sets the rotation point of the structure.
-setup <listdoubleList>	Specifies the new setup of the camera.

Appendix A: Tcl Commands

set_contour_labels_prop

Argument	Description
-show_rotation_point -hide_rotation_point	Shows or hides the rotation point on the plot.
-view_angle <doubleValue>	Sets the view angle of the camera in degrees.
-view_up <doubleList>	Sets the view-up vector of the camera.

Returns

Integer.

Example

```
set_camera_prop -rotation_point {0.2 0.35 -0.25}  
#-> 0
```

set_contour_labels_prop

Sets properties of contour labels.

Syntax

```
set_contour_labels_prop <stringList>  
[-anchor left | center | right]  
[-color_bg <#rrggbb>]  
[-font_att normal | bold | italic | underline | strikeout]  
[-font_color <#rrggbb>]  
[-font_factor <doubleValue>]  
[-font_family arial | courier | times]  
[-format engineering | fixed | preferred | scientific]  
[-geom <stringValue>]  
[-level_skip <intValue>]  
[-plot <stringValue>]  
[-precision <intValue>]  
[-reset]  
[-show_bg | -hide_bg]  
[-show_index | -hide_index]  
[-spacing <intValue>]
```

Argument	Description
<stringList>	List of fields on which to apply the specified properties.
-anchor left center right	Sets the contour labels anchor.

Appendix A: Tcl Commands

set_contour_labels_prop

Argument	Description
-color_bg <#rrggbb>	Sets the background color of the contour labels.
-font_att normal bold italic underline strikeout	Sets the font attribute of the contour labels.
-font_color <#rrggbb>	Sets the color of the contour labels.
-font_factor <doubleValue>	Sets font factor of the contour labels.
-font_family arial courier times	Sets the font type of the contour labels.
-format engineering fixed preferred scientific	Sets the format of the contour labels.
-geom <stringValue>	Name of the dataset (or geometry). If not specified, then the command uses the main one from the active plot.
-level_skip <intValue>	Sets the level skip of the contour labels. The value 0 means every level is labeled.
-plot <stringValue>	Name of the plot. If not specified, then the command uses the selected plot.
-precision <intValue>	Sets the precision of the contour labels.
-reset	Resets the contour labels properties.
-show_bg -hide_bg	Shows or hides the background of the contour labels.
-show_index -hide_index	Shows or hides the contour labels index.
-spacing <intValue>	Sets the spacing of the contour labels.

Returns

Integer.

Example

```
set_contour_labels_prop {BandGap DopingConcentration} -precision 3  
#-> 0
```

Appendix A: Tcl Commands

set_curve_prop

set_curve_prop

Sets curve properties.

Note:

The command applies to xy plots only.

Syntax

```
set_curve_prop <stringList>
    [-axis left | right] [-color <#rrggbb>] [-deriv <intValue> | -integ]
    [-function <stringValue>] [-label <stringValue>]
    [-line_style solid | dot | dash | dashdot | dashdotdot]
    [-line_width <intValue>]
    [-markers_size <intValue>]
    [-markers_type circle | circlef | square | squaref | diamond |
        diamondf | cross | plus]
    [-plot <stringValue>]
    [-reset]
    [-show | -hide] [-show_legend | -hide_legend]
    [-show_line | -hide_line]
    [-show_markers | -hide_markers]
    [-xScale <doubleValue>] [-xShift <doubleValue>]
    [-yScale <doubleValue>] [-yShift <doubleValue>]
```

Argument	Description
<stringList>	List of curves on which to apply the specified properties.
-axis left right	Plots values on the left or right y-axis.
-color <#rrggbb>	Sets color of the curve. Does not apply to discrete traps.
-deriv <intValue> -integ	Options are: <ul style="list-style-type: none">Derives the curve, specifying the order of the derivative (either first order or second order).Integrates the curve.
-function <stringValue>	Applies a function to a curve. For details on functions, see Appendix D on page 405 .
-label <stringValue>	Sets a label to the curve.
-line_style solid dot dash dashdot dashdotdot	Sets style of the curve line. Does not apply to discrete traps.
-line_width <intValue>	Sets line width of the curve line. Does not apply to discrete traps.

Appendix A: Tcl Commands

set_curve_prop

Argument	Description
-markers_size <intValue>	Sets size of the markers.
-markers_type circle circlef square squaref diamond diamondf cross plus	Sets type of markers of the curve.
-plot <stringValue>	Name of the plot. If not specified, the command uses the selected plot.
-reset	Resets curve parameters.
-show -hide	Shows or hides the specified curves.
-show_legend -hide_legend	Shows or hides the curve title from the legend.
-show_line -hide_line	Shows or hides the curve line. Does not apply to discrete traps.
-show_markers -hide_markers	Shows or hides the curve markers. Does not apply to discrete traps.
-xScale <doubleValue>	Sets the x-scale of the curve.
-xShift <doubleValue>	Sets the x-shift of the curve.
-yScale <doubleValue>	Sets the y-scale of the curve.
-yShift <doubleValue>	Sets the y-shift of the curve.

Returns

None.

Example

```
set_curve_prop Curve_1 -label "NetActive Field (Cut from structure_1 at  
x=0)"
```

Appendix A: Tcl Commands

set_cutline_prop

set_cutline_prop

Changes cutline properties.

Note:

This command applies to 2D and 3D plots only.

Syntax

```
set_cutline_prop <stringValue> -plot <stringValue>
    [-handles_color <#rrggbb>]
    [-label_normal | -label_op] [-label_pos 0 | 1]
    [-label_size <doubleValue>] [-line_color <#rrggbb>]
    [-line_style solid | dot | dash | dashdot | dashdotdot]
    [-line_width <intValue>]
    [-pos1 {<x> <y> [<z>]}] [-pos2 {<x> <y> [<z>]}]
    [-show_handles | -hide_handles] [-show_label | -hide_label]
```

Argument	Description
<stringValue>	Name of the cutline from which the property will be returned.
-plot <stringValue>	Name of the plot in which the cutline is located.
-handles_color <#rrggbb>	Sets the color of the handles.
-label_normal -label_op	Sets the side of the label with respect to the endpoint of the label where the label will be displayed.
-label_pos 0 1	Sets the label position. The value can be either 0 or 1 indicating the edge of the cutline.
-label_size <doubleValue>	Sets the label size of the cutline.
-line_color <#rrggbb>	Sets the color of the cutline.
-line_style solid dot dash dashdot dashdotdot	Sets the style of the cutline.
-line_width <intValue>	Sets the width of the cutline.
-pos1 {<x> <y> [<z>]}	Sets the position of the first point of the cutline.
-pos2 {<x> <y> [<z>]}	Sets the position of the second point of the cutline.
-show_handles -hide_handles	Shows or hides the handles of the cutline.
-show_label -hide_label	Shows or hides the label of the cutline.

Appendix A: Tcl Commands

set_cutplane_prop

Returns

Integer.

Example

```
set_cutline_prop C1 -plot Plot_2D -pos1 {0.1 2.7891 0}  
#-> 0
```

set_cutplane_prop

Changes cutplane properties.

Note:

This command applies to 3D plots only.

Syntax

```
set_cutplane_prop <stringValue> -plot <stringValue>  
  [-at <doubleValue> | -normal {<X> <Y> <Z>} -origin {<X> <Y> <Z>} ]  
  [-label_position <intValue>]  
  [-label_size <doubleValue>]  
  [-show_label | -hide_label]
```

Argument	Description
<stringValue>	Name of the cutplane in which the properties will be changed.
-plot <stringValue>	Name of the plot in which the cutplane is located.
-at <doubleValue> -normal {<X> <Y> <Z>} -origin {<X> <Y> <Z>}	Sets the values of the position of the cutplane. If the cutplane is an -at type, the -at property is available. Otherwise, the normal and origin properties can be changed.
-label_position <intValue>	Sets the label position. The value can be 0, 1, or 2, indicating different corners of the cutplane.
-label_size <doubleValue>	Sets the label size of the cutplane.
-show_label -hide_label	Shows or hides the label of the cutplane.

Returns

Integer.

Appendix A: Tcl Commands

set_deformation

Example

```
set_cutplane_prop C1 -plot Plot_3D -at 0.483  
#-> 0
```

set_deformation

Sets the deformation properties for a structure, or creates a plot with an already deformed structure.

Note:

The command applies to 2D and 3D plots only.

Syntax

```
set_deformation <stringValue>  
  [-factor <doubleValue> | -reset] [-new_plot]  
  [-plot <stringValue>]
```

Argument	Description
<stringValue>	Name of the vector field to be used to deform the structure.
-factor <doubleValue> -reset	Factor to be applied to the deformation. The value must be greater than zero. If not specified, the default value is 1. Alternatively, reset the current deformation.
-new_plot	Creates a new plot.
-plot <stringValue>	Name of the plot to be used to deform the structure. If not specified, the command uses the selected plot.

Returns

String (name of affected plot).

Example

```
set_deformation -plot Plot_n60_des Displacement-V -factor 1e2  
#-> Plot_n60_des
```

Appendix A: Tcl Commands

set_ellipse_prop

set_ellipse_prop

Sets the properties of an ellipse.

Syntax

```
set_ellipse_prop <stringValue>
    [-fill_color <#rrggbb>] [-line_color <#rrggbb>]
    [-line_style solid | dot | dash | dashdot | dashdotdot]
    [-line_width <intValue>] [-p1 <doubleList>] [-p2 <doubleList>]
    [-plot <stringValue>]
```

Argument	Description
<stringValue>	Name of the ellipse.
-fill_color <#rrggbb>	Specifies the fill color for the ellipse. Default: transparent.
-line_color <#rrggbb>	Specifies line color of the ellipse.
-line_style solid dot dash dashdot dashdotdot	Specifies line style of the ellipse.
-line_width <intValue>	Specifies the line width.
-p1 <doubleList>	Specifies the upper-left corner of the <i>invisible</i> rectangle in which the ellipse is drawn.
-p2 <doubleList>	Specifies the lower-right corner of the <i>invisible</i> rectangle in which the ellipse is drawn.
-plot <stringValue>	Name of the plot where the command will search for the ellipse. If not specified, the command uses the selected plot.

Returns

String.

Example

```
set_ellipse_prop Ellipse_1 -fill_color red -line_style dash
#-> 0
```

Appendix A: Tcl Commands

set_field_prop

set_field_prop

Sets field properties for the specified field name (<stringValue>). If the field name is not specified, the properties are set for the selected field.

Note:

The command applies to 2D and 3D plots only.

Syntax

```
set_field_prop <stringValue>
    [-contour_labels_on | -contour_labels_off]
    [-custom_levels <doubleList> |
     -scale (linear | log | asinh | logabs) -levels <intValue>]
    [-geom <stringValue>] [-interpolated_values | -primary_values]
    [-label <stringValue>] [-line_color <stringValue>]
    [-line_width <intValue>]
    [-max <doubleValue>] [-max_auto | -max_fixed]
    [-min <doubleValue>] [-min_auto | -min_fixed]
    [-plot <stringValue>] [-range {<min> <max>} | -reset]
    [-show | -hide] [-show_bands | -hide_bands]
```

Argument	Description
<stringValue>	Name of the field.
-contour_labels_on -contour_labels_off	Shows or hides the contour labels of the selected field.
-custom_levels <doubleList>	Specifies a custom list of levels.
-geom <stringValue>	Name of the dataset (or geometry). If not specified, the command uses the main one from the active plot.
-interpolated_values -primary_values	Specifies whether the primary values of a cell or the interpolated values on its vertices are used for visualization (this property is only valid for fields defined on cells).
-label <stringValue>	Specifies the label of the selected field.
-line_color <color>	Sets the color of the contour lines.
-line_width <intValue>	Sets the width of the contour lines.
-max <doubleValue>	Sets the maximum value of the field.

Appendix A: Tcl Commands

set_field_prop

Argument	Description
-max_auto -max_fixed	Sets the maximum value of the field automatically, or fixes the maximum value of the field. If -max_fixed is specified and, for example, plots are linked, the change to the field values does not update the range.
-min <doubleValue>	Sets the minimum value of the field.
-min_auto -min_fixed	Sets the minimum value of the field automatically, or fixes the minimum value of the field. If -min_fixed is specified and, for example, plots are linked, the change to the field values does not update the range.
-plot <stringValue>	Name of the plot. If not specified, the command uses the selected plot.
-range {<min> <max>} -reset	Sets range of the field contour plot, or resets to the default values.
-scale linear log asinh logabs -levels <intValue>	Sets a scale with the specified <intValue> levels. The default is linear. Do not use with the -custom_levels argument.
-show -hide	Shows or hides the contour plot.
-show_bands -hide_bands	Shows or hides contour bands.

Returns

Integer.

Example

```
set_field_prop -range {-1e20 1e20}  
#-> 0
```

Appendix A: Tcl Commands

set_grid_prop

set_grid_prop

Sets grid properties.

Note:

The command applies to xy and 2D plots only.

Syntax

```
set_grid_prop
  [-align left | right]
  [-line1_color <#rrggbb>]
  [-line1_style solid | dot | dash | dashdot | dashdotdot]
  [-line1_width <intValue>]
  [-line2_color <#rrggbb>]
  [-line2_style solid | dot | dash | dashdot | dashdotdot]
  [-line2_width <intValue>]
  [-plot <stringValue>]
  [-reset] [-show | -hide] [-show_minor_lines | -hide_minor_lines]
```

Argument	Description
-align left right	Aligns of the grid to the left or right (applies to xy plots only).
-line1_color <#rrggbb>	Sets color of the major grid lines.
-line1_style solid dot dash dashdot dashdotdot	Sets style of the major grid lines (applies to xy plots only).
-line1_width <intValue>	Sets width of the major grid lines.
-line2_color <#rrggbb>	Sets color of the minor grid lines.
-line2_style solid dot dash dashdot dashdotdot	Sets style of the minor grid lines (applies to xy plots only).
-line2_width <intValue>	Sets width of the minor grid lines.
-plot <stringValue>	Name of the plot. If not specified, the command uses the selected plot.
-reset	Resets plot grid properties (applies to xy plots only).
-show -hide	Shows or hides the major grid lines.
-show_minor_lines -hide_minor_lines	Shows or hides the minor grid lines.

Appendix A: Tcl Commands

set_legend_prop

Returns

None.

Example

```
set_grid_prop -show_minor_lines
```

set_legend_prop

Sets legend properties.

These properties apply to xy plots only: `-color_bg`, `-color_fg`, `-label_font_size`, `-location`, and `-margins`.

These properties apply to 2D and 3D plots only: `-label_format`, `-nof_labels`, `-orientation`, and `-precision`.

Syntax

```
set_legend_prop
    [-color_bg <#rrggbb>]
    [-color_fg <#rrggbb>]
    [-label_font_att normal | bold | italic | underline | strikeout]
    [-label_font_color <#rrggbb>]
    [-label_font_factor <doubleValue>]
    [-label_font_family arial | courier | times]
    [-label_font_size <intValue>]
    [-label_format preferred | scientific | engineering | fixed]
    [-location top_left | top_right | bottom_left | bottom_right]
    [-margins <intList>] [-nof_labels <intValue>]
    [-orientation vertical | horizontal]
    [-plot <stringValue>]
    [-position {<x> <y>}]
    [-precision <intValue>] [-reset]
    [-show_background | -hide_background] [-size {<x> <y>}]
    [-title_font_att normal | bold | italic | underline | strikeout]
    [-title_font_color <#rrggbb>]
    [-title_font_factor <doubleValue>]
    [-title_font_family arial | courier | times]
```

Argument	Description
<code>-color_bg <#rrggbb></code>	Sets background color (xy plots only).
<code>-color_fg <#rrggbb></code>	Sets foreground color (xy plots only).

Appendix A: Tcl Commands

set_legend_prop

Argument	Description
-label_font_att normal bold italic underline strikeout	Sets the attribute for the labels font.
-label_font_color <#rrggbb>	Sets font color of labels.
-label_font_factor <doubleValue>	Sets the font size of labels by specifying a factor for resizing the font (2D and 3D plots only).
-label_font_family arial courier times	Sets labels font.
-label_font_size <intValue>	Sets the font size for the labels by specifying an integer (xy plots only).
-label_format <stringValue>	Sets label format (2D and 3D plots only).
-location top_left top_right bottom_left bottom_right	Sets legend location (xy plots only).
-margins <intList>	Sets legend margins (xy plots only).
-nof_labels <intValue>	Sets number of labels, (2D and 3D plots only).
-orientation vertical horizontal	Sets legend orientation (2D and 3D plots only).
-plot <stringValue>	Name of the plot. If not specified, the command uses the selected plot.
-position {<x> <y>}	Sets the legend position that is normalized to the window coordinates between 0 and 1.
-precision <intValue>	Sets precision of labels (2D and 3D plots only).
-reset	Resets legend properties.
-show_background -hide_background	Sets the legend background as either solid or translucent.
-size {<x> <y>}	Sets the legend size normalized to window coordinates.
-title_font_att normal bold italic underline strikeout	Sets the attribute for the legend title font.

Appendix A: Tcl Commands

set_legend_prop

Argument	Description
-title_font_color <#rrggbb>	Sets font color of legend title.
-title_font_factor <doubleValue>	Sets legend title font size using a factor to resize the font.
-title_font_family arial courier times	Sets legend title font.

Returns

None.

Example

```
set_legend_prop -nof_labels 4 -orientation horizontal
```

Appendix A: Tcl Commands

set_line_prop

set_line_prop

Sets the properties of a line.

Syntax

```
set_line_prop <stringValue>
  [-line_color <#rrggbb>]
  [-line_style solid | dot | dash | dashdot | dashdotdot]
  [-line_width <intValue>] [-p1 <doubleList>] [-p2 <doubleList>]
  [-plot <stringValue>]
```

Argument	Description
<stringValue>	Name of a line.
-line_color <#rrggbb>	Specifies the line color.
-line_style solid dot dash dashdot dashdotdot	Specifies the line style (xy plots only).
-line_width <intValue>	Specifies the line width.
-p1 <doubleList>	Specifies the start point of the line.
-p2 <doubleList>	Specifies the end point of the line.
-plot <stringValue>	Name of the plot where the command will search for the line. If not specified, the command uses the selected plot.

Returns

String.

Example

```
set_line_prop Line_1 -line_style dot -line_width 2
#-> 0
```

Appendix A: Tcl Commands

set_material_prop

set_material_prop

Sets material properties.

If `-plot` is not specified, the properties are set for the selected material.

Note:

The command applies to 2D and 3D plots only.

Syntax

```
set_material_prop <stringList>
  [-border_color <#rrggbb>] [-border_width <intValue>]
  [-color <#rrggbb>]
  [-geom <stringValue>]
  [-mesh_color <#rrggbb>] [-mesh_width <intValue>]
  [-on | -off]
  [-particles_size <doubleValue>]
  [-plot <stringValue>]
  [-show_all | -hide_all] [-show_border | -hide_border]
  [-show_bulk | -hide_bulk] [-show_field | -hide_field]
  [-show_mesh | -hide_mesh] [-show_vector | -hide_vector]
  [-translucency_level <doubleValue>]
  [-translucency_on | -translucency_off]
```

Argument	Description
<code><stringList></code>	List of materials on which to apply the specified properties.
<code>-border_color <#rrggbb></code>	Specifies the border color of the materials. Color in format RGB is expected (see Colors on page 214).
<code>-border_width <intValue></code>	Specifies the border width (in pixels) of the selected materials. The default width is 1 pixel with the following exceptions: <ul style="list-style-type: none">For depletion regions and overlays, the default width is 2 pixels.For junction lines, the default width is 3 pixels.For contacts and interfaces, the default width is 2 pixels for 2D geometries and 3 pixels for 3D geometries.
<code>-color <#rrggbb></code>	Specifies the bulk color of the materials. Color in format RGB is expected (see Colors on page 214).
<code>-geom <stringValue></code>	Name of the dataset (or geometry). If not specified, uses the main one from the active plot.
<code>-mesh_color <#rrggbb></code>	Specifies the mesh color of the materials. Color in format RGB is expected (see Colors on page 214).

Appendix A: Tcl Commands

set_material_prop

Argument	Description
-mesh_width <intValue>	Specifies the mesh width (in pixels) of the selected materials. The default width is 1 pixel with the following exception: for interface regions, the default width is 2 pixels.
-on -off	Shows or hides the material.
-particles_size <doubleValue>	Sets the size of particles of particle (kinetic Monte Carlo) material.
-plot <stringValue>	Name of the plot. If not specified, the command uses the selected plot.
-show_all -hide_all	Shows or hides all the properties of the materials.
-show_border -hide_border	Shows or hides the border of the materials.
-show_bulk -hide_bulk	Shows or hides the bulk of the materials.
-show_field -hide_field	Shows or hides the scalar fields of the materials.
-show_mesh -hide_mesh	Shows or hides the mesh of the materials.
-show_vector -hide_vector	Shows or hides the vector fields of the materials.
-translucency_level <doubleValue>	Specifies the level of translucency when -translucency_on is specified.
-translucency_on -translucency_off	Activates or deactivates the translucency of the materials.

Returns

Integer.

Example

```
set_material_prop {Oxide Silicon} -show_field  
#-> 0
```

Appendix A: Tcl Commands

set_plot_prop

set_plot_prop

Sets plot properties.

Syntax

```
set_plot_prop
    [-axes_interchanged | -not_axes_interchanged]
    [-bg_gradient | -bg_solid]
    [-caption_font_size <intValue>]
    [-caption_leader_on | -caption_leader_off]
    [-caption_material | -caption_region]
    [-color_bg <#rrggbb>] [-color_fg <#rrggbb>]
    [-color_map grayscale | default]
    [-contacts_color constant | list | map]
    [-enable_path_limit | -disable_path_limit]
    [-frame_width <intValue>] [-gradient_colors <stringValue>]
    [-keep_aspect_ratio | -not_keep_aspect_ratio]
    [-material_colors classic | vivid]
    [-path_depth <intValue>]
    [-plot <stringList>]
    [-ratio_xtoy <doubleValue>] [-reset]
    [-show | -hide] [-show_axes | -hide_axes]
    [-show_axes_label | -hide_axes_label]
    [-show_axes_title | -hide_axes_title]
    [-show_cube_axes | -hide_cube_axes]
    [-show_curve_lines | -hide_curve_lines]
    [-show_curve_markers | -hide_curve_markers]
    [-show_grid | -hide_grid] [-show_legend | -hide_legend]
    [-show_major_ticks | -hide_major_ticks]
    [-show_max_marker | -hide_max_marker]
    [-show_min_marker | -hide_min_marker]
    [-show_minor_ticks | -hide_minor_ticks] [-show_path | -hide_path]
    [-show_regions_bg | -hide_regions_bg] [-show_title | -hide_title]
    [-tdr_state <stringValue> | -tdr_state_index <intValue>]
    [-title <stringValue>]
    [-title_font_att normal | bold | italic | underline | strikeout]
    [-title_font_color <#rrggbb>]
    [-title_font_factor <doubleValue>]
    [-title_font_family arial | courier | times]
    [-title_font_size <intValue>]
    [-transformation {<x> <y> <z>}]
```

Argument	Description
-axes_interchanged -not_axes_interchanged	Interchanges axes (xy plots only).
-bg_gradient -bg_solid	Sets the plot background to display either a gradient or a solid color (2D and 3D plots only).

Appendix A: Tcl Commands

set_plot_prop

Argument	Description
-caption_font_size <intValue>	Sets the size of the caption font (3D plots only).
-caption_leader_on -caption_leader_off	Sets the visibility of the caption leaders (3D plots only).
-caption_material -caption_region	Specifies whether to display material names or region names as captions.
-color_bg <#rrggbb>	Sets the background color used when a solid background is active.
-color_fg <#rrggbb>	Sets the foreground color.
-color_map grayscale default	Sets the color map used in the plot (2D and 3D plots only). Values are: <ul style="list-style-type: none">• When set to <code>default</code>, uses normal color map (full palette).• When set to <code>grayscale</code>, uses only grayscale colors.
-contacts_color constant list map	Sets the behavior of the contact colors. The <code>list</code> and <code>map</code> options must be configured in the user preferences first.
-enable_path_limit -disable_path_limit	Activates or deactivates the path limit when the path is displayed in the plot title.
-frame_width <intValue>	Sets the plot frame width, which must be a positive integer value less than 8 (xy plots only).
-gradient_colors <stringValue>	Sets the background colors to use when a gradient background is selected (2D and 3D plots only). The argument is a list of two colors, where the first is used as the top color and the second is used as the bottom color.
-keep_aspect_ratio -not_keep_aspect_ratio	Configures the aspect ratio (2D and 3D plots only).
-material_colors classic vivid	Sets the color scheme to be used for materials (2D and 3D plots only).
-path_depth <intValue>	Sets the number of path directory names to be displayed in the plot title (when activated). This number does not consider the file name. You must specify an integer between 0 and 99.
-plot <stringList>	Specifies a list of plot names. If not specified, the command uses the selected plot.

Appendix A: Tcl Commands

set_plot_prop

Argument	Description
-ratio_xtoy <doubleValue>	Sets the x to y ratio of the plot (2D plots only).
-reset	Resets plot properties.
-show -hide	Shows or hides the plot.
-show_axes -hide_axes	Shows or hides the axes.
-show_axes_label -hide_axes_label	Shows or hides the axes labels.
-show_axes_title -hide_axes_title	Shows or hides the axes title.
-show_cube_axes -hide_cube_axes	Shows or hides cube axes (3D plots only).
-show_curve_lines -hide_curve_lines	Shows or hides the curve lines (xy plots only).
-show_curve_markers -hide_curve_markers	Shows or hides the curve markers (xy plots only).
-show_grid -hide_grid	Shows or hides the grid (xy and 2D plots only).
-show_legend -hide_legend	Shows or hides the plot legend.
-show_major_ticks -hide_major_ticks	Shows or hides the major ticks (3D plots only).
-show_max_marker -hide_max_marker	Shows or hides the maximum marker (2D and 3D plots only).
-show_min_marker -hide_min_marker	Shows or hides the minimum marker (2D and 3D plots only).
-show_minor_ticks -hide_minor_ticks	Shows or hides the minor ticks (3D plots only).
-show_path -hide_path	Shows or hides the path in the plot title.
-show_regions_bg -hide_regions_bg	Shows or hides the background color of regions.
-show_title -hide_title	Shows or hides the plot title.

Appendix A: Tcl Commands

set_plot_prop

Argument	Description
-tdr_state <stringValue> -tdr_state_index <intValue>	Sets the current TDR state from the state name or the state index (2D and 3D plots only). To display the last state, specify:-tdr_state_index -1
-title <stringValue>	Title of the plot.
-title_font_att normal bold italic underline strikeout	Sets the attribute of the title font.
-title_font_color <#rrggbb>	Sets the title font color.
-title_font_factor <doubleValue>	Multiplies the font size by a factor (2D and 3D plots only).
-title_font_family arial courier times	Sets the title font.
-title_font_size <intValue>	Sets the font size of the title (xy plots only).
-transformation {<x> <y> <z>}	Sets a linear coordinate transformation (3D plots only).

Returns

None.

Example

```
set_plot_prop -title "Example 3D Structure"
set_plot_prop -plot "Plot_n98_des Plot_n99_des" -color_bg #aa4534
```

Appendix A: Tcl Commands

set_rectangle_prop

set_rectangle_prop

Sets the properties of a rectangle.

Syntax

```
set_rectangle_prop <stringValue>
    [-fill_color <#rrggbb>] [-line_color <#rrggbb>]
    [-line_style solid | dot | dash | dashdot | dashdotdot]
    [-line_width <intValue>] [-p1 <doubleList>] [-p2 <doubleList>]
    [-plot <stringValue>]
```

Argument	Description
<stringValue>	Name of a rectangle.
-fill_color <#rrggbb>	Specifies the fill color of the rectangle. Transparency is the default (only for xy plots).
-line_color <#rrggbb>	Specifies the line color of the rectangle.
-line_style solid dot dash dashdot dashdotdot	Specifies the line style of the rectangle (only for xy plots).
-line_width <intValue>	Specifies the line width of the rectangle.
-p1 <doubleList>	Specifies the lower-left corner of the rectangle.
-p2 <doubleList>	Specifies the upper-right corner of the rectangle.
-plot <stringValue>	Name of the plot where the command will search for the rectangle. If not specified, the command uses the selected plot.

Returns

String.

Example

```
set_rectangle_prop Rectangle_1 -line_width 5
#-> 0
```

Appendix A: Tcl Commands

set_region_prop

set_region_prop

Sets region properties.

If `-plot` is not specified, the properties are set for the selected region.

Note:

The command applies to 2D and 3D plots only.

Syntax

```
set_region_prop <stringList>
  [-border_color <#rrggbb>] [-border_width <intValue>]
  [-color <#rrggbb>]
  [-geom <stringValue>]
  [-mesh_color <#rrggbb>] [-mesh_width <intValue>]
  [-on | -off]
  [-particles_size <doubleValue>]
  [-plot <stringValue>]
  [-show_all | -hide_all] [-show_border | -hide_border]
  [-show_bulk | -hide_bulk] [-show_caption | -hide_caption]
  [-show_field | -hide_field]
  [-show_mesh | -hide_mesh] [-show_vector | -hide_vector]
  [-translucency_level <doubleValue>]
  [-translucency_on | -translucency_off]
```

Argument	Description
<code><stringList></code>	List of regions of the plot where the properties will be applied.
<code>-plot <stringValue></code>	Name of the plot. If not specified, the command uses the selected plot.
<code>-geom <stringValue></code>	Name of the dataset (or geometry). If not specified, the command uses the main one from the active plot.
<code>-border_color <#rrggbb></code>	Specifies the border color of the region. Color in format RGB is expected (see Colors on page 214).
<code>-border_width <intValue></code>	Specifies the border width (in pixels) of the selected regions. The default width is 1 pixel with the following exceptions: <ul style="list-style-type: none">For depletion regions and overlays, the default width is 2 pixels.For junction lines, the default width is 3 pixels.For contacts and interfaces, the default width is 2 pixels for 2D geometries and 3 pixels for 3D geometries.
<code>-color <#rrggbb></code>	Specifies the bulk color of the region. Color in format RGB is expected (see Colors).

Appendix A: Tcl Commands

set_region_prop

Argument	Description
-mesh_color <#rrggbb>	Specifies the mesh color of the regions. Color in format RGB is expected (see Colors on page 214).
-mesh_width <intValue>	Specifies the mesh width (in pixels) of the selected regions. The default width is 1 pixel with the following exception: for interface regions, the default width is 2 pixels.
-on -off	Shows or hides the region.
-particles_size <doubleValue>	Sets the size of particles of particle (kinetic Monte Carlo) regions.
-show_all -hide_all	Shows or hides all the properties of the regions.
-show_border -hide_border	Shows or hides the border of the regions.
-show_bulk -hide_bulk	Shows or hides the bulk of the regions.
-show_caption -hide_caption	Shows or hides captions (3D plots only).
-show_field -hide_field	Shows or hides the scalar fields of the regions.
-show_mesh -hide_mesh	Shows or hides the mesh of the regions.
-show_vector -hide_vector	Shows or hides the vector fields of the regions.
-translucency_level <doubleValue>	Specifies the level of translucency when the option <code>-translucency_on</code> is specified.
-translucency_on -translucency_off	Activates or deactivates translucency of the regions.

Returns

Integer.

Example

```
set_region_prop {source gate drain} -show_mesh  
#-> 0
```

Appendix A: Tcl Commands

set_ruler_prop

set_ruler_prop

Sets ruler properties.

Note:

This command applies to 2D and 3D plots only.

Syntax

```
set_ruler_prop
    [-color <#rrggbb>] [-id <intValue>]
    [-plot <stringValue>]
    [-pos1 <stringValue>] [-pos2 <stringValue>]
    [-precision <intValue>]
    [-show_label | -hide_label]
    [-snap_on | -snap_off] [-width <intValue>]
```

Argument	Description
-color <#rrggbb>	Sets the color of the ruler.
-id <intValue>	Specifies the ID of the ruler where the command will search for properties. If not specified, the command uses the last selected ruler.
-plot <stringValue>	Name of the plot where the command will apply the properties. If not specified, the command uses the selected plot.
-pos1 <stringValue>	Sets the position of the first point of the ruler.
-pos2 <stringValue>	Sets the position of the second point of the ruler.
-precision <intValue>	Sets the decimal precision of the measurements.
-show_label -hide_label	Specifies whether to show or hide the ruler label.
-snap_on -snap_off	Specifies whether to activate the snap-to-mesh feature.
-width <intValue>	Sets the width of the ruler in pixels.

Returns

String.

Example

```
set_ruler_prop -width 5 -precision 2
#-> 0
```

Appendix A: Tcl Commands

set_streamline_prop

set_streamline_prop

Sets the properties of streamlines.

Note:

This command applies to 2D and 3D plots only.

Syntax

```
set_streamline_prop <stringList> -plot <stringValue>
    [-arrow_angle <intValue>] [-arrow_color <#rrggbb>]
    [-arrow_size <doubleValue>] [-arrow_step <doubleValue>]
    [-arrow_style solid | dash | dot | dashdot | dashdotdot]
    [-arrow_width <intValue>] [-constant_arrow | -normal_arrow]
    [-line_color <#rrggbb>] [-line_resolution <doubleValue>]
    [-line_style solid | dash | dot | dashdot | dashdotdot]
    [-line_width <intValue>]
    [-positive_direction | -negative_direction]
    [-show_arrows | -hide_arrows] [-show_line | -hide_line]
```

Argument	Description
<stringList>	List of the streamlines to be modified.
-plot <stringValue>	Name of the plot where the command will search for streamlines. If not specified, the command uses the selected plot.
-arrow_angle <intValue>	Specifies the arrowhead angle in degrees. It must be between 1 and 89.
-arrow_color <#rrggbb>	Specifies the color of the arrows.
-arrow_size <doubleValue>	Specifies the size of the arrows.
-arrow_step <doubleValue>	Specifies the step between arrows. This value must be greater than the line resolution.
-arrow_style solid dash dot dashdot dashdotdot	Specifies the arrow line style.
-arrow_width <intValue>	Specifies the width of the arrowheads.
-constant_arrow -normal_arrow	Specifies whether the size of the arrowheads on screen does not change regardless of the zoom level (-constant_arrow), or whether the size of the arrowheads changes on screen depending on the zoom level (-normal_arrow).

Appendix A: Tcl Commands

set_streamline_prop

Argument	Description
-line_color <#rrggbb>	Specifies the color of the line.
-line_resolution <doubleValue>	Specifies the distance between the points that conform the line. Lower values imply better line quality but lower performance.
-line_style solid dash dot dashdot dashdotdot	Specifies the line style.
-line_width <intValue>	Specifies the width of the line.
-positive_direction -negative_direction	Specifies whether the arrow will be shown in the normal view or inverted view. Default: -positive_direction.
-show_arrows -hide_arrows	Shows or hides the arrows.
-show_line -hide_line	Shows or hides the line.

Returns

Integer.

Example

```
set_streamline_prop Streamline_1 -plot Plot_2D -arrow_angle 45  
#-> 0
```

Appendix A: Tcl Commands

set_tag_prop

set_tag_prop

Prints text in a box.

The text displayed can be changed only with the argument `-custom_text`. The size of the text depends on the box size.

Note:

The command applies only to 2D and 3D plots.

Syntax

```
set_tag_prop [-custom_text <stringValue>] [-plot <stringValue>]  
[-show | -hide]
```

Argument	Description
<code>-custom_text <stringValue></code>	Specifies the text to be displayed. The text is not shown unless <code>-show</code> is specified.
<code>-plot <stringValue></code>	Names of the plot where the tag will be displayed. If not specified, the command uses the selected plot.
<code>-show -hide</code>	Shows or hides the text.

Returns

Integer.

Example

```
set_tag_prop -plot Plot_2D -custom_text "Test to be displayed." -show  
#->0
```

Appendix A: Tcl Commands

set_textbox_prop

set_textbox_prop

Sets the specified properties of a text box.

Note:

This command applies to xy and 2D plots only.

Syntax

```
set_textbox_prop <stringValue>
    [-anchor_pos {<x> <y>}] [-arrow_size <intValue>]
    [-font_att normal | bold | italic | underline | strikeout]
    [-font_color <#rrggbb>] [-font_factor <doubleValue>]
    [-font_family arial | courier | times] [-font_size <intValue>]
    [-line_color <#rrggbb>]
    [-line_style solid | dot | dash | dashdot | dashdotdot]
    [-line_width <intValue>]
    [-plot <stringValue>]
    [-pos {<x> <y>}] [-rotation <intValue>]
    [-show_anchor | -hide_anchor] [-show_border | -hide_border]
    [-text <stringValue>]
```

Argument	Description
<stringValue>	Name of the text box to be modified.
-anchor_pos {<x> <y>}	Specifies the anchor position using the world coordinate system (only for 2D plots).
-arrow_size <intValue>	Specifies the arrow size (only for 2D plots).
-font_att normal bold italic underline strikeout	Specifies the font attribute of the text. Several attributes can be combined using braces. For example: -font_att {bold italic}
-font_color <#rrggbb>	Specifies the color of the text font.
-font_factor <doubleValue>	Specifies the multiplier for the text font (only for 2D plots).
-font_family arial courier times	Specifies the text font family.
-font_size <intValue>	Specifies the font size (only for xy plots).
-line_color <#rrggbb>	Specifies the line and arrow color (only for 2D plots).
-line_style solid dot dash dashdot dashdotdot	Specifies the representation style of the text box line (only for 2D plots).

Appendix A: Tcl Commands

set_textbox_prop

Argument	Description
-line_width <intValue>	Specifies the width of the text box and anchor line (only for 2D plots).
-plot <stringValue>	Name of the plot where the command will search for the text box. If not specified, the command uses the selected plot.
-pos {<x> <y>}	Specifies the lower-left corner position of the text box. For xy plots, this is a point in the world coordinate system {x, y}. For 2D plots, this is a relative normalized screen coordinates pair (from 0.0 to 1.0).
-rotation <intValue>	Specifies the rotation of the text box in degrees (only for xy plots).
-show_anchor -hide_anchor	Shows or hides the text box anchor (only for 2D plots).
-show_border -hide_border	Shows or hides the text box border (only for 2D plots).
-text <stringValue>	Specifies the text in the text box.

Returns

Integer.

Example

```
set_textbox_prop Textbox_1 -text "Label Text" -font_color #ff0000  
#-> 0
```

Appendix A: Tcl Commands

set_transformation

set_transformation

Applies a transformation to a certain geometry. You can scale a geometry, shift a geometry, or both scale and shift a geometry.

Note:

This command applies to 2D and 3D plots only.

Syntax

```
set_transformation
  -scale {<scaleX> <scaleY> <scaleZ>} |
  -shift {<shiftX> <shiftY> <shiftZ>} |
  -scale {<scaleX> <scaleY> <scaleZ>} -shift {<shiftX> <shiftY> <shiftZ>}
  [-geom <stringValue>] [-plot <stringValue>]
```

Argument	Description
-scale {<scaleX> <scaleY> <scaleZ>}	Sets or returns the scale value of the axis. The list has the form {x y z} where the parameters x, y, and z are positive doubles that represent the scale value applied to each axis.
-shift {<shiftX> <shiftY> <shiftZ>}	Sets or returns the shift value of the axis. The list has the form {x y z} where the parameters x, y, and z are doubles that represent the shift value applied to each axis.
-geom <stringValue>	Name of the geometry in which the transformation will be applied.
-plot <stringValue>	Name of the plot from which the geometry will be obtained.

Returns

Integer.

Example

```
set_transformation -scale {0.5 1 1}
#-> 0
```

Appendix A: Tcl Commands

set_value_blankning

set_value_blankning

Sets value blanking.

If `-field` is not specified, the command uses the selected field.

Note:

The command applies to 2D and 3D plots only.

Syntax

```
set_value_blankning -field <stringValue>
    -less <doubleValue> | -greater <doubleValue> | -reset
    [-blank all | any | inter] [-cons <intValue>]
    [-plot <stringValue>] [-union | -intersection]
```

Argument	Description
<code>-field <stringValue></code>	Name of the field to set blanking parameters.
<code>-less <doubleValue> -greater <doubleValue> -reset</code>	If you specify <code>-less</code> , all values less than the specified value are blanked. If you specify <code>-greater</code> , all values greater than the specified value are blanked. Specify <code>-reset</code> to remove value blanking rules.
<code>-blank all any inter</code>	Selects the value blanking option where: <ul style="list-style-type: none">• <code>all</code> is all vertices.• <code>any</code> is any vertex.• <code>inter</code> is interpolate vertices. If not specified, the command uses the <code>all</code> option.
<code>-cons <intValue></code>	Number of the value blanking rule. Options are between 1 and 8. Default: 1.
<code>-plot <stringValue></code>	Name of the plot. If not specified, the command uses the selected plot.
<code>-union -intersection</code>	Sets whether the constraints will be united or will intersect. If not specified, the command uses the <code>-union</code> option.

Returns

Integer.

Appendix A: Tcl Commands

set_vector_prop

Example

```
set_value_blanking -field DopingConcentration -greater 0.0  
#-> 0
```

set_vector_prop

Sets the properties of a vector field.

Note:

This command applies to 2D and 3D plots only.

Syntax

```
set_vector_prop <stringValue>  
[-constant_heads | -normal_heads] [-fill_color <#rrggbb>]  
[-geom <stringValue>]  
[-head_angle <intValue>]  
[-head_shape arrow | arrow_solid | head | head_closed | head_solid]  
[-head_size <doubleValue>]  
[-line_color <#rrggbb>]  
[-line_pattern solid | dot | dash | dashdot | dashdotdot]  
[-line_width <intValue>]  
[-plot <stringValue>]  
[-scale_grid | uniform]  
[-scale_factor_grid <doubleValue>]  
[-scale_factor_uniform <doubleValue>] [-show | -hide]
```

Argument	Description
<stringValue>	Name of the vector field to be modified.
-constant_heads -normal_heads	Specifies whether the arrows are constant to the plot area regardless of the vector magnitude or proportional (normal) to the vector magnitude.
-fill_color <#rrggbb>	Specifies the color of a solid arrowhead. Otherwise, this argument has no effect.
-geom <stringValue>	Specifies the geometry where the command will search for the vector.
-head_angle <intValue>	Specifies the arrowhead angle in degrees. It must be between 1 and 89.
-head_shape arrow arrow_solid head head_closed head_solid	Specifies the shape of the arrows.

Appendix A: Tcl Commands

set_vector_prop

Argument	Description
-head_size <doubleValue>	Specifies the length of the arrows.
-line_color <#rrggbb>	Specifies the color of the arrows.
-line_pattern solid dot dash dashdot dashdotdot	Specifies the line pattern of the arrows.
-line_width <intValue>	Specifies the width of the arrows.
-plot <stringValue>	Name of the plot where the command will search for the geometry. If not specified, the command uses the selected plot.
-scale grid uniform	Specifies the scale for displaying the arrows, either uniform size or a grid display.
-scale_factor_grid <doubleValue>	Specifies the grid factor for displaying the arrows.
-scale_factor_uniform <doubleValue>	Specifies the uniform factor for displaying the arrows.
-show -hide	Shows or hides the arrows.

Returns

Integer.

Example

```
set_vector_prop ElectricField-V -plot Plot_2D -geom 2D -scale grid  
#-> 0
```

Appendix A: Tcl Commands

set_vertical_lines_prop

set_vertical_lines_prop

Sets the properties of vertical lines.

Note:

This command applies to xy plots only.

Syntax

```
set_vertical_lines_prop <stringValue>
    [-line_color <#rrggbb>]
    [-line_style solid | dot | dash | dashdot | dashdotdot]
    [-line_width <intValue>] [-plot <stringValue>] [-show | -hide]
```

Argument	Description
<stringValue>	Name of a vertical line.
-line_color <#rrggbb>	Specifies the color of the vertical line.
-line_style solid dot dash dashdot dashdotdot	Specifies the style of the vertical line.
-line_width <intValue>	Specifies the width of the vertical line in pixels.
-plot <stringValue>	Name of the plot where the command searches for the vertical line. If not specified, the command uses the selected plot.
-show -hide	Shows or hides the vertical line.

Returns

Integer.

Example

```
set_vertical_lines_prop VerticalLine _1 -line_style dot
#-> 0
```

Appendix A: Tcl Commands

set_window_full

set_window_full

Sets the full plot view.

Syntax

```
set_window_full -on | -off
```

Argument	Description
-on -off	Activates or deactivates the full plot view.

Returns

Integer.

Example

```
set_window_full -on  
#-> 0
```

set_window_size

Sets the size of the main window of the user interface.

Syntax

```
set_window_size <width>x<height>
```

Argument	Description
<width>x<height>	Sets the width and the height of the main window in pixels (minimum of 200x200 pixels).

Returns

Integer.

Example

```
set_window_size 1280x800  
#-> 0
```

Appendix A: Tcl Commands

show_msg

show_msg

Displays a message in a dialog box.

Syntax

```
show_msg <stringValue>
    [-error | -info | -warning] [-title <stringValue>]
```

Argument	Description
<stringValue>	Specifies the text to be displayed.
-error -info -warning	Specifies the type of message to display. Default: -info.
-title <stringValue>	Specifies the title of the dialog box.

Returns

None.

Example

```
show_msg -warning -title "Bad Value" "There was a problem extracting the
threshold voltage"
```

Appendix A: Tcl Commands

start_movie

start_movie

Starts the recording of a new movie by creating a new frame buffer.

Note:

This command only starts the creation of a movie, so you must use the `add_frame` and `export_movie` commands to complete the operations.

Syntax

```
start_movie [-resolution <width>x<height>]
```

Argument	Description
<code>-resolution <width>x<height></code>	Specifies the resolution of each captured frame in pixels. If not specified, uses the current screen resolution.

Returns

None.

Example

```
start_movie
```

stop_movie

Stops recording a movie.

Note:

This command deletes the stored frame buffer. It does not save it into a file.

Syntax

```
stop_movie
```

Returns

None.

Example

```
stop_movie
```

Appendix A: Tcl Commands

undo

undo

Undoes the last command implemented or the number of commands specified.

Syntax

```
undo [<intValue>]
```

Argument	Description
<intValue>	Number of commands to be reverted.

Returns

None.

Example

```
undo 2
```

unload_file

Unloads all the datasets belonging to the specified file.

Syntax

```
unload_file <stringValue>
```

Argument	Description
<stringValue>	Name of the file.

Returns

Integer.

Example

```
unload_file structure2D.tdr  
#-> 0
```

version

Returns the version of Sentaurus Visual.

Syntax

`version`

Returns

String.

Example

```
version  
#-> 31.0.7
```

Appendix A: Tcl Commands

windows_style

windows_style

Specifies the type of window style to use for the user interface of Sentaurus Visual.

Syntax

```
windows_style
  [-aspect_ratio_on | -aspect_ratio_off]
  [-direction right_down | down_right]
  [-max <intValue>]
  [-sort <stringList>]
  [-style horizontal | vertical | grid | max | custom]
```

Argument	Description
-aspect_ratio_on -aspect_ratio_off	Specifies whether the aspect ratio is maintained for all the plots displayed.
-direction right_down down_right	Specifies the viewing direction of plots and where they will stretch: <ul style="list-style-type: none">When using the <code>right_down</code> direction, the grid fills to the right until it is full and then continues adding new plots in a new row downwards from the first row.When using the <code>down_right</code> direction, this order is inverted.
-max <intValue>	Specifies the maximum number of columns in which to display the plots when they are in a custom grid configuration.
-sort <stringList>	Specifies the plots to be displayed.
-style horizontal vertical grid max custom	Specifies a horizontal or vertical orientation, or grid style, or the use of maximum space or custom style.

Returns

Integer.

Example

```
windows_style -style grid
#-> 0
```

Appendix A: Tcl Commands

zoom_plot

zoom_plot

Zooms into a plot.

Syntax

```
zoom_plot  
    -axis (x | y | z) -range {<min> <max>} |  
    -box {<minX> <maxX> <minY> <maxY> <minZ> <maxZ>} |  
    -factor <doubleValue> |  
    -reset |  
    -window {<x1> <y1> [<z1>]<x2> <y2> [<z2>]}  
    [-plot <stringValue>]
```

Argument	Description
-axis x y z	Specifies the axis where the range is applied.
-range {<min> <max>}	The <code>-range</code> argument defines the minimum and maximum values of the range.
-box {<minX> <maxX> <minY> <maxY> <minZ> <maxZ>}	Defines the three ranges for the boundary box.
-factor <doubleValue>	Sets the zoom factor. If the value is greater than 1, it zooms into a plot. If the value is smaller than 1, it zooms out of a plot.
-reset	Resets the zoom status of the plot.
-window {<x1> <y1> [<z1>] <x2> <y2> [<z2>]}	Sets the zoom window. It zooms into the specified window between two x,y pairs. For 2D and 3D plots, the argument accepts only four values. For 3D plots, the values can be entered in the form of pixels of the plot frame (integers) or can be normalized to screen values (doubles between 0 and 1).
-plot <stringValue>	Name of the plot.

Returns

None.

Example

```
zoom_plot -reset
```

B

Python Commands

This appendix describes the Python commands that can be used in Sentaurus Visual.

The Python commands apply to the same plots and structures defined in the Tcl commands in [Appendix A on page 213](#).

General Information

Sentaurus Visual Python Mode imports the `svisual` package as `sv` automatically at the start of the application. This package contains the commands presented in this appendix.

As in Tcl mode, `sv.echo()` prints to the log file `SVisualPy.log`, and the `print()` function acts as the Tcl `puts` command printing to standard output.

Accessing Documentation for Python Commands

To access the documentation for the Python commands, from the Sentaurus Visual GUI, choose **Help > Python API Reference**.

Alternatively, you can use the `help()` function to access command information:

```
help(sv.<command_name>)
```

Syntax Conventions

Each Python command is implemented as a Python function and, as such, it must comply with the Python syntax conventions.

Appendix B: Python Commands

Common Properties

Get Property Commands

These commands share the syntax `get_<object>_prop()`. All these commands have a positional parameter called `property`, which receives a string with the name of the property requested. See the corresponding Tcl command for a list of the possible properties to query.

Set Property Commands

These commands share the syntax `set_<object>_prop()` and have several keyword arguments, one for each property to be set.

Common Properties

The following properties are used in several Python commands.

Colors

In commands that allow you to specify color properties (such as the `color` argument), a string specifying red, green, and blue components of the RGB system is expected. The string is preceded by a hash (#) character, and each value is provided in hexadecimal form. Common colors also have aliases.

Table 15 *Common colors*

Alias	General form	Description
white	#ffffff	White
black	#000000	Black
red	#ff0000	Red
darkRed	#800000	Dark red
lime	#00ff00	Light green
green	#008000	Dark green
darkGreen	#006400	Darker green
blue	#0000ff	Blue

Appendix B: Python Commands

Common Properties

Table 15 Common colors (*Continued*)

Alias	General form	Description
darkBlue	#000080	Dark blue
cyan	#00ffff	Cyan
darkCyan	#008080	Dark cyan
magenta	#ff00ff	Magenta
darkMagenta	#800080	Dark magenta
yellow	#ffff00	Yellow
olive	#808000	Olive or dark yellow
orange	#ffa500	Orange
darkOrange	#ff8c00	Dark orange
gray	#a0a0a4	Gray
darkGray	#808080	Dark gray
lightGray	#c0c0c0	Light gray
skyblue	#87ceeb	Sky blue
slategray	#708090	Slate gray
chocolate	#d2691e	Chocolate

Appendix B: Python Commands

Common Properties

Fonts

For commands that allow you to adjust font properties, Sentaurus Visual defines a specific list of font families and attributes.

Table 16 Font families and their attributes

Font family	Attribute
Arial	Bold
Courier	Italic
Times	Normal
	Strikeout
	Underline

Note:

In xy plots, the font size of different elements of the plot are set with the `font_size` argument; whereas in 2D and 3D plots, the font size cannot be set directly. Instead, the font size is set as a factor of the plot frame (the default value is 1.0), with the `font_factor` argument.

Lines

For commands that allow you to adjust line properties (such as the `line_style` option), Sentaurus Visual defines a specific list of line styles. You can provide the name of the style or its short form directly.

Table 17 Line styles

Name of line style	Short form of line style	Description
solid	—	Continuous line: _____
dot	.	Dotted line:
dash	-	Dashed line: -----
dashdot	-.	Alternating dash-and-dot line: -.-.-.-.-
dashdotdot	-..	Alternating dash-and-two-dots line: -...-...-..-

Appendix B: Python Commands

Common Properties

Markers

Different markers are available to use in xy plots in Sentaurus Visual. You can use the name or the short form of each marker.

Table 18 Marker types

Name of marker type	Short form of marker type	Description
circle	o	○
circlef	of	●
diamond		◇
diamondf		◆
square		□
squaref		■
plus	+	+
cross	x	x

C

Menus and Toolbars of User Interface

This appendix describes the menus and toolbars of the user interface of Sentaurus Visual.

Menus

This section lists the commands of the different menus.

File Menu

Table 19 File menu

Command	Button	Shortcut keys	Description
Open		Ctrl+O	Loads a dataset or multiple datasets.
Reload All		F5 key	Reloads all loaded datasets.
Reload Selected		Shift+F5	Reloads only the selected datasets.
Automatic Reload Dataset			Displays Automatic Reload Dataset dialog box, where you set up automatic reloading of datasets.
Export Plot		Ctrl+E	Exports the selected plots to an image.
Import Image		Ctrl+I	Displays the Import Image dialog box.
Run Script			Runs Tcl, Python, or Inspect scripts.
Print Plots		Ctrl+P	Prints the selected plots.

Appendix C: Menus and Toolbars of User Interface

Menus

Table 19 File menu (Continued)

Command	Button	Shortcut keys	Description
Recent Files			Lists the recently opened datasets, up to five.
Exit		Ctrl+Q	Quits Sentaurus Visual.

Edit Menu

Table 20 Edit menu

Command	Button	Shortcut keys	Description
Undo		Ctrl+Z	Reverts the last operation executed.
Select All Plots		Ctrl+A	Selects all the active plots.
Redraw All Plots		Ctrl+R	Redraws all the active plots.
Delete Selected Plots		Ctrl+D	Deletes all the selected plots.
Preferences			Displays User Preferences dialog box.
Draw			Displays the Draw toolbar for drawing lines, rectangles, and ellipses, and inserting text. Available for xy and 2D plots only.

View Menu

Table 21 View menu

Command	Button	Shortcut keys	Description
Panels			Shows or hides the Data Selection panel, Properties panel, and Tcl or Python Console.
Toolbars			Shows or hides the File, Edit, View, Tools, and Movies toolbars.
Select			Enables selection (default) mode.

Appendix C: Menus and Toolbars of User Interface

Menus

Table 21 View menu (Continued)

Command	Button	Shortcut keys	Description
Select/Rotate			Enables selection (default) mode. Available for 3D points only.
Reset		Ctrl+Shift+F	Resets plot to the default values.
Zoom		Ctrl+Shift+Z	Enables zoom tool.
Scale to Image			Displays the Scale to Image dialog box, where you can overlay an image onto a plot.
Zoom to Ranges			Displays the Zoom to Ranges dialog box, where you can zoom by specifying the range of one of the three axes using the Box tab. Available for 3D plots only.
Best Look		Ctrl+Shift+L	Adjusts plotting parameters automatically. Available for xy plots only.
Spherical Rotation			Performs a spherical rotation of the view. Available for 3D plots only.
Rotation Axis X			Fixes the rotation of a 3D plot to the x-axis. Available for 3D plots only.
Rotation Axis Y			Fixes the rotation of a 3D plot to the y-axis. Available for 3D plots only.
Rotation Axis Z			Fixes the rotation of a 3D plot to the z-axis. Available for 3D plots only.
Rotate			Displays Rotate dialog box for rotate modes and angles for 3D plots. Available for 3D plots only.
View Plane XY			Shows a 3D plot in the xy plane. Available for 3D plots only.
View Plane YZ			Shows a 3D plot in the yz plane. Available for 3D plots only.
View Plane XZ			Shows a 3D plot in the xz plane. Available for 3D plots only.

Appendix C: Menus and Toolbars of User Interface

Menus

Table 21 View menu (Continued)

Command	Button	Shortcut keys	Description
Default View			Restores a 3D plot point of view. Available for 3D plots only.
Fast Draw			If selected, 3D plot becomes an outline during a rotation or move. Available for 3D plots only.
Subsampling			Activates or deactivates subsampling in 2D and 3D plots. Available for 2D and 3D plots only.
Camera Configuration			Camera configuration for 3D plots. Available for 3D plots only.
Lights Configuration			Lighting parameters for 3D plots. Available for 3D plots only.

Tools Menu

Table 22 Tools menu

Command	Button	Shortcut keys	Description
Link		Ctrl+L	Links two or more plot properties.
Special Link			Displays Special Link dialog box where you can set up special linking to link only specified properties.
Movies			Provides commands to start recording a movie, to add frames to a movie, and to stop recording a movie.
Probe		Ctrl+Shift+P	Probes the values on a plot.
Analysis			Performs analysis on a curve. Available for xy plots only.
Calculate Scalar			Displays the Calculate Scalar dialog box, where you can create a function to calculate scalar values. Available for xy plots only. See Performing Complex Mathematical Operations on 1D Data on page 95.

Appendix C: Menus and Toolbars of User Interface

Menus

Table 22 Tools menu (Continued)

Command	Button	Shortcut keys	Description
Precision Cuts			Displays the Cutlines and Cutplanes dialog box. Available for 2D and 3D plots only.
Cutline		Ctrl+Shift+C	Generates a custom cutline on a 2D plot. Available for 2D and 3D plots only.
Cut X		Ctrl+Shift+X	Generates a cutplane (3D) or cutline (2D) orthogonal to the x-axis. Available for 2D and 3D plots only.
Cut Y		Ctrl+Shift+Y	Generates a cutplane (3D) or cutline (2D) orthogonal to the y-axis. Available for 2D and 3D plots only.
Cut Z		Ctrl+Shift+Z	Generates a cutplane (3D) or cutline (2D) orthogonal to the z-axis. Available for 2D and 3D plots only.
Ruler		Ctrl+Shift+R	Enables measuring distances. Available for 2D and 3D plots only.
Value Blanking		Ctrl+Shift+V	Displays Value Blanking dialog box. Available for 3D plots only.
Streamlines			Displays Streamlines dialog box where you can enable drawing streamlines of a vector field. Available for 2D and 3D plots only.
Overlay		Ctrl+Shift+Y	Overlays two or more plots onto one plot. Available for 2D and 3D plots only.
Diff Plots			Enables tool to plot the difference between common fields. Available for 2D and 3D plots only.
Integrate			Displays the Field Integration dialog box, where you can perform integration over the active field of a plot. Available for 2D and 3D plots only.
Create Projection			Displays the 2D Projection dialog box, where you can create a 2D minimum or maximum projection of a field from a 3D plot. Available for 3D plots only.
Deformation			Displays Deformation dialog box, where you can create a deformed structure in the same plot or in a new one. Available for 2D and 3D plots only.

Appendix C: Menus and Toolbars of User Interface

Menus

Table 22 Tools menu (Continued)

Command	Button	Shortcut keys	Description
Min/Max Field Value			Displays Minimum/Maximum Field Value dialog box, where you can select certain regions or materials for the search, and you can define a 3D box limiting the search area. Available for 2D and 3D plots only.
Create Isovalue			Displays Create Isovalue Geometry dialog box, where you can create a new geometry from a constant field value in a structure. Available for 2D and 3D plots only.
Surface Plot			Displays Surface Plot dialog box, where you can create a surface plot from a 3D dataset.
Extract Path			Displays Extract Path dialog box, where you can extract a path of either the minimum or maximum values of a specified scalar field.

Data Menu

Table 23 Data menu

Command	Button	Shortcut keys	Description
View Info Loaded			Displays Manage Loaded Data dialog box, showing all the datasets and plots currently loaded.
Curve Properties		Ctrl+Shift+E	Displays Curve Properties dialog box. Available for xy plots only.
Region Properties		Ctrl+Shift+E	Displays Region Properties dialog box. Available for 2D and 3D plots only.
Export XY Data			Displays Export XY Data dialog box. Available for xy plots only.
Save Plot			Displays a dialog box where you can save all the plot data and settings to a Tcl file. Available for xy plots only.
New XY Plot			Generates a new empty xy plot. Available for xy plots only.

Appendix C: Menus and Toolbars of User Interface

Menus

Table 23 Data menu (Continued)

Command	Button	Shortcut keys	Description
Duplicate Plot			Duplicates the current plot as an xy plot. Available for xy plots only.
TDR Tags			Displays TDR Tags dialog box, where you can select which tags to display on the selected plot. Available for 2D and 3D plots only.
Dataset Information			Displays Dataset Information dialog box, where you can access 2D or 3D dataset information, such as the number of points or elements in a specific material or region. Available for 2D and 3D plots only.

Window Menu

Table 24 Window menu

Command	Button	Shortcut keys	Description
Tile Grid			Organizes loaded plots into a grid.
Tile Vertically			Organizes loaded plots vertically in the plot area.
Tile Horizontally			Organizes loaded plots horizontally in the plot area.
Set Default State			Restores the toolbars and workspace to their default positions in the user interface.
Manage Frames			Displays the Manage Frames dialog box.
Previous Plot		Page Up key	Moves to the previous loaded plot.
Next Plot		Page Down key	Moves to the next loaded plot.
Minimize Plot			Minimizes the selected plot.
Maximize		F10 key	Maximizes the selected plot.
Full Plot View		F12 key	Hides the toolbars and zooms into a plot using the entire workspace.

Appendix C: Menus and Toolbars of User Interface

Menus

Table 24 *Window menu (Continued)*

Command	Button	Shortcut keys	Description
Restore All Plots			Restores all minimized plots.
Plots			Lists the open plots.

Help Menu

Table 25 *Help menu*

Command	Button	Shortcut keys	Description
User Guide			Opens PDF file of <i>Sentaurus™ Visual User Guide</i> .
Python API Reference			Opens the API that documents the Python commands available in Sentaurus Visual.
Tutorial			Opens Sentaurus Visual module of TCAD Sentaurus Tutorial (HTML).
About		Ctrl+B	Shows information about Sentaurus Visual.

Note:

The default viewer for the PDF file of the *Sentaurus™ Visual User Guide* is Adobe® Reader®. Using another PDF viewer might deactivate some cross-references or external links in the PDF file.

Appendix C: Menus and Toolbars of User Interface

Toolbars

Toolbars

This section describes the different toolbars.

File Toolbar

Table 26 File toolbar

Button	Description	Button	Description
	Open		Export Plot
	Reload All		Run Script
	Reload Selected		Print Plots

Edit Toolbar

Table 27 Edit toolbar

Button	Description	Button	Description
	Undo		Draw (xy and 2D plots only)

Draw Toolbar

Table 28 Draw toolbar (available for xy and 2D plots only)

Button	Description	Button	Description
	Draw Line		Draw Ellipse
	Draw Rectangle		Insert Text

Appendix C: Menus and Toolbars of User Interface

Toolbars

View Toolbar

Table 29 View toolbar

Button	Description	Button	Description
	Select Select/Rotate (3D plots only)		Rotation Axis X (3D plots only)
	Reset		Rotation Axis Y (3D plots only)
	Zoom		Rotation Axis Z (3D plots only)
	Best Look (xy plots only)		View Plane XY (3D plots only)
	Log Scale X (xy plots only)		View Plane YZ (3D plots only)
	Log Scale Y (xy plots only)		View Plane XZ (3D plots only)
	Log Scale Y Right (xy plots only)		Fast Draw (3D plots only)
	Spherical Rotation (3D plots only)		

Tools Toolbar

Table 30 Tools toolbar

Button	Description	Button	Description
	Link		Cut X (2D and 3D plots only)
	Special Link		Cut Y (2D and 3D plots only)
	Curve Properties (xy plots only)		Cut Z (2D and 3D plots only)
	Region Properties (2D and 3D plots only)		Ruler (2D and 3D plots only)

Appendix C: Menus and Toolbars of User Interface

Toolbars

Table 30 Tools toolbar (Continued)

Button	Description	Button	Description
	Probe		Value Blanking (3D plots only)
	Analysis (xy plots only)		Streamlines (2D and 3D plots only)
	Plot Band Diagram (xy plots only)		Overlay (2D and 3D plots only)
	Precision Cuts (2D and 3D plots only)		Diff Plots (2D and 3D plots only)
	Cutline (2D plots only)		Integrate (2D and 3D plots only)

Movies Toolbar

Table 31 Movies toolbar

Button	Description	Button	Description
	Start Recording		Add Frames
	Stop Recording		

Look Toolbar

Table 32 Look toolbar

Button	Description	Button	Description
	Change Panel View (Changes presentation of left pane from separate tabs to one view)		Tcl Console (Shows or hides Tcl Console)
	Properties Panel (Shows or hides properties panel for whichever plot is selected)		Python Console (Shows or hides Python Console)

Appendix C: Menus and Toolbars of User Interface

Additional Keyboard Shortcuts (2D and 3D Plots)

Table 32 Look toolbar (Continued)

Button	Description	Button	Description
 Data	Data Selection Panel (Shows or hides Data Selection panel)		

Additional Keyboard Shortcuts (2D and 3D Plots)

Table 33 Additional keyboard shortcuts for 2D and 3D plots

Action	Shortcut keys	Description
Basic rotation	Press the N key while dragging the cursor	Enables <i>rollerball</i> rotation until you release the N key (applies to 3D plots only).
Rotate around x-axis	Press the X key while dragging the cursor	Enables rotation around the x-axis until you release the X key (applies to 3D plots only).
Rotate around y-axis	Press the Y key while dragging the cursor	Enables rotation around the y-axis until you release the Y key (applies to 3D plots only).
Rotate around z-axis	Press the Z key while dragging the cursor	Enables rotation around the z-axis until you release the Z key (applies to 3D plots only).
Spherical rotation	Press the S key while dragging the cursor	Enables spherical rotation until you release the S key (applies to 3D plots only).
Change rotation center point	Press O key	Updates the rotation point of the structure. A new point will be placed at the cursor position.
Switch on or off 3D guide axis	Press I key	Switches on or off the 3D guide axis. However, to see this change, a minor rotation is required.
Enable zoom navigation	Press Ctrl+Shift while dragging the cursor	Enables zoom navigation (equivalent to clicking and dragging the middle mouse button).
Zoom to cursor position	Press F key	When you place the cursor somewhere on the structure (you do not click) and then press the F key, the structure changes view so that the new center of the plot is where the cursor was placed.
Highlight material or region menu	Double-click	Highlights the region or material in blue in the Data Selection panel.

Appendix C: Menus and Toolbars of User Interface

Additional Keyboard Shortcuts (2D and 3D Plots)

Table 33 Additional keyboard shortcuts for 2D and 3D plots (Continued)

Action	Shortcut keys	Description
Highlight region	Press P key	Highlights the selected region using a red box. To cancel this operation, press the P key when the cursor is not positioned over any region.
Reset view	Press R key	Returns the structure to the default view.
Enable wireframe view	Press W key	Changes the display of the structure to a mesh view.
Enable solid view	Press S key	Changes the display of the structure to a non-mesh view.

D

Available Formulas

This appendix presents an overview of the functions available in Sentaurus Visual as well as the syntax of the formulas used to create curves, variables, and fields.

Creating a New Variable

Note:

For new variables, variables of an existing dataset can be used in the function specification or a list of values.

To create a new variable, use the `create_variable` command. For example, to create the common logarithm of the variable Y present in the dataset `myDataset` as a new variable, you can use the command:

```
create_variable -name commonLogY -dataset "myDataset"  
-function "log(<Y:myDataset>)"
```

To access variables on functions, use the format `<VARIABLE:DATASET>`. This new variable will appear in the variables list of the dataset in which it was created.

Note:

Variables can be created on the **Data** tab of the Data Selection panel by clicking **New Variable**. A dialog box is displayed where you can interactively add functions, operators, and variables to create a new formula.

Creating a New Curve

Note:

For new curves, existing curves can be used in the function specification.

To create a new curve, use the `create_curve` command. For example, to create the derivative of `Curve_1` and name it `newCurve`, you can use the command:

```
create_curve -name newCurve -function diff(<Curve_1>)
```

Appendix D: Available Formulas

Applying Functions to a Curve

To use the curves on formulas, you must write the curve identifier in angle brackets. For example, to use the data on `Curve_1` for the differentiation function, it is written as `<Curve_1>`.

Note:

If you want to create a new curve from more than one curve using a function. For example:

```
create_curve -name newCurve_2 -function <Curve_1>*<Curve_2>
```

both curves `Curve_1` and `Curve_2` must share the same x-axis and must have the same amount of valid data. Otherwise, this could lead to unexpected results.

Curves can be created on the **Curves** tab of the Data Selection panel by clicking the **New** button. A dialog box is displayed where you can interactively add functions, operators, and curves to create a new curve based on a formula.

Applying Functions to a Curve

To apply a function to an existing curve, use the `set_curve_prop` command. For example, to apply the absolute value function to `Curve_1`, use the command:

```
set_curve_prop Curve_1 -function "abs"
```

Alternatively, you can use the Curve Properties panel:

1. Select the curve.
2. Click the **Trans.** tab.
3. From the **Function** list, select the required function.

Note:

It is not possible to apply more than one function to an existing curve. Instead, it is recommended to create a new curve.

Furthermore, as an exception, you can apply the integral, or a first-derivative or second-derivative function in addition to the other function, using the same command, but with another parameter (`-integ` or `-deriv`):

```
set_curve_prop Curve_1 -integ  
set_curve_prop Curve_1 -deriv 2
```

Alternatively, you can use the Curve Properties panel:

1. Select the curve.
2. Click the **Trans.** tab.
3. From the **Deriv / Integ** list, select the function to apply.

Appendix D: Available Formulas

Creating a New Field

Creating a New Field

Note:

For new fields, existing fields can be used in the function specification.

To create a new field, use the `create_field` command. Existing fields are used to create new fields based on functions and operations specified by the user. In the following example, consider two fields called `ElectricField-X` and `ElectricField-Y`. You want to create a new field that contains the absolute value of the sum of both fields. This can be done with the following command:

```
create_field -name AbsSumElectricField -dataset 2D  
-function "abs(<ElectricField-X>+<ElectricField-Y>)" -show
```

Note:

New fields also can be created on the **More** tab of the Data Selection panel by clicking **Add Field**.

Available Functions

[Table 34 on page 408](#) lists the available functions. The function arguments are:

- *Double*: Numeric values, scalar field names.
- *Vector*: Vector field names.
- *Curve*: 1D curve names.

For example:

```
-function "sin(<ElectricField-X>+<ElectricField-Y>)" (Double or Scalar)  
-function "sin(<ElectricField-V>)" (Vector)  
-function "sin(<Curve_1>)" (Curve)
```

Note:

For functions that only return a Double value and are used as the outer function in formulas, the result will be displayed only in a dialog box and cannot be used in Tcl scripts.

Appendix D: Available Formulas

Available Functions

Table 34 Available functions

Function	Arguments	Returns	Description
abs(x)	Double	Double	Absolute value
	Vector	Vector	
	Curve	Curve	
acos(x)	Double	Double	ArcCosine
	Vector	Vector	
	Curve	Curve	
acosh(x)	Double	Double	Hyperbolic ArcCosine
	Vector	Vector	
	Curve	Curve	
asin(x)	Double	Double	ArcSine
	Vector	Vector	
	Curve	Curve	
asinh(x)	Double	Double	Hyperbolic ArcSine
	Vector	Vector	
	Curve	Curve	
atan(x)	Double	Double	ArcTangent
	Vector	Vector	
	Curve	Curve	
atanh(x)	Double	Double	Hyperbolic ArcTangent
	Vector	Vector	
	Curve	Curve	
bessel_j0(x)	Double	Double	Bessel function of first kind, order zero
	Vector	Vector	
	Curve	Curve	
bessel_j1(x)	Double	Double	Bessel function of first kind, first order
	Vector	Vector	
	Curve	Curve	
bessel_y0(x)	Double	Double	Bessel function of second kind, order zero
	Vector	Vector	
	Curve	Curve	

Appendix D: Available Formulas

Available Functions

Table 34 Available functions (Continued)

Function	Arguments	Returns	Description
bessel_y1(x)	Double Vector Curve	Double Vector Curve	Bessel function of second kind, first order
cbrt(x)	Double Vector Curve	Double Vector Curve	Cube root of x
ceil(x)	Double Vector Curve	Double Vector Curve	Approximates to the next integer
cfftim(x,y)	x: Vector, Curve y: Vector, Curve	Vector Curve	Fast Fourier transform, imaginary value
cfftre(x,y)	x: Vector, Curve y: Vector, Curve	Vector Curve	Fast Fourier transform, real value
cifftim(x,y)	x: Vector, Curve y: Vector, Curve	Vector Curve	Inverse fast Fourier transform, imaginary value
cifftre(x,y)	x: Vector, Curve y: Vector, Curve	Vector Curve	Inverse Fourier transform, real value
cos(x)	Double Vector Curve	Double Vector Curve	Cosine
cosh(x)	Double Vector Curve	Double Vector Curve	Hyperbolic cosine
crop(y, x, min, max) crop(c, min, max)	x: Vector y: Vector min: Double max: Double c: Curve	Vector Curve	Crops the values depending of the minimum and maximum range of x
diff(c) diff(y,x)	x: Vector y: Vector c: Curve	Vector Curve	First-order derivative

Appendix D: Available Formulas

Available Functions

Table 34 Available functions (Continued)

Function	Arguments	Returns	Description
erf(x)	Double Vector Curve	Double Vector Curve	Error function
erfc(x)	Double Vector Curve	Double Vector Curve	Complementary error function
exp(x)	Double Vector Curve	Double Vector Curve	Evaluates $e^{(x)}$
fftabs(y,x)	x: Vector, Curve y: Vector, Curve	Vector Curve	Fast Fourier transform, absolute value
fftim(x)	Vector Curve	Vector Curve	Fast Fourier transform, imaginary value
fftre(x)	Vector Curve	Vector Curve	Fast Fourier transform, real value
floor(x)	Double Vector Curve	Double Vector Curve	Approximates to the previous integer
gamma(x)	Double Vector Curve	Double Vector Curve	Gamma function
ifftim(x)	Vector Curve	Vector Curve	Inverse Fourier transform, imaginary value
ifftre(x)	Vector Curve	Vector Curve	Inverse Fourier transform, real value
integr(y,x) integr(c)	y: Vector x: Vector c: Curve	Vector Curve	Integrates the vector y over the range specified by x, or integrates the curve c
inverse(x)	Double Vector Curve	Double Vector Curve	Inverse value

Appendix D: Available Formulas

Available Functions

Table 34 Available functions (Continued)

Function	Arguments	Returns	Description
lgamma(x)	Double Vector Curve	Double Vector Curve	Logarithmic gamma function
log(x)	Double Vector Curve	Double Vector Curve	Natural logarithm
log10(x)	Double Vector Curve	Double Vector Curve	Common logarithm
minmax(x, min, max)	x: Vector min: Double max: Double	Vector	All values that are less than min are replaced with min, and all values greater than max are replaced with max
pow(x,y)	x: Double x: Vector x: Curve y: Double	Double Vector Curve	Evaluates xy , where x is a double value, a vector of values, or a curve
rms(x,y)	x: Vector x: Curve y: Vector y: Curve	Double	Root mean square value
sign(x)	Double Vector Curve	Double Vector Curve	Sign
sin(x)	Double Vector Curve	Double Vector Curve	Sine
sinh(x)	Double Vector Curve	Double Vector Curve	Hyperbolic sine
sqrt(x)	Double Vector Curve	Double Vector Curve	Square root

Appendix D: Available Formulas

Available Functions

Table 34 Available functions (Continued)

Function	Arguments	Returns	Description
<code>tan(x)</code>	Double Vector Curve	Double Vector Curve	Tangent
<code>tangent(c, v)</code> <code>tangent(y, x, v)</code>	x: Vector y: Vector c: Curve v: Double	Vector Curve	Creates a tangent line in the point v on the curve c, or the curve defined by the vectors x and y
<code>tanh(x)</code>	Double Vector Curve	Double Vector Curve	Hyperbolic tangent
<code>vecmax(x)</code> <code>vecmax(c)</code>	x: Vector c: Curve	Double	Returns the maximum y-value of a curve, or the maximum value of a vector
<code>vecmin(x)</code> <code>vecmin(c)</code>	x: Vector c: Curve	Double	Returns the minimum y-value of a curve, or the minimum value of a vector
<code>vecvalx(y, x, v)</code> <code>vecvalx(c, v)</code>	x: Vector y: Vector c: Curve v: Double	Double	Returns the x-value when y=v of a curve
<code>vecvaly(y, x, v)</code> <code>vecvaly(c, v)</code>	x: Vector y: Vector c: Curve v: Double	Double	Returns the y-value when x=v of a curve
<code>veczero(y, x)</code> <code>veczero(c)</code>	x: Vector y: Vector c: Curve	Double	Returns the x-value when y=0 of a curve

E

Inspect Support in Sentaurs Visual

This appendix provides information about the level of support for running Inspect scripts in Sentaurs Visual.

The support of Inspect commands is available only if the commands are in a script file and it is loaded using one of the following options:

- From the command line, for example, pass an Inspect script file as an argument with the corresponding option:

```
svvisual -inspect test_ins.cmd
```

- From the user interface, choose **File > Run Script**. In the Open Script File dialog box, select **Inspect Command File (*.cmd)** in the **Files of type** field.

- Load a Tcl script using the `load_script_file` Tcl command. For example:

```
load_script_file test_ins.cmd -inspect
```

Fully Supported Commands

Sentaurs Visual fully supports the following Inspect commands:

cv_abs	cv_compute	cv_create	cv_createDS
cv_createFromScript	cv_createWithFormula	cv_delete	cv_display
cv_getVals	cv_getValsX	cv_getValsY	cv_getXaxis
cv_getYaxis	cv_getZero	cv_lineColor	cvLineStyle
cv_log10Scale	cv_logScale	cv_printVals	cv_split
cv_split_disc			
f_Gamma	f_gm	f_IDSS	f_KP

Appendix E: Inspect Support in Sentaurus Visual

Partially Supported Commands

f_Ron	f_Rout	f_TetaG	f_VT
ft_scalar			
gr_createLabel	gr_formatAxis	gr_mappedAxis	gr_precision
gr_setGridAttr	gr_setLegend	gr_setLegendPos	
load_library	macro_define		
proj_getDataSet	proj_getList	proj_getNodeList	proj_load
proj_unload			
script_exit	script_sleep		

Partially Supported Commands

Sentaurus Visual only partially supports the Inspect commands listed in [Table 35](#).

Table 35 *Partially supported Inspect commands*

Command	Limitations
cv_renameCurve	Works only if the curve is not displayed.
cv_set_interp	Works only if the curve is displayed.
cv_setCurveAttr	Cannot set color and width of the marker outline. Cannot set the fill color of the marker. Triangle marker is not available.
gr_setAxisAttr	Cannot set color and width of the axis line. Cannot set number of secondary ticks and angle at which the tick labels are drawn.
gr_setGeneralAttr	Only background color can be set.
gr_setLegendAttr	Cannot set frame color, width position, and anchor.
grSetTitleAttr	Cannot set title justification.
script_break	Suspends the script, displaying a message.

Appendix E: Inspect Support in Sentaurus Visual

Not Supported Commands

Not Supported Commands

Sentaurus Visual does not support the following Inspect commands:

cv_delPts	cv_inv	cv_reset	cv_write
f_hideInternalCurves	f_showInternalCurves	f_VT1	f_VT2
fi_writeBitmap	fi_writeEps	fi_writePs	
gb_setpreferences	gr_deleteLabel	graph_load	graph_write
param_load	param_write	proj_write	

Script Library Support

This section explains the support Sentaurus Visual provides for different Inspect script libraries.

Extraction Library

All the commands from this library are fully supported only if they are calculated over displayed curves:

ExtractEarlyV	ExtractGm	ExtractGmb	ExtractIoff
ExtractMax	ExtractRon	ExtractSS	ExtractValue
ExtractVtgm	ExtractVtgm	ExtractVti	

If the curve is created but not displayed, the result will be the same for all commands except ExtractIoff because the interpolation will be linear not logarithmic.

Curve Comparison Library

Both commands generate a new curve with specific visual properties of the marker, which are not necessarily the same as in Inspect, that is, there is a visual difference:

- cvcmp_CompareTwoCurves
- cvcmp_DeltaTwoCurves

The extend Library

Sentaurus Visual only partially supports the commands listed in [Table 36](#).

Table 36 Partially supported commands of extend library

Command	Limitations
<code>cv_autoIncrStyle</code> <code>cv_disp</code>	Depends on <code>cv_setCurveAttr</code> , which is not fully supported. For details about the limitations of <code>cv_setCurveAttr</code> , see Table 35 on page 414 .
<code>cv_nextSymbol</code> <code>cv_setSymbol</code>	Triangle marker is not available.

Sentaurus Visual does not support the following commands:

<code>cv_exists</code>	<code>cv_resetFillColor</code>	<code>cv_setFillColor</code>	<code>ds_getValue</code>
<code>proj_check</code>	<code>proj_datasetExists</code>	<code>proj_getGroups</code>	<code>proj_groupExists</code>

F

Extraction Library

This appendix provides information about the procedures of the extraction library.

The procedures of the extraction library are used to extract various parameters from the I–V characteristics of various device types. The extraction library takes I–V data in the form of two Tcl lists: one list contains the voltages points and the other list contains the corresponding current values.

The extraction library is loaded automatically when Sentaurus Visual starts. However, if you have switched off the automatic loading of extension libraries, then you can load the extraction library explicitly with the command:

```
load_library extract
```

Syntax Conventions

The extraction library uses a unique namespace identifier (`ext:::`) for its procedures. All procedures and variables associated with this library are called with the namespace identifier prepended. For example:

```
ext:::<proc_name>
```

Each procedure has several arguments. The extraction library uses an input parser that accepts arguments of the form:

```
-keyword <value>
```

Note:

All Sentaurus Visual libraries support the standard Sentaurus Visual syntax in which keywords are preceded by a dash. For backward compatibility, all Sentaurus Visual libraries continue to support the `keyword= <value>` syntax as well. For each procedure call, you can use either the `-keyword <value>` syntax or the `keyword= <value>` syntax. However, within any one procedure call, only one type of syntax can be used. Otherwise, an error message will be generated. Only the new syntax is documented. If you want to continue using the

Appendix F: Extraction Library

Syntax Conventions

keyword= <value> syntax, you also can insert space between the keyword and the equal sign, for example, keyword = <value>. Omitting the space between the equal sign and the value field will result in a failure if the value is a de-referenced Tcl variable. Use keyword= \$val (*not* keyword=\$val).

The parser accepts arguments in any order. For some arguments, default values are predefined. Such arguments can be omitted. If arguments for which no defaults are predefined are omitted, the procedure will exit with an error message. In addition, unrecognized arguments result in an error message.

Instead of using the standard Tcl method of using the return value of the procedure to pass results back to the calling program, the extraction library uses a *passing-by-reference* method to return the results to the calling program. The procedure keyword -out is used to pass the results back to the calling program:

-out <var_name>, <list_name>, or <array_name>

The following conventions are used for the syntax of Tcl commands:

- Angle brackets – <> – indicate text that must be replaced, but they are *not* part of the syntax. In particular, the following type identifiers are used:
 - <r>: Replace with a real number, or a de-referenced Tcl variable that evaluates to a real number. For example: \$val.
 - <i>: Replace with an integer, or a de-referenced Tcl variable that evaluates to an integer. For example: \$i.
 - <string>: Replace with a string, or a de-referenced Tcl variable that evaluates to a string. For example: \$file.
 - <list_of_r>: Replace with a list of real numbers, or a de-referenced Tcl variable that evaluates to a list of real numbers. For example: \$values.
 - <list_of_strings>: Replace with a list of strings, or a de-referenced Tcl variable that evaluates to a list of strings. For example: \$files.
 - <var_name>: Replace with the *name* of a local Tcl variable.
For example: val (*not* \$val).
 - <list_name>: Replace with the *name* of a local Tcl list.
For example: values (*not* \$values).
 - <array_name>: Replace with the *name* of a local Tcl array.
For example: myarray (*not* \$myarray).

Appendix F: Extraction Library

Help for Procedures

- Brackets – [] – indicate that the argument is optional, but they are *not* part of the syntax.
- A vertical bar – | – indicates options, only one of which can be specified.

Help for Procedures

To request help on a specific procedure, in Tcl mode, set the `-help` keyword to 1:

```
ext::<procedure_name> -help 1
```

If this command is included in a Sentaurus Visual file, when Sentaurus Visual is executed in:

- Batch mode in Sentaurus Workbench, the help information is printed to the runtime output file (with the extension `.out`) of the corresponding Sentaurus Visual node.
- Interactive mode in Sentaurus Workbench, the help information is displayed in the Tcl Console as well as printed in the Sentaurus Visual output file.

You also can enter the command in the Tcl Console of the user interface, in which case, the help information is displayed in the Console.

Output of Procedures

As discussed in [Syntax Conventions on page 417](#), all procedures of the extraction library pass the results back to the calling program by storing the results in a Tcl variable. The name of this Tcl variable is specified as the value of the `-out` keyword. All procedures beginning with `ext::Extract` extract a device parameter. For example, the procedure

`ext::ExtractVtgm` extracts the threshold voltage:

```
ext::ExtractVtgm -out Vt -name Vtgm -v $Vgs -i $absIds
```

Here, since `-out Vt` is used, the extracted threshold voltage is stored in the Tcl variable `Vt`.

All procedures of the extraction library beginning with `ext::Extract` pass the extracted value to the Sentaurus Workbench Family Tree (if the `-name` keyword differs from "noprint"). The extracted quantity is displayed as a Sentaurus Workbench variable.

If `-name "noprint"` is used, the extracted variable is not passed to the Sentaurus Workbench Family Tree. If `-name out` is used, the name of the variable specified by the `-out` keyword also is used as the name that appears in the Sentaurus Workbench Family Tree.

Here, since `-name Vtgm` is used, the extracted threshold voltage value is displayed as the Sentaurus Workbench variable `Vtgm`.

If there are errors in the extraction library procedures, the behavior of Sentaurus Visual depends on whether it is executed in batch mode or interactive mode in Sentaurus

Appendix F: Extraction Library

Output of Procedures

Workbench. In batch mode, Sentaurus Visual exits and an error message is printed only in the Sentaurus Visual error file (with the extension .err). In interactive mode, the error message is displayed in the Tcl Console as well as printed in the Sentaurus Visual error file. You can handle the errors raised by the procedures of the extraction library, for example, by using the Tcl `catch` command.

All procedures also print several messages (including warning messages). If Sentaurus Visual is executed in batch mode, the messages are printed only in the Sentaurus Visual output file; whereas, in interactive mode, the messages are displayed in the Tcl Console as well as printed in the Sentaurus Visual output file.

The amount of information printed depends on the information level specified by the procedure `lib::SetInfoDef`. Irrespective of the specified information level, the extracted value is printed in the output file by the procedures beginning with `ext::Extract`.

For example, if the information level is set to 0 for all procedures using the `lib::SetInfoDef` procedure:

```
lib::SetInfoDef 0
ext::ExtractVtgm -out Vt -name Vtgm -v $Vgs -i $absIds
```

the following message is printed:

```
DOE: Vtgm 0.316
```

If the information level for the procedure `ext::ExtractVtgm` is set to 1 using the `lib::SetInfoDef` procedure:

```
lib::SetInfoDef 1
ext::ExtractVtgm -out Vt -name Vtgm -v $Vgs -i $Ids -vo 1e-4
```

or by using the `-info` keyword:

```
lib::SetInfoDef 0
ext::ExtractVtgm -out Vt -name Vtgm -v $Vgs -i $Ids -vo 1e-4 -info 1
```

the following message is printed:

```
DOE: Vtgm 0.316
Vtgm (Max gm method): 0.316
```

If the extraction library procedure cannot extract the parameter, the parameter is set to the character 'x' and a message is printed. In the case of `ext::ExtractVtgm`, the following message is printed:

```
DOE: Vtgm x
ext::ExtractVtgm: Vtgm not found!
```

Appendix F: Extraction Library

ext::AbsList

ext::AbsList

Computes the absolute value of all elements of a list.

Syntax

```
ext::AbsList -out <list_name> -x <list_of_r>
  [-info 0 | 1 | 2 | 3] [-help 0 | 1]
```

Argument	Description
-out <list_name>	Name of a list to store the list of absolute values. (List name, no default)
-x <list_of_r>	Input list. (List of real numbers, no default)
-info 0 1 2 3	Sets local information level. Default: 0
-help 0 1	Prints a help screen if set to 1. Default: 0

Returns

None.

Example

```
load_file DATA/IdVgLin_pMOS_des.plt -name DC
set Ids [get_variable_data "drain TotalCurrent" -dataset DC]
puts "Ids: $Ids"

# Compute absolute value of drain currents
ext::AbsList -out Idabs -x $Ids
puts "Idabs: $Idabs"

#-> Ids: -5.40e-10 -9.56e-10 -9.11723e-08 ... -6.73e-05
#-> Idabs: 5.40e-10 9.56e-10 9.11723e-08 ... 6.73e-05
```

Appendix F: Extraction Library

ext::DiffForwardList

ext::DiffForwardList

Computes the first-order derivative of a curve using the forward finite difference method. The curve is represented by two Tcl lists: one contains the x-values (independent variable) and one contains the corresponding y-values (dependent variable).

Syntax

```
ext::DiffForwardList -out <array_name> -x <list_of_r> -y <list_of_r>
    [-info 0 | 1 | 2 | 3] [-help 0 | 1]
```

Argument	Description
-out <array_name>	Name of an array to store the results. The array has one string-valued index. The index contains the elements <code>x</code> and <code>dy</code> . The values of the <code>x</code> element and the <code>dy</code> element are lists of x-values and first-order derivatives, respectively. (Array name, no default)
-x <list_of_r>	List containing the x-values (independent variable). (List of real numbers, no default)
-y <list_of_r>	List containing the y-values (dependent variable). (List of real numbers, no default)
-info 0 1 2 3	Sets local information level. Default: 0
-help 0 1	Prints a help screen if set to 1. Default: 0

Returns

None.

Example

```
set Xs [list 1.0 2.0 3.0 4.0]
# Generate Ys using y=2*x
set Ys [list]
foreach x $Xs {
    lappend Ys [expr 2*$x]
}

ext::DiffForwardList -out DyDx -x $Xs -y $Ys

puts "x-values: $Xs"
puts "xnew-values: $DyDx(x)"
puts "y-values: $Ys"
puts "derivative: $DyDx(dy)"
```

Appendix F: Extraction Library

ext::DiffList

```
#-> x-values: 1.0 2.0 3.0 4.0
#-> xnew-values: 1.5 2.5 3.5
#-> y-values: = 2.0 4.0 6.0 8.0
#-> derivative: 2.0 2.0 2.0
```

ext::DiffList

Computes the first-order derivative of a curve. The curve is represented by two Tcl lists: one contains the x-values (independent variable) and one contains the corresponding y-values (dependent variable).

Note:

The procedure `ext::DiffList` uses the central finite difference method to compute the derivative at a data point. This method uses the x- and y-values of two adjacent points, which are computed internally by the procedure using either linear or logarithmic interpolation.

Syntax

```
ext::DiffList -out <list_name> -x <list_of_r> -y <list_of_r>
              [-yLog 0 | 1] [-xLog 0 | 1] [-info 0 | 1 | 2 | 3] [-help 0 | 1]
```

Argument	Description
<code>-out <list_name></code>	Name of a list to store the first-order derivative. (List name, no default)
<code>-x <list_of_r></code>	List containing the x-values (independent variable). (List of real numbers, no default)
<code>-y <list_of_r></code>	List containing the y-values (dependent variable). (List of real numbers, no default)
<code>-yLog 0 1</code>	Selects linear (0) or logarithmic (1) interpolation for y-axis values for computing the derivative. Default: 0
<code>-xLog 0 1</code>	Selects linear (0) or logarithmic (1) interpolation for x-axis values for computing the derivative. Default: 0
<code>-info 0 1 2 3</code>	Sets local information level. Default: 0
<code>-help 0 1</code>	Prints a help screen if set to 1. Default: 0

Returns

None.

Appendix F: Extraction Library

ext::ExtractBVi

Example

```
set Xs [list 1 1.5 2.5 6 7 7.5 8.5 8.7 8.8 10]
# Generate Ys using y=exp(x)+1
set Ys [list]
foreach x $Xs {
    lappend Ys [expr exp($x) + 1]
}

# For exponential function, use logarithmic interpolation
# for y-axis values
set yLog 1
ext::DiffList -out dydx -x $Xs -y $Ys -yLog $yLog

puts "x-values: $Xs"
puts "y-values: $Ys"
puts "derivative: $dydx"

#-> x-values: 1 1.5 2.5 6 7 7.5 8.5 8.7 8.8 10
#-> y-values: 3.718 5.481 13.182 ... 22027.465
#-> derivative: 2.887 4.532 12.231 ... 22012.323
```

ext::ExtractBVi

Extracts the breakdown voltage from an I–V curve. The breakdown voltage is defined as the bias voltage at which the current reaches a certain level. The curve is represented by two Tcl lists: one contains the voltage points and one contains the corresponding current values.

Syntax

```
ext::ExtractBVi -out <var_name> -v <list_of_r> -i <list_of_r> -io <r>
    [-name <string>] [-f <string>] [-info 0 | 1 | 2 | 3] [-help 0 | 1]
```

Argument	Description
-out <var_name>	Variable name to store the value of the breakdown voltage. (Real number, no default)
-v <list_of_r>	List containing the voltage values. (List of real numbers, no default)
-i <list_of_r>	List containing the current values. (List of real numbers, no default)
-io <r>	Current level. (Nonzero real number, no default)

Appendix F: Extraction Library

ext::ExtractBVv

Argument	Description
-name <string>	Name of the extracted variable to appear in the Sentaurus Workbench Family Tree. Note: If -name "noprint" is used, Sentaurus Workbench extraction is suppressed. If -name "out" is used, the name of the variable specified by the -out keyword also is used as the name that appears in the Sentaurus Workbench Family Tree.
	(String, default: "BVi")
-f <string>	Format string used to write the extracted variable to the Sentaurus Workbench Family Tree. (String, default: "% .3e")
-info 0 1 2 3	Sets local information level. Default: 0
-help 0 1	Prints a help screen if set to 1. Default: 0

Returns

None.

Example

```
# Extract breakdown voltage of a n-p-n bipolar transistor
load_file DATA/IcVcBV_npn_des.plt -name BV
set Vcbs [get_variable_data "collector InnerVoltage" -dataset BV]
set Ics [get_variable_data "collector TotalCurrent" -dataset BV]

ext::ExtractBVi -out BVcboi -name "out" -v $Vcbs -i $Ics -io 1e-12 \
-f "% .3f"
```

ext::ExtractBVv

Extracts the breakdown voltage from an I–V curve. The breakdown voltage is defined as the maximum voltage that can be applied to a contact. The curve is represented by two Tcl lists: one contains the voltage points and one contains the corresponding current values.

Syntax

```
ext::ExtractBVv -out <var_name> -v <list_of_r> -i <list_of_r>
-sign <+1 | -1>
[-name <string>] [-f <string>] [-info 0 | 1 | 2 | 3] [-help 0 | 1]
```

Appendix F: Extraction Library

ext::ExtractBVv

Argument	Description
-out <var_name>	Variable name to store the value of the breakdown voltage. (Real number, no default)
-v <list_of_r>	List containing the voltage values. (List of real numbers, no default)
-i <list_of_r>	List containing the current values. (List of real numbers, no default)
-sign <+1 -1>	Distinguishes different types of bipolar transistor: +1: n-p-n transistor -1: p-n-p transistor In general, set -sign -1 if the breakdown occurs at a negative bias. (No default)
-name <string>	Name of the extracted variable to appear in the Sentaurus Workbench Family Tree. Note: If -name "noprint" is used, Sentaurus Workbench extraction is suppressed. If -name "out" is used, the name of the variable specified by the -out keyword also is used as the name that appears in the Sentaurus Workbench Family Tree.
	(String, default: "BVv")
-f <string>	Format string used to write the extracted variable to the Sentaurus Workbench Family Tree. (String, default: "% .2e")
-info 0 1 2 3	Sets local information level. Default: 0
-help 0 1	Prints a help screen if set to 1. Default: 0

Returns

None.

Example

```
# Extract breakdown voltage of a n-p-n bipolar transistor
load_file DATA/IcVcBV_npn_des.plt -name BV
set Vcbs [get_variable_data "collector InnerVoltage" -dataset BV]
set Ics [get_variable_data "collector TotalCurrent" -dataset BV]

ext::ExtractBVv -out BVcbov -name "out" -v $Vcbs -i $Ics -sign 1 \
-f "%.3f"
```

Appendix F: Extraction Library

ext::ExtractEarlyV

ext::ExtractEarlyV

Extracts the Early voltage from an I_c - V_{ce} curve. The curve is represented by two Tcl lists: one contains the voltage points and one contains the corresponding current values.

Syntax

```
ext::ExtractEarlyV -out <var_name> -v <list_of_r> -i <list_of_r>
    -vo <r>
    [-name <string>] [-f <string>] [-info 0 | 1 | 2 | 3] [-help 0 | 1]
```

Argument	Description
-out <var_name>	Variable name to store the value of the early voltage. (Real number, no default)
-v <list_of_r>	List containing the voltage values. (List of real numbers, no default)
-i <list_of_r>	List containing the collector current values. (List of real numbers, no default)
-vo <r>	Bias point at which the slope of the I_c - V_{ce} curve is determined for the computation of the Early voltage. (Real number, no default)
-name <string>	Name of the extracted variable to appear in the Sentaurus Workbench Family Tree. Note: If -name "noprint" is used, Sentaurus Workbench extraction is suppressed. If -name "out" is used, the name of the variable specified by the -out keyword also is used as the name that appears in the Sentaurus Workbench Family Tree. (String, default: "va")
-f <string>	Format string used to write the extracted variable to the Sentaurus Workbench Family Tree. (String, default: "% .3e")
-info 0 1 2 3	Sets local information level. Default: 0
-help 0 1	Prints a help screen if set to 1. Default: 0

Returns

None.

Appendix F: Extraction Library

ext::ExtractExtremum

Example

```
# Extract Early voltage for a p-n-p bipolar transistor
load_file DATA/IcVc_pnp_des.plt -name IcVce

set Vcs [get_variable_data "collector OuterVoltage" -dataset IcVce]
set Ics [get_variable_data "collector TotalCurrent" -dataset IcVce]
ext::AbsList -out absIcs -x $Ics

# Compute absolute value of collector current
ext::ExtractEarlyV -out Va -name "out" -v $Vcs -i $absIcs -vo -1.25 \
-f "%.2f"
```

ext::ExtractExtremum

Extracts the maximum or minimum of a curve. The curve is represented by two Tcl lists: one contains the x-values and one contains the corresponding y-values.

Syntax

```
ext::ExtractExtremum -out <var_name> -x <list_of_r> -y <list_of_r>
[-type "max" | "min"] [-name <string>] [-f <string>]
[-info 0 | 1 | 2 | 3] [-help 0 | 1]
```

Argument	Description
-out <var_name>	Variable name to store the maximum or minimum of the curve. (Real number, no default)
-x <list_of_r>	List containing the x-values. (List of real numbers, no default)
-y <list_of_r>	List containing the y-values. (List of real numbers, no default)
-type "max" "min"	Selects whether to extract the minimum ("min") or maximum ("max") of a curve. Default: "max"
-name <string>	Name of the extracted variable to appear in the Sentaurus Workbench Family Tree.
Note:	
If -name "noprint" is used, Sentaurus Workbench extraction is suppressed.	
If -name "out" is used, the name of the variable specified by the -out keyword also is used as the name that appears in the Sentaurus Workbench Family Tree.	
(String, default: "out")	

Appendix F: Extraction Library

ext::ExtractGm

Argument	Description
-f <string>	Format string used to write the extracted variable to the Sentaurus Workbench Family Tree. (String, default: "%.3e")
-info 0 1 2 3	Sets local information level. Default: 0
-help 0 1	Prints a help screen if set to 1. Default: 0

Returns

None.

Example

```
load_file DATA/IdVgLin_nMOS_des.plt -name DC
set Vgs [get_variable_data "gate OuterVoltage" -dataset DC]
set Ids [get_variable_data "drain TotalCurrent" -dataset DC]

ext::ExtractExtremum -out IdLin -name "out" -x $Vgs -y $Ids -type "max"
```

ext::ExtractGm

Extracts the maximum transconductance from an I_d - V_{gs} curve. The transconductance g_m is defined as:

$$g_m = \frac{dI_d}{dV_g} \quad (1)$$

The gate bias at which the maximum transconductance occurs is computed using parabolic interpolation. The curve is represented by two Tcl lists: one contains the voltage points and one contains the corresponding current values.

Syntax

```
ext::ExtractGm -out <var_name> -v <list_of_r> -i <list_of_r>
               [-name <string>] [-f <string>] [-info 0 | 1 | 2 | 3] [-help 0 | 1]
```

Argument	Description
-out <var_name>	Variable name to store the value of the maximum transconductance. (Real number, no default)
-v <list_of_r>	List containing the gate voltage values. (List of real numbers, no default)

Appendix F: Extraction Library

ext::ExtractGm

Argument	Description
-i <list_of_r>	List containing the drain current values. (List of real numbers, no default)
-name <string>	Name of the extracted variable to appear in the Sentaurus Workbench Family Tree. Note: If -name "noprint" is used, Sentaurus Workbench extraction is suppressed. If -name "out" is used, the name of the variable specified by the -out keyword also is used as the name that appears in the Sentaurus Workbench Family Tree.
	(String, default: "gm")
-f <string>	Format string used to write the extracted variable to the Sentaurus Workbench Family Tree. (String, default: "% .3e")
-info 0 1 2 3	Sets local information level. Default: 0
-help 0 1	Prints a help screen if set to 1. Default: 0

Returns

None.

Example

```
# Extract gm for a p-MOSFET
load_file DATA/IdVgLin_pMOS_des.plt -name DC
set Vgs [get_variable_data "gate OuterVoltage" -dataset DC]
set Ids [get_variable_data "drain TotalCurrent" -dataset DC]

ext::AbsList -out absIds -x $Ids
ext::ExtractGm -out gmLin -name "out" -v $Vgs -i $absIds

puts "Max gm: [format %.3e $gmLin] S/um"
#-> Max gm: 1.913e-04 S/um
```

Appendix F: Extraction Library

ext::Extractloff

ext::Extractloff

Extracts the drain leakage current at the specified gate voltage from an I_d - V_{gs} curve (computed for a high drain bias). The curve is represented by two Tcl lists: one contains the voltage points and one contains the corresponding current values. The drain leakage current is extracted at a small nonzero gate voltage value to avoid noise.

Syntax

```
ext::Extractloff -out <var_name> -v <list_of_r> -i <list_of_r> -vo <r>
    [-log10 0 | 1] [-name <string>] [-f <string>]
    [-info 0 | 1 | 2 | 3] [-help 0 | 1]
```

Argument	Description
<code>-out <var_name></code>	Variable name to store the value of the drain leakage current. (Real number, no default)
<code>-v <list_of_r></code>	List containing the gate voltage values. (List of real numbers, no default)
<code>-i <list_of_r></code>	List containing the drain current values. (List of real numbers, no default)
<code>-vo <r></code>	Gate voltage at which the drain leakage current is extracted. It is recommended to use a small but nonzero value, such as 0.1 mV for an NMOS device and -0.1 mV for a PMOS device. (Real number, no default)
<code>-log10 0 1</code>	Procedure returns $\log_{10}(I_{off})$ if set to 1. Otherwise, the procedure returns I_{off} . Default: 0
<code>-name <string></code>	Name of the extracted variable to appear in the Sentaurus Workbench Family Tree. Note: If <code>-name "noprint"</code> is used, Sentaurus Workbench extraction is suppressed. If <code>-name "out"</code> is used, the name of the variable specified by the <code>-out</code> keyword also is used as the name that appears in the Sentaurus Workbench Family Tree. (String, default: "Ioff")
<code>-f <string></code>	Format string used to write the extracted variable to the Sentaurus Workbench Family Tree. (String, default: "% .3e")
<code>-info 0 1 2 3</code>	Sets local information level. Default: 0

Appendix F: Extraction Library

ext::ExtractIoff

Argument	Description
-help 0 1	Prints a help screen if set to 1. Default: 0

Returns

None.

Example

```
load_file DATA/IdVgSat_pMOS_des.plt -name DC
set Vgs [get_variable_data "gate OuterVoltage" -dataset DC]
set Ids [get_variable_data "drain TotalCurrent" -dataset DC]
ext::AbsList -out absIds -x $Ids

# Compute absolute value of drain currents
ext::ExtractIoff -out Ioff -name "out" -v $Vgs -i $absIds -vo -le-4 \
    -log10 0
puts "Ioff: [format %.3e $Ioff] A/um"

ext::ExtractIoff -out log10Ioff -name "noprint" -v $Vgs -i $absIds \
    -vo -le-4 -log10 1
puts "Log10Ioff: [format %.3f $log10Ioff]"

#-> Ioff: 4.009e-06 A/um
#-> Log10Ioff: -5.397
```

Appendix F: Extraction Library

ext::ExtractRdiff

ext::ExtractRdiff

Extracts the differential resistance R_{diff} from an I–V curve at a specified voltage. R_{diff} is defined as:

$$R_{\text{diff}} = \frac{dV}{dI} \quad (2)$$

The curve is represented by two Tcl lists: one contains the voltage points and one contains the corresponding current values.

Syntax

```
ext::ExtractRdiff -out <var_name> -v <list_of_r> -i <list_of_r>
    -vo <r>
    [-yLog 0 | 1] [-xLog 0 | 1] [-name <string>] [-f <string>]
    [-info 0 | 1 | 2 | 3] [-help 0 | 1]
```

Argument	Description
-out <var_name>	Variable name to store the value of the differential resistance. (Real number, no default)
-v <list_of_r>	List containing the voltage values. (List of real numbers, no default)
-i <list_of_r>	List containing the current values. (List of real numbers, no default)
-vo <r>	Voltage at which the differential resistance is extracted. (Real number, no default)
-yLog 0 1	Selects linear (0) or logarithmic (1) interpolation for y-axis values for computing the derivative. See note in ext::DiffList on page 423 . Default: 0
-xLog 0 1	Selects linear (0) or logarithmic (1) interpolation for x-axis values for computing the derivative. See note in ext::DiffList on page 423 . Default: 0

Appendix F: Extraction Library

ext::ExtractRdiff

Argument	Description
-name <string>	Name of the extracted variable to appear in the Sentaurus Workbench Family Tree. Note: If -name "noprint" is used, Sentaurus Workbench extraction is suppressed. If -name "out" is used, the name of the variable specified by the -out keyword also is used as the name that appears in the Sentaurus Workbench Family Tree.
	(String, default: "Rdiff")
-f <string>	Format string used to write the extracted variable to the Sentaurus Workbench Family Tree. (String, default: "% .3f")
-info 0 1 2 3	Sets local information level. Default: 0
-help 0 1	Prints a help screen if set to 1. Default: 0

Returns

None.

Example

```
# Extract on-state output resistance of a p-n-p bipolar transistor
load_file DATA/IcVc_pnp_des.plt -name IcVce
set Vcs [get_variable_data "collector OuterVoltage" -dataset IcVce]
set Ics [get_variable_data "collector TotalCurrent" -dataset IcVce]
ext::AbsList -out absIcs -x $Ics

# Compute absolute value of collector current
ext::ExtractRdiff -out Ron -name "out" -v $Vcs -i $absIcs -vo -1.25 \
-f "% .2e"

puts "Ron (at Vcs= -1.25 V): [format %.2e $Ron] Ohm-um"
#-> Ron: 3.35e+04 Ohm-um
```

Appendix F: Extraction Library

ext::ExtractRsh

ext::ExtractRsh

Calculates the sheet resistance $R_{sh}[\Omega/\text{sq}]$ and the p-n junction depth of semiconductor layers in the vertical direction in a 2D structure by creating an axis-aligned cutline. It also calculates the total sheet resistance (the sum of the sheet resistance of each layer).

Note:

This procedure applies only to 2D structures.

The sheet resistance of each semiconductor layer is computed using:

$$R_{sh} = \frac{1}{d} \int_0^d \sigma(x) dx \quad (3)$$

where d is the thickness of the semiconductor layer, and σ is its conductivity given by:

$$\sigma(x) = q[n(x)\mu_n(x) + p(x)\mu_p(x)] \quad (4)$$

where:

- q is the elementary charge.
- n and p are the electron and hole density, respectively.
- μ_n and μ_p are the electron and hole mobility, respectively.

The resistivity ρ is given by:

$$\rho(x) = \frac{1}{\sigma(x)} \quad (5)$$

Note:

The axis-aligned cutline is created using the `create_cutline` command (see [create_cutline on page 224](#)).

Syntax

```
ext::ExtractRsh -out <array_name> -dataset <dataName>
                 -semdataset <dataName>
                 -type <x | y> -at <r>
                 [-info 0 | 1 | 2 | 3] [-help 0 | 1]
```

Appendix F: Extraction Library

ext::ExtractRsh

Argument	Description
-out <array_name>	Name of an array to store the results. The array has one string-valued index. The index contains the elements <code>d</code> , <code>Rsh</code> , <code>RshTotal</code> , <code>RshTop</code> , <code>xjTop</code> , and other elements. The values of the elements <code>d</code> and <code>Rsh</code> are the thickness and sheet resistance of each semiconductor layer. The values of the elements <code>RshTotal</code> , <code>RshTop</code> , and <code>xjTop</code> are the total sheet resistance, the sheet resistance of the top semiconductor layer, and the junction depth, respectively. The index also contains the elements <code>x</code> (for <code>-type y</code>) or <code>y</code> (for <code>-type x</code>), <code>DopingConcentration</code> , <code>eMobility</code> , <code>hMobility</code> , <code>eDensity</code> , <code>hDensity</code> , <code>Conductivity</code> , and <code>Resistivity</code> . The values of the elements <code>x</code> and <code>y</code> are lists of x-axis values and y-axis values, respectively. The values of the elements <code>DopingConcentration</code> , <code>eMobility</code> , <code>hMobility</code> , <code>eDensity</code> , <code>hDensity</code> , <code>Conductivity</code> , and <code>Resistivity</code> are lists of the doping concentration, electron mobility, hole mobility, electron density, hole density, conductivity, and resistivity, respectively. (Array name, no default)
-dataset <dataname>	Name of the dataset from where the cutline will be generated. (String, no default)
-semdataset <dataname>	Name of the dataset containing the variables <code>x</code> (for <code>-type y</code>) or <code>y</code> (for <code>-type x</code>), <code>DopingConcentration</code> , <code>eMobility</code> , <code>hMobility</code> , <code>eDensity</code> , <code>hDensity</code> , <code>Conductivity</code> , and <code>Resistivity</code> . These variables contain a list of x-axis values (for <code>-type y</code>), or y-axis values (for <code>-type x</code>), doping concentration, electron mobility, hole mobility, electron density, hole density, conductivity, and resistivity, respectively. (String, no default)
-type x y	Links the cutline to the specified axis. If <code>-type x</code> (<code>-type y</code>) is specified, the cutline is linked to the x-axis (y-axis), and the cutline is created axis-aligned to the y-axis (x-axis). Specify <code>-type</code> so that the cutline is created in the vertical direction. (String, no default)
-at <r>	Value where the axis-aligned cutline cuts the axis to which it is linked. (Real number, no default)
-info 0 1 2 3	Sets local information level. Default: 0
-help 0 1	Prints a help screen if set to 1. Default: 0

Appendix F: Extraction Library

ext::ExtractSS

Returns

None.

Example

```
load_file DATA/LDMOS_des.tdr -name Structure2d
create_plot -name Plot_Structure2d -dataset Structure2d
# Create cutline at y=5.0 and extract Rsh
ext::ExtractRsh -out Rsh -dataset Structure2d -semdataset RshProfile \
    -type y -at 5.0

# Extract sheet resistance of top layer
puts "DOE: Rshtop [format %.2e $Rsh(RshTop)]"

# Extract p-n junction depth
puts "DOE: xj [format %.3f $Rsh(xjTop)]"

puts "Rshtop: [format %.2e $Rsh(RshTop)] Ohm/sq"
puts "xj: [format %.3f $Rsh(xjTop)] um"
#-> Rshtop: 7.36e+04 Ohm/sq
#-> xj: 2.029 um

# Plot conductivity profile
create_plot -1d -name Plot_Profile
create_curve -dataset RshProfile -axisX X -axisY Conductivity
```

ext::ExtractSS

Extracts the subthreshold voltage swing, for a given gate voltage V_{go} , from an I_d - V_{gs} curve. The subthreshold voltage swing (SS) is defined as:

$$SS = \frac{1000}{\frac{d}{dV_g} \log_{10} I_d} \quad (6)$$

where V_g is given in V, I_d is given in A/ μ m or A, and SS is given in mV/decade. The curve is represented by two Tcl lists: one contains the voltage points and one contains the corresponding current values.

Note:

The slope might be *noisy* at the beginning of the curve or at very low current levels. Better results are often obtained when setting V_{go} to a small but nonzero value.

Syntax

```
ext::ExtractSS -out <var_name> -v <list_of_r> -i <list_of_r> -vo <r>
    [-name <string>] [-f <string>] [-info 0 | 1 | 2 | 3] [-help 0 | 1]
```

Appendix F: Extraction Library

ext::ExtractSS

Argument	Description
-out <var_name>	Variable name to store the value of the subthreshold voltage swing. (Real number, no default)
-v <list_of_r>	List containing the gate voltage values. (List of real numbers, no default)
-i <list_of_r>	List containing the absolute value of the drain currents. (List of real numbers, no default)
-vo <r>	Gate voltage at which the slope is extracted. It should be a value well below the threshold voltage. (Real number, no default)
-name <string>	Name of the extracted variable to appear in the Sentaurus Workbench Family Tree.
Note:	
	If -name "noprint" is used, Sentaurus Workbench extraction is suppressed.
	If -name "out" is used, the name of the variable specified by the -out keyword also is used as the name that appears in the Sentaurus Workbench Family Tree.
	(String, default: "ss")
-f <string>	Format string used to write the extracted variable to the Sentaurus Workbench Family Tree. (String, default: "% .3f")
-info 0 1 2 3	Sets local information level. Default: 0
-help 0 1	Prints a help screen if set to 1. Default: 0

Returns

None.

Example

```
load_file DATA/IdVgLin_nMOS_des.plt -name DC
set Vgs [get_variable_data "gate OuterVoltage" -dataset DC]
set Ids [get_variable_data "drain TotalCurrent" -dataset DC]

set Vgo 1e-2
ext::ExtractSS -out SSlin -name "out" -v $Vgs -i $Ids -vo $Vgo
```

Appendix F: Extraction Library

ext::ExtractSsub

ext::ExtractSsub

Extracts the subthreshold voltage swing from an I_d - V_{gs} curve. The subthreshold voltage swing (SS) is defined as:

$$SS = \frac{1000}{Max\left(\frac{d}{dV_g} \log_{10} I_d\right)} \quad (7)$$

where V_g is given in V, I_d is given in A/ μ m or A, SS is given in mV/decade, and $Max\left(\frac{d}{dV_g} \log_{10} I_d\right)$ is the maxima of $\frac{d}{dV_g} \log_{10} I_d$.

The curve is represented by two Tcl lists: one contains the voltage points and one contains the corresponding current values.

Syntax

```
ext::ExtractSsub -out <var_name> -v <list_of_r> -i <list_of_r>
    [-name <string>] [-f <string>] [-info 0 | 1 | 2 | 3] [-help 0 | 1]
```

Argument	Description
-out <var_name>	Variable name to store the value of the subthreshold voltage swing. (Real number, no default)
-v <list_of_r>	List containing the gate voltage values. (List of real numbers, no default)
-i <list_of_r>	List containing the absolute value of the drain currents. (List of real numbers, no default)
-name <string>	Name of the extracted variable to appear in the Sentaurus Workbench Family Tree.
Note:	
	If -name "noprint" is used, Sentaurus Workbench extraction is suppressed.
	If -name "out" is used, the name of the variable specified by the -out keyword also is used as the name that appears in the Sentaurus Workbench Family Tree.
	(String, default: "ss")
-f <string>	Format string used to write the extracted variable to the Sentaurus Workbench Family Tree. (String, default: "% .3f")
-info 0 1 2 3	Sets local information level. Default: 0

Appendix F: Extraction Library

ext::ExtractValue

Argument	Description
-help 0 1	Prints a help screen if set to 1. Default: 0

Returns

None.

Example

```
load_file DATA/IdVgLin_nMOS_des.plt -name DC
set Vgs [get_variable_data "gate OuterVoltage" -dataset DC]
set Ids [get_variable_data "drain TotalCurrent" -dataset DC]

ext::ExtractSsub -out Ssub -name "out" -v $Vgs -i $Ids
```

ext::ExtractValue

For a given target x-value, the procedure extracts the n -th interpolated y-value in a curve. The curve is represented by two Tcl lists: one contains the x-values and one contains the corresponding y-values.

Note:

To find the interpolated x-value for a given y-value, swap the arguments `x` and `y`.

Syntax

```
ext::ExtractValue -out <var_name> -x <list_of_r> -y <list_of_r> -xo <r>
[-occurrence <i>] [-yLog 0 | 1] [-xLog 0 | 1] [-name <string>]
[-f <string>] [-info 0 | 1 | 2 | 3] [-help 0 | 1]
```

Argument	Description
-out <var_name>	Variable name to store the first found y-value. (Real number, no default)
-x <list_of_r>	List containing the x-values. (List of real numbers, no default)
-y <list_of_r>	List containing the y-values. (List of real numbers, no default)
-xo <r>	Target x-value. (Real number, no default)
-occurrence <i>	Specifies the n -th interpolated y-value to be extracted. (Integer, default: 1)

Appendix F: Extraction Library

ext::ExtractValue

Argument	Description
-yLog 0 1	Selects linear (0) or logarithmic (1) interpolation for y-axis values. Default: 0
-xLog 0 1	Selects linear (0) or logarithmic (1) interpolation for x-axis values. Default: 0
-name <string>	Name of the extracted variable to appear in the Sentaurus Workbench Family Tree. Note: If -name "noprint" is used, Sentaurus Workbench extraction is suppressed. If -name "out" is used, the name of the variable specified by the -out keyword also is used as the name that appears in the Sentaurus Workbench Family Tree.
	(String, default: "ss")
-f <string>	Format string used to write the extracted variable to the Sentaurus Workbench Family Tree. (String, default: "% .3f")
-info 0 1 2 3	Sets local information level. Default: 0
-help 0 1	Prints a help screen if set to 1. Default: 0

Returns

None.

Example

```
set Xs [list 1e2 1e3 1e4 1e5 1e6 1e7 1e8 1e9 1e10 1e11 1e12]
set Ys [list -100 -25 25 50 100 100 75 50 25 -25 -100]

# Extract the first interpolated x-value for the given target y-value
# using the default linear interpolation
ext::ExtractValue -out Xlin -x $Ys -y $Xs -yLog 0 -xo 0
puts "Xlin: [format %.3e $Xlin]"

# Extract the first interpolated x-value for the given target y-value
# using logarithmic interpolation
ext::ExtractValue -out Xlog -x $Ys -y $Xs -yLog 1 -xo 0
puts "Xlog: [format %.3e $Xlog]"

# Extract the 2D interpolated x-value for the given target y-value
# using logarithmic interpolation
ext::ExtractValue -out X2nd_log -x $Ys -y $Xs -yLog 1 -xo 0 \
```

Appendix F: Extraction Library

ext::ExtractVdlin

```
-occurrence 2  
puts "X2nd_log: [format %.3e $X2nd_log]"
```

ext::ExtractVdlin

Extracts the drain voltage for a given drain current level from an I_d - V_{ds} curve using linear interpolation. The curve is represented by two Tcl lists: one contains the voltage points and one contains the corresponding current values.

Syntax

```
ext::ExtractVdlin -out <var_name> -v <list_of_r> -i <list_of_r> -io <r>  
[-name <string>] [-f <string>] [-info 0 | 1 | 2 | 3] [-help 0 | 1]
```

Argument	Description
<code>-out <var_name></code>	Variable name to store the value of the drain voltage corresponding to the specified drain current level. (Real number, no default)
<code>-v <list_of_r></code>	List containing the drain voltage values. (List of real numbers, no default)
<code>-i <list_of_r></code>	List containing the drain current values. (List of real numbers, no default)
<code>-io <r></code>	Drain current level. (Real number, no default)
<code>-name <string></code>	Name of the extracted variable to appear in the Sentaurus Workbench Family Tree.
Note:	
	If <code>-name "noprint"</code> is used, Sentaurus Workbench extraction is suppressed.
	If <code>-name "out"</code> is used, the name of the variable specified by the <code>-out</code> keyword also is used as the name that appears in the Sentaurus Workbench Family Tree.
	(String, default: "ss")
<code>-f <string></code>	Format string used to write the extracted variable to the Sentaurus Workbench Family Tree. (String, default: "% .3f")
<code>-info 0 1 2 3</code>	Sets local information level. Default: 0
<code>-help 0 1</code>	Prints a help screen if set to 1. Default: 0

Appendix F: Extraction Library

ext::ExtractVdlog

Returns

None.

Example

```
load_file DATA/IdVd_nMOS_des.plt -name DC
set Vds [get_variable_data "drain OuterVoltage" -dataset DC]
set Ids [get_variable_data "drain TotalCurrent" -dataset DC]

ext::ExtractVdlin -out Vdlin -name "out" -v $Vds -i $Ids -io 1e-4 \
-f "%.4f"
```

ext::ExtractVdlog

Extracts the drain voltage for a given drain current level from an I_d - V_{ds} curve using logarithmic interpolation. The curve is represented by two Tcl lists: one contains the voltage points and one contains the corresponding current values.

Syntax

```
ext::ExtractVdlog -out <var_name> -v <list_of_r> -i <list_of_r> -io <r>
[-name <string>] [-f <string>] [-info 0 | 1 | 2 | 3] [-help 0 | 1]
```

Argument	Description
-out <var_name>	Variable name to store the value of the drain voltage corresponding to the specified drain current level. (Real number, no default)
-v <list_of_r>	List containing the drain voltage values. (List of real numbers, no default)
-i <list_of_r>	List containing the drain current values. (List of real numbers, no default)
-io <r>	Drain current level. (Real number, no default)
-name <string>	Name of the extracted variable to appear in the Sentaurus Workbench Family Tree.

Note:

If -name "noprint" is used, Sentaurus Workbench extraction is suppressed.
If -name "out" is used, the name of the variable specified by the -out keyword also is used as the name that appears in the Sentaurus Workbench Family Tree.

(String, default: "ss")

Appendix F: Extraction Library

ext::ExtractVglin

Argument	Description
-f <string>	Format string used to write the extracted variable to the Sentaurus Workbench Family Tree. (String, default: "% .3f")
-info 0 1 2 3	Sets local information level. Default: 0
-help 0 1	Prints a help screen if set to 1. Default: 0

Returns

None.

Example

```
load_file DATA/IdVd_nMOS_des.plt -name DC
set Vds [get_variable_data "drain OuterVoltage" -dataset DC]
set Ids [get_variable_data "drain TotalCurrent" -dataset DC]

ext::ExtractVdlog -out Vdlog -name "out" -v $Vds -i $Ids -io 1e-4 \
-f "% .4f"
```

ext::ExtractVglin

Extracts the gate voltage for a given drain current level from an I_d – V_{gs} curve using linear interpolation. The curve is represented by two Tcl lists: one contains the voltage points and one contains the corresponding current values.

Syntax

```
ext::ExtractVglin -out <var_name> -v <list_of_r> -i <list_of_r> -io <r>
[-name <string>] [-f <string>] [-info 0 | 1 | 2 | 3] [-help 0 | 1]
```

Argument	Description
-out <var_name>	Variable name to store the value of the drain voltage corresponding to the specified drain current level. (Real number, no default)
-v <list_of_r>	List containing the drain voltage values. (List of real numbers, no default)
-i <list_of_r>	List containing the drain current values. (List of real numbers, no default)
-io <r>	Drain current level. (Real number, no default)

Appendix F: Extraction Library

ext::ExtractVglin

Argument	Description
-name <string>	Name of the extracted variable to appear in the Sentaurus Workbench Family Tree. Note: If -name "noprint" is used, Sentaurus Workbench extraction is suppressed. If -name "out" is used, the name of the variable specified by the -out keyword also is used as the name that appears in the Sentaurus Workbench Family Tree.
	(String, default: "ss")
-f <string>	Format string used to write the extracted variable to the Sentaurus Workbench Family Tree. (String, default: "%.3f")
-info 0 1 2 3	Sets local information level. Default: 0
-help 0 1	Prints a help screen if set to 1. Default: 0

Returns

None.

Example

```
load_file DATA/IdVgLin_nMOS_des.plt -name DC
set Vgs [get_variable_data "gate OuterVoltage" -dataset DC]
set Ids [get_variable_data "drain TotalCurrent" -dataset DC]

ExtractVglin -out Vglin -name "out" -v $Vgs -i $Ids -io 1e-7 -f "%.4f"
```

Appendix F: Extraction Library

ext::ExtractVglog

ext::ExtractVglog

Extracts the gate voltage for a given drain current level from an I_d-V_{gs} curve using logarithmic interpolation. The curve is represented by two Tcl lists: one contains the voltage points and one contains the corresponding current values.

Syntax

```
ext::ExtractVglog -out <var_name> -v <list_of_r> -i <list_of_r> -io <r>
[-name <string>] [-f <string>] [-info 0 | 1 | 2 | 3] [-help 0 | 1]
```

Argument	Description
-out <var_name>	Variable name to store the value of the drain voltage corresponding to the specified drain current level. (Real number, no default)
-v <list_of_r>	List containing the drain voltage values. (List of real numbers, no default)
-i <list_of_r>	List containing the drain current values. (List of real numbers, no default)
-io <r>	Drain current level. (Real number, no default)
-name <string>	Name of the extracted variable to appear in the Sentaurus Workbench Family Tree. Note: If -name "noprint" is used, Sentaurus Workbench extraction is suppressed. If -name "out" is used, the name of the variable specified by the -out keyword also is used as the name that appears in the Sentaurus Workbench Family Tree. (String, default: "ss")
-f <string>	Format string used to write the extracted variable to the Sentaurus Workbench Family Tree. (String, default: "% .3f")
-info 0 1 2 3	Sets local information level. Default: 0
-help 0 1	Prints a help screen if set to 1. Default: 0

Returns

None.

Appendix F: Extraction Library

ext::ExtractVtgm

Example

```
load_file DATA/IdVgLin_nMOS_des.plt -name DC
set Vgs [get_variable_data "gate OuterVoltage" -dataset DC]
set Ids [get_variable_data "drain TotalCurrent" -dataset DC]

ExtractVglog -out Vglog -name "out" -v $Vgs -i $Ids -io 1e-7 -f "%.4f"
```

ext::ExtractVtgm

Extracts the threshold voltage from an I_d - V_{gs} curve using the maximum transconductance method. The threshold voltage is defined as the gate-voltage axis intercept of the tangent line at the maximum transconductance g_m point. The gate bias at which the maximum transconductance occurs is computed using parabolic interpolation. The curve is represented by two Tcl lists: one contains the voltage points and one contains the corresponding current values.

Syntax

```
ext::ExtractVtgm -out <var_name> -v <list_of_r> -i <list_of_r>
                  [-name <string>] [-f <string>] [-info 0 | 1 | 2 | 3] [-help 0 | 1]
```

Argument	Description
<code>-out <var_name></code>	Variable name to store the value of the drain voltage corresponding to the specified drain current level. (Real number, no default)
<code>-v <list_of_r></code>	List containing the drain voltage values. (List of real numbers, no default)
<code>-i <list_of_r></code>	List containing the drain current values. (List of real numbers, no default)
<code>-name <string></code>	Name of the extracted variable to appear in the Sentaurus Workbench Family Tree.
Note:	If <code>-name "noprint"</code> is used, Sentaurus Workbench extraction is suppressed. If <code>-name "out"</code> is used, the name of the variable specified by the <code>-out</code> keyword also is used as the name that appears in the Sentaurus Workbench Family Tree.
	(String, default: "ss")
<code>-f <string></code>	Format string used to write the extracted variable to the Sentaurus Workbench Family Tree. (String, default: "% .3f")

Appendix F: Extraction Library

ext::ExtractVti

Argument	Description
-info 0 1 2 3	Sets local information level. Default: 0
-help 0 1	Prints a help screen if set to 1. Default: 0

Returns

None.

Example

```
# Extract Vtgm for a p-MOSFET
load_file DATA/IdVgLin_pMOS_des.plt -name DC
set Vgs [get_variable_data "gate OuterVoltage" -dataset DC]
set Ids [get_variable_data "drain TotalCurrent" -dataset DC]
ext::AbsList -out absIds -x $Ids

ext::ExtractVtgm -out Vtgm -name "out" -v $Vgs -i $absIds

puts "Vt (Max gm method): [format %.3f $Vtgm] V"
#-> Vt (Max gm method): -0.503 V
```

ext::ExtractVti

Extracts the threshold voltage for a given subthreshold current level from an I_d - V_{gs} curve. The threshold voltage is defined as the gate voltage at which the drain current reaches the current level. The curve is represented by two Tcl lists: one contains the voltage points and one contains the corresponding current values.

Syntax

```
ext::ExtractVti -out <var_name> -v <list_of_r> -i <list_of_r> -io <r>
[-name <string>] [-f <string>] [-info 0 | 1 | 2 | 3] [-help 0 | 1]
```

Argument	Description
-out <var_name>	Variable name to store the value of the drain voltage corresponding to the specified drain current level. (Real number, no default)
-v <list_of_r>	List containing the drain voltage values. (List of real numbers, no default)
-i <list_of_r>	List containing the drain current values. (List of real numbers, no default)

Appendix F: Extraction Library

ext::ExtractVti

Argument	Description
-io <r>	Subthreshold current level. (Real number, no default)
-name <string>	Name of the extracted variable to appear in the Sentaurus Workbench Family Tree.
Note:	If -name "noprint" is used, Sentaurus Workbench extraction is suppressed. If -name "out" is used, the name of the variable specified by the -out keyword also is used as the name that appears in the Sentaurus Workbench Family Tree.
	(String, default: "ss")
-f <string>	Format string used to write the extracted variable to the Sentaurus Workbench Family Tree. (String, default: "% .3f")
-info 0 1 2 3	Sets local information level. Default: 0
-help 0 1	Prints a help screen if set to 1. Default: 0

Returns

None.

Example

```
load_file DATA/IdVgLin_nMOS_des.plt -name DC
set Vgs [get_variable_data "gate OuterVoltage" -dataset DC]
set Ids [get_variable_data "drain TotalCurrent" -dataset DC]

set lgate 0.022          ;# um
set io [expr 100e-9/$lgate]    ;# subthreshold current level [A/um]
ext::ExtractVti -out VtiLin -name "out" -v $Vgs -i $Ids -io $io
```

Appendix F: Extraction Library

ext::ExtractVtsat

ext::ExtractVtsat

Extracts the threshold voltage from a $\sqrt{I_d} - V_{gs}$ curve. The threshold voltage is defined as the intercept with the gate-voltage axis from the point of maximum slope of the $\sqrt{I_d} - V_{gs}$ curve. The gate bias at which the maximum slope of the $\sqrt{I_d} - V_{gs}$ curve occurs is computed using parabolic interpolation. The curve is represented by two Tcl lists: one contains the voltage points and one contains the corresponding current values.

Syntax

```
ext::ExtractVtsat -out <var_name> -v <list_of_r> -i <list_of_r>
    [-name <string>] [-f <string>] [-info 0 | 1 | 2 | 3] [-help 0 | 1]
```

Argument	Description
-out <var_name>	Variable name to store the value of the drain voltage corresponding to the specified drain current level. (Real number, no default)
-v <list_of_r>	List containing the drain voltage values. (List of real numbers, no default)
-i <list_of_r>	List containing the drain current values. (List of real numbers, no default)
-name <string>	Name of the extracted variable to appear in the Sentaurus Workbench Family Tree. Note: If -name "noprint" is used, Sentaurus Workbench extraction is suppressed. If -name "out" is used, the name of the variable specified by the -out keyword also is used as the name that appears in the Sentaurus Workbench Family Tree. (String, default: "ss")
-f <string>	Format string used to write the extracted variable to the Sentaurus Workbench Family Tree. (String, default: "% .3f")
-info 0 1 2 3	Sets local information level. Default: 0
-help 0 1	Prints a help screen if set to 1. Default: 0

Appendix F: Extraction Library

ext::FilterTable

Returns

None.

Example

```
load_file DATA/IdVgLin_nMOS_des.plt -name DC
set Vgs [get_variable_data "gate OuterVoltage" -dataset DC]
set Ids [get_variable_data "drain TotalCurrent" -dataset DC]

ext::ExtractVtsat -out VtLin -name "out" -v $Vgs -i $Ids
```

ext::FilterTable

Processes data from the Sentaurus Workbench Family Tree for the purpose of creating a graph of one Sentaurus Workbench parameter (y-values) as a function of another Sentaurus Workbench parameter (x-values) for a certain subset of experiments. The data is specified in the form of two lists identifying the x- and y-values, which are preprocessed to create a graph. The condition that an experiment must fulfill to be included in the graph is specified using a pair of target values and a corresponding list of Sentaurus Workbench parameters.

Syntax

```
ext::FilterTable -out <array_name> -x <list_of_r> -y <list_of_r>
  -conditions <array_name> -ncond <i>
  [-info 0 | 1 | 2 | 3] [-help 0 | 1]
```

Argument	Description
-out <array_name>	Name of an array to store the results. The array has one string-valued index. The index contains the elements X and Y. The values of the X element are a subset of a list of values, specified using the keyword -x. These values are in ascending order. The values of the Y element are a subset of a list of values, specified using the keyword -y. All entries of the 'y'-list that contain a nonnumeric value are ignored. (Array name, no default)
-x <list_of_r>	List containing the values of a Sentaurus Workbench parameter to be preprocessed: the x-values. (List of real numbers, no default)
-y <list_of_r>	List containing the values of a Sentaurus Workbench parameter to be preprocessed: the y-values. (List of real numbers, no default)

Appendix F: Extraction Library

ext::FilterTable

Argument	Description
-conditions <array_name>	Array with two indices. The string-indexed array contains the elements "Target" and "Values". The value of the "Target" element contains the required value of a Sentaurus Workbench parameter to be used as a filter condition. The "Values" element contains the corresponding value list of the Sentaurus Workbench parameter for all the experiments.
	The second integer counter enumerates the conditions. The enumerations start with 1. (Array name, no default)
-ncond <i>	Number of conditions contained in the array specified using the keyword -conditions. (Integer, no default)
-info 0 1 2 3	Sets local information level. Default: 0
-help 0 1	Prints a help screen if set to 1. Default: 0

Returns

None.

Example

```
# Plot Vt roll-off curve for PMOS under stress
set Types [list nMOS nMOS nMOS nMOS nMOS nMOS nMOS nMOS \
           pMOS pMOS pMOS pMOS pMOS pMOS pMOS]
set Lgs [list 0.090 0.090 0.045 0.045 0.130 0.130 0.065 0.065 \
          0.065 0.065 0.045 0.045 0.130 0.130 0.090 0.090]
set Stress [list no yes no yes no yes no yes \
            no yes no yes no yes no yes]
set Vtgms [list 0.424 0.0374 0.313 0.263 0.414 0.364 0.408 0.358 \
           -0.344 -0.294 -0.232 -0.182 x x -0.374 -0.324]
set Conditions(Target,1) "pMOS"
set Conditions(Values,1) $Types
set Conditions(Target,2) "yes"
set Conditions(Values,2) $Stress
ext::FilterTable -out LgVt -x $Lgs -y $Vtgms -conditions Conditions
-ncond 2
create_variable -name Lg -dataset Vtgmlg -values $LgVt(X)
create_variable -name Vtgm -dataset Vtgmlg -values $LgVt(Y)
create_plot -1d -name Plot_VtRollOff
create_curve -name VtRollOff -dataset Vtgmlg -axisX "Lg" -axisY "Vtgm"
puts "Lg= $LgVt(X)"
puts "Vtgm= $LgVt(Y)"
#-> Lg= 0.045 0.065 0.090
#-> Vtgm= -0.182 -0.294 -0.324
```

Appendix F: Extraction Library

ext::FilterTable

Usage Under Sentaurus Workbench

In a Sentaurus Visual script, you can use the dynamic preprocessing feature of Sentaurus Workbench @<parameter_name>:all@ to access a list of input parameters and extracted values for all Sentaurus Workbench experiments. For example, the lists `Types`, `Lgs`, `Stress`, and `Vtgms` in the above example are generated automatically as a result of the following commands in the Sentaurus Visual script:

```
set Types [list @Type:all@]
set Lgs [list @lgate:all@]
set Stress [list @stress:all@]
set Vtgms [list @Vt:all@]
```

Here, the Tcl list `Types` contains, for all experiments, the values of the Sentaurus Workbench input parameter `Type`, which for example takes on the values `nMOS` or `pMOS`, depending on whether in this experiment an NMOS or a PMOS structure is created.

Similarly, the Tcl list `Lgs` contains, for all experiments, a ‘parallel’ list of values of another Sentaurus Workbench input parameter, which for example contains the value of the gate length of the given MOSFETs.

The corresponding extracted parameter can be accessed in the same way. For example, the Tcl list `Vtgms` contains the extracted values for the threshold voltage for each respective experiment.

Note:

The values in the various lists might or might not be numeric, and the values might not necessarily be ordered.

The lists of x- and y-values, which will be processed (filtered) to create the graph, are specified using the keywords `-x` and `-y` in the procedure `ext::FilterTable`. In the above example, the lists of gate lengths (`-x $Lgs`) and `Vtgm` values (`-y $Vtgms`) are processed by `ext::FilterTable`.

The keyword `-conditions` controls the conditions an experiment must fulfill to be included in the graph. The total number of conditions is specified by the keyword `-ncond`. All the conditions are specified in a string-indexed array using the keyword `-conditions`. Each condition is defined by both a target value and a corresponding list of Sentaurus Workbench parameters. The target value is the required value of the parameter to be used as a filter condition.

Each element of the string-indexed array has two indices. The first index is either “Target” or “Values”. The second index is the condition number. For each condition number:

- The target value is specified using the “Target” element (element with first index named “Target”) of the array.
- The corresponding list of Sentaurus Workbench parameters is specified using the “Values” element (element with first index named “Values”) of the array.

Appendix F: Extraction Library

ext::FilterTable

In the above example, the following code filters out the gate length (L_g) and threshold voltage (V_{tgm}) values for PMOS devices (condition number 1). This condition is defined using the array named Conditions:

```
set Conditions(Target,1) "pMOS"
set Conditions(Values,1) $Types
```

Here, the target value is "pMOS" and the corresponding list of Sentaurus Workbench parameters is Types.

To filter out L_g and V_{tgm} values for devices under stress (condition number 2), the following additional elements of the Conditions array are defined:

```
set Conditions(Target,2) "yes"
set Conditions(Values,2) $Stress
```

As a result of specifying both the conditions (-conditions Conditions -ncond 2), the procedure ext::FilterTable filters out L_g and V_{tgm} values for PMOS devices under stress.

In the above example, if both the conditions are defined in the Conditions array but the number of conditions is set to 1 (-conditions Conditions -ncond 1), the procedure filters out the gate length and V_{tgm} values for all the PMOS devices (with and without stress). The second condition will not be taken into account.

The procedure returns an array (specified by the keyword -out) with a one string-valued index. The index contains the elements X and Y. The values of the X element are a subset of a list of values, specified using the keyword -x. These values are in ascending order. The values of the Y element are a subset of a list of values, specified using the keyword -y. These lists in the array can be used to create a graph.

In the above example, the procedure returns the array LgVt (-out LgVt) consisting of a list of L_g values and a list of V_{tgm} values for PMOS devices under stress. These lists can be used directly to create the V_t roll-off curve:

```
create_variable -name Lg -dataset VtgmLg -values $LgVt(X)
create_variable -name Vtgm -dataset VtgmLg -values $LgVt(Y)
create_plot -1d -name Plot_VtRolloff
create_curve -name VtRolloff -dataset VtgmLg -axisX "Lg" -axisY "Vtgm"
```

As an additional feature, the ext::FilterTable procedure ignores all entries of the y-values that contain a nonnumeric value. Use this feature to omit failed extractions. In the tool input file that performs the extraction, for example, a previous Sentaurus Visual tool instance, use the #set directive to preset the extracted variable to the value x:

```
#set Vtgm x
...
ext::ExtractVtgm -out Vtgm -name "out" -v $Vgs -i $absIds
```

Appendix F: Extraction Library

ext::FindExtrema

The actual extraction process, here using the `ext::ExtractVtgm` procedure, overwrites the preset value `x` with the actual value. However, if the extraction process fails, the preset value persists.

The output of the above example shows that the `Vtgm` value (= `x`) for the 130 nm gate length (`Lg=0.130`) PMOS device under stress is not included in the array `LgVt`. In addition, the gate lengths in the array `LgVt` are in ascending order:

```
puts "Lg= $LgVt(X)"
puts "Vtgm= $LgVt(Y)"
#-> Lg= 0.045 0.065 0.090
#-> Vtgm= -0.182 -0.294 -0.324
```

ext::FindExtrema

Computes all local extrema (either maxima or minima) of a curve. The curve is represented by two Tcl lists, one containing the x-values (independent variable) and one containing the corresponding y-values (dependent variable).

Note:

If a curve exhibits a flat top or bottom (two or more neighboring x-values have the same y-value), then the last x-value is returned as the extrema point.

Syntax

```
ext::FindExtrema -out <array_name> -x <list_of_r> -y <list_of_r>
[-type "max" | "min"] [-eps <r>]
[-info 0 | 1 | 2 | 3] [-help 0 | 1]
```

Argument	Description
<code>-out <array_name></code>	Name of an array to store the results. The array has one string-valued index. The index contains the elements X and Y. The values of the X element are the x-values corresponding to all the extrema. The values of the Y element are the extrema. (Array name, no default)
<code>-x <list_of_r></code>	List containing the x-values. These must be in either ascending or descending order. (List of real numbers, no default)
<code>-y <list_of_r></code>	List containing the y-values. (List of real numbers, no default)
<code>-type "max" "min"</code>	Selects whether to extract the maxima ("max") or minima ("min") of a curve. Default: "max"
<code>-eps <r></code>	If the difference between two adjacent elements of the list specified using the keyword <code>-y</code> is less than the value of <code>-eps</code> , both elements are considered to be equal. (Real number, default: 1×10^{-10})

Appendix F: Extraction Library

ext::FindVals

Argument	Description
-info 0 1 2 3	Sets local information level. Default: 0
-help 0 1	Prints a help screen if set to 1. Default: 0

Returns

None.

Example

```
set Xs [list 0.0 1.5 2.0 3.0]
set Ys [list 0.0 1.0 0.5 2.0]

ext::FindExtrema -out XY -x $Xs -y $Ys

puts "All maxima: $XY(Y)"
puts "All x-values corresponding to the maxima: $XY(X)"

#-> All maxima: 1.0 2.0
#-> All x values corresponding to the maxima: 1.5 3.0
```

ext::FindVals

For a given target x-value, this procedure extracts all of the corresponding interpolated y-values in a curve. The curve is represented by two Tcl lists: one contains the x-values and one contains the corresponding y-values.

Note:

To find the interpolated x-values for a given y-value, swap the value of the keywords `-x` and `-y`.

Syntax

```
ext::FindVals -out <list_name> -x <list_of_x> -y <list_of_y> -xo <r>
[-yLog 0 | 1] [-xLog 0 | 1] [-info 0 | 1 | 2 | 3] [-help 0 | 1]
```

Argument	Description
-out <list_name>	Name of a list to store the y-values. (List name, no default)
-x <list_of_x>	List containing the x-values. (List of real numbers, no default)
-y <list_of_y>	List containing the y-values. (List of real numbers, no default)

Appendix F: Extraction Library

ext::FindVals

Argument	Description
-xo <r>	Target x-value. (Real number, no default)
-yLog 0 1	Selects linear (0) or logarithmic (1) interpolation for y-axis values. Default: 0
-xLog 0 1	Selects linear (0) or logarithmic (1) interpolation for x-axis values. Default: 0
-info 0 1 2 3	Sets local information level. Default: 0
-help 0 1	Prints a help screen if set to 1. Default: 0

Returns

None.

Example

```
set Xs [list 0.0 1.0 2.0 3.0 4.0]
set Ys [list 0.0 2.0 4.0 2.0 0.0]

# Find all the elements of Xs corresponding to Ys= 2.0
ext::FindVals -out xos -x $Ys -y $Xs -xo 2.0
puts "The elements of Xs corresponding to Ys= 2.0 are $xos"

# Find the first element of Xs corresponding to Ys= 2.0
ext::ExtractValue -out xos -name "noprint" -x $Ys -y $Xs -xo 2.0
puts "The first element of Xs corresponding to Ys= 2.0 is $xos"
#-> The elements of Xs corresponding to Ys= 2.0 are 1.0 3.0
#-> The first element of Xs corresponding to Ys= 2.0 is 1.0
```

Appendix F: Extraction Library

ext::LinFit

ext::LinFit

Performs a linear fit $y = x \bullet m + b$ to a curve using least-squares regression. The curve is represented by two Tcl lists, one containing the x-values (independent variable) and one containing the corresponding y-values (dependent variable).

Syntax

```
ext::LinFit -out <array_name> -x <list_of_r> -y <list_of_r>
             -xmin <r> -xmax <r> [-npar 1 | 2] [-weighted "off" | "on"]
             [-weights <list_of_r>] [-info 0 | 1 | 2 | 3] [-help 0 | 1]
```

Argument	Description
-out <array_name>	Name of an array to store the results. The array has one string-valued index. The index contains the elements X, Yestimate, residuals, slope, yintercept, n, dof, and RMSE. The values of the X element are a subset of a list of values, specified using the keyword -x. The values of the Y element are a subset of a list of values, specified using the keyword -y. The values of the Yestimate, residuals, slope, yintercept (for -npar 2), dof, and RMSE elements are the estimated Y values (y_i), the residuals (e_i), the estimated slope (\hat{m}), the estimated y-intercept (\hat{b}), the degrees of freedom (dof), and the root-mean-square error (RMSE), respectively. The value of the n element is the number of elements (n) in the list represented by the X element. (Array name, no default)
-x <list_of_r>	List containing the x-values. These must be in either ascending or descending order. (List of real numbers, no default)
-y <list_of_r>	List containing the y-values. (List of real numbers, no default)
-xmin <r>	Minimum x-value in the range of x-values over which the linear fit is performed. (Real number, no default)
-xmax <r>	Maximum x-value in the range of x-values over which the linear fit is performed. (Real number, no default)
-npar 1 2	Number of computed parameters. If -npar 1 is used, only the slope is computed and the y-intercept is assumed to be 0. Default: 2
-weighted "off" "on"	Selects either unweighted ("off") or weighted ("on") linear regression. Default: "off"
-weights <list_of_r>	List containing the values of the weights for each x-value. (List of real numbers, no default)

Appendix F: Extraction Library

ext::LinFit

Argument	Description
-info 0 1 2 3	Sets local information level. Default: 0
-help 0 1	Prints a help screen if set to 1. Default: 0

Returns

None.

Example

```
set Xs [list 20 60 100 140 180 220 260 300 340 380]
set Ys [list 0.18 0.37 0.35 0.78 0.56 0.75 1.18 1.36 1.17 1.65]

ext::LinFit -out XY -x $Xs -y $Ys -xmin [lindex $Xs 0] \
-xmax [lindex $Xs end]

puts "Estimated slope= $XY(slope)"
puts "Estimated y-intercept= $XY(yintercept)"
puts "Root-MSE= $XY(RMSE)"

#-> Estimated slope= 0.00383
#-> Estimated y-intercept= 0.06924
#-> Root-MSE= 0.159
```

Linear Fitting Using Least-Squares Regression

The regression curve of Y as a function of X is:

$$y = x \bullet m + b \quad (8)$$

where m is the slope and b is the y-intercept.

The residual e_i of the i -th data point (x_i, y_i) is defined as:

$$e_i = y_i - \hat{y}_i \quad (9)$$

where \hat{y}_i is the estimate of the y-value of the i -th data point.

The sum of squares due to error (SSE) is defined as:

$$\text{SSE} = \sum_{i=1}^n e_i^2 = \sum_{i=1}^n (y_i - \hat{y}_i)^2 \quad (10)$$

Here, the number of the data point is n .

The fitted or estimated regression line:

$$\hat{y} = x \bullet \hat{m} + \hat{b} \quad (11)$$

Appendix F: Extraction Library

ext::Linspace

is computed by minimizing SSE. Here, \hat{y} , \hat{m} , and \hat{b} are the estimated y-value, slope, and y-intercept, respectively [1].

The RMSE is defined as:

$$\text{RMSE} = \sqrt{\frac{\text{SSE}}{\text{dof}}} \quad (12)$$

where the degrees of freedom (dof) for n data points is defined as:

$$\text{dof} = n - 2 \quad (13)$$

ext::Linspace

Creates a list of n linearly spaced values between and including two real numbers (`xmin` and `xmax`).

Syntax

```
ext::Linspace -out <list_name> -xmin <r> -xmax <r> -n <i>
[-info 0 | 1 | 2 | 3] [-help 0 | 1]
```

Argument	Description
<code>-out <list_name></code>	Name of a list to store the list of linearly spaced values. (List name, no default)
<code>-xmin <r></code>	Minimum x-value in the range of x-values over which the list is obtained. (Real number, no default)
<code>-xmax <r></code>	Maximum x-value in the range of x-values over which the list is obtained. (Real number, no default)
<code>-n <i></code>	Number of values created, where the value of <code>-n</code> should be a positive integer greater than 1. (Integer, no default)
<code>-info 0 1 2 3</code>	Sets local information level. Default: 0
<code>-help 0 1</code>	Prints a help screen if set to 1. Default: 0

Returns

None.

Appendix F: Extraction Library

ext::LinTransList

Example

```
# Create a list of 11 linearly spaced values between 0 and 1
ext::Linspace -out X -xmin 0 -xmax 1 -n 11

puts "Xs: $X"
#-> Xs: 0 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1
```

ext::LinTransList

Applies a linear transformation to the elements of a list. The elements of the list are replaced by the transformed values given by:

$$X' = X \bullet m + b \quad (14)$$

Syntax

```
ext::LinTransList -out <list_name> -x <list_of_r>
                  [-m <r>] [-b <r>] [-info 0 | 1 | 2 | 3] [-help 0 | 1]
```

Argument	Description
-out <list_name>	Name of a list to store the list of transformed values. (List name, no default)
-x <list_of_r>	Input list. (List of real numbers, no default)
-m <r>	Slope of the linear transformation. (Real number, default: 1.0)
-b <r>	Offset of the linear transformation. (Real number, default: 0.0)
-info 0 1 2 3	Sets local information level. Default: 0
-help 0 1	Prints a help screen if set to 1. Default: 0

Returns

None.

Example

```
load_file DATA/IdVgLin_nMOS_des.plt -name DC
create_plot -1d -name Plot_IdVg
set Vgs [get_variable_data "gate OuterVoltage" -dataset DC]
set Ids [get_variable_data "drain TotalCurrent" -dataset DC]

ext::LinTransList -out VgTrans -x $Vgs -b 0.55
                  ;# Shift Vg values by 0.55 V
```

Appendix F: Extraction Library

ext::Log10List

```
ext::LinTransList -out IdTrans -x $Ids -m 1e6
                   ;# Scale Id values from A/um to mA/mm

# Create the shifted and scaled Id-Vg curve
create_variable -name VgTrans -dataset IdVgTrans -values $VgTrans
create_variable -name IdTrans -dataset IdVgTrans -values $IdTrans
create_plot -id -name Plot_IdVg
create_curve -name IdVgTrans -dataset IdVgTrans \
             -axisX "VgTrans" -axisY "IdTrans"
```

ext::Log10List

Applies the log10 function to the elements of a list. The elements of the list are replaced by the function values.

Syntax

```
ext::Log10List -out <list_name> -x <list_of_r>
                [-info 0 | 1 | 2 | 3] [-help 0 | 1]
```

Argument	Description
-out <list_name>	Name of a list to store the list of values. (List name, no default)
-x <list_of_r>	Input list. (List of real numbers, no default)
-info 0 1 2 3	Sets local information level. Default: 0
-help 0 1	Prints a help screen if set to 1. Default: 0

Returns

None.

Example

```
set Xs [list 10 100 1000]
ext::Log10List -out Ys -x $Xs

puts "log10(Xs): $Ys"
#-> log10(Xs): 1.0 2.0 3.0
```

Appendix F: Extraction Library

ext::RemoveDuplicates

ext::RemoveDuplicates

For a pair of lists x and y , removes duplicate elements of the list x and the corresponding elements of the list y .

Syntax

```
ext::RemoveDuplicates -out <array_name> -x <list_of_r> -y <list_of_r>
    [-eps <r>] [-info 0 | 1 | 2 | 3] [-help 0 | 1]
```

Argument	Description
-out <array_name>	Name of an array to store the results. The array has one string-valued index. The index contains the elements X and Y. The values of the X element are a subset of a list of values, specified using the keyword $-x$. These do not contain duplicate values. The corresponding elements of the list specified using the keyword $-y$ are stored in the Y element. The values of the Y element are a subset of a list of values, specified using the keyword $-y$. (Array name, no default)
$-x$ <list_of_r>	Input list. (List of real numbers, no default)
$-y$ <list_of_r>	Input list. (List of real numbers, no default)
$-eps$ <r>	If the difference between two adjacent elements of the list specified using the keyword $-x$ is less than $-eps$, the first element is removed. (Real number, default: 10^{-40})
$-info$ 0 1 2 3	Sets local information level. Default: 0
$-help$ 0 1	Prints a help screen if set to 1. Default: 0

Returns

None.

Example

```
set x [list 1 1 2 3 1 1 2 2 2 3]
set y [list 10 20 30 40 50 60 70 80 90 100]

ext::RemoveDuplicates -out XY -x $x -y $y

set Xs $XY(X)
set Ys $XY(Y)
puts "Xs: $Xs"
puts "Ys: $Ys"
```

Appendix F: Extraction Library

ext::RemoveZeros

```
#-> Xs: 1 2 3 1 2 3  
#-> Ys: 20 30 40 60 90 100
```

ext::RemoveZeros

For a pair of lists x and y , removes zero elements of the list x and the corresponding elements of the list y .

Syntax

```
ext::RemoveZeros -out <array_name> -x <list_of_r> -y <list_of_r>  
[-iplists "x" | "y" | "xy"] [-eps <r>] [-info 0 | 1 | 2 | 3]  
[-help 0 | 1]
```

Argument	Description
-out <array_name>	Name of an array to store the results. The array has one string-valued index. The index contains the elements X and Y. The values of the X element are a subset of a list of values, specified using the keyword $-x$. These do not contain zero values. The corresponding elements of the list specified using the keyword $-y$ are stored in the Y element. The values of the Y element are a subset of a list of values, specified using the keyword $-y$. (Array name, no default)
$-x$ <list_of_r>	Input list. (List of real numbers, no default)
$-y$ <list_of_r>	Input list. (List of real numbers, no default)
-iplists "x" "y" "xy"	Input list from which zeros are removed. If $-iplists "x"$ is used, the zeros are removed from the list specified using the keyword $-x$. If $-iplists "xy"$ is used, zeros are removed from both lists specified using the keywords $-x$ and $-y$. (String, no default)
$-eps$ <r>	If an element of the list specified using the keyword $-x$ is less than value of $-eps$, it is removed. (Real number, default: 10^{-40})
$-info$ 0 1 2 3	Sets local information level. Default: 0
$-help$ 0 1	Prints a help screen if set to 1. Default: 0

Returns

None.

Appendix F: Extraction Library

ext::SubLists

Example

```
set xs [list 0 1 2 3 0 -1]
set ys [list 10 20 30 40 50 0]

# Remove zeros only from list xs
ext::RemoveZeros -out XY -x $xs -y $ys -iplists "x"
puts "Xs: $XY(X)"
puts "Ys: $XY(Y)"
#-> Xs: 1 2 3 -1
#-> Ys: 20 30 40 0
```

ext::SubLists

Creates a pair of sublists from a pair of lists. One of the lists is the list of x-values. The second list is the list of y-values. The sublist is created using a range of x-values.

Note:

To create a pair of sublists using a range of y-values, swap the keywords `-x` and `-y`, and specify the range of y-values using the keywords `-xmin` and `-xmax`. To create a sublist from a single list, specify the value of the list using both the `-x` and `-y` keywords.

Syntax

```
ext::SubLists -out <array_name> -x <list_of_r> -y <list_of_r>
              -xmin <r> -xmax <r>
              [-info 0 | 1 | 2 | 3] [-help 0 | 1]
```

Argument	Description
<code>-out <array_name></code>	Name of an array to store the results. The array has one string-valued index. The index contains the elements X and Y. The values of the X element are a subset of a list of values, specified using the keyword <code>-x</code> . The values of the Y element are a subset of a list of values, specified using the keyword <code>-y</code> . (Array name, no default)
<code>-x <list_of_r></code>	List containing the x-values. These must be in either ascending or descending order. (List of real numbers, no default)
<code>-y <list_of_r></code>	List containing the y-values. (List of real numbers, no default)
<code>-xmin <r></code>	Minimum x-value in the range of x-values over which the sublist is obtained. (Real number, no default)

Appendix F: Extraction Library

ext::SubLists

Argument	Description
-xmax <r>	Maximum x-value in the range of x-values over which the sublist is obtained. (Real number, no default)
-info 0 1 2 3	Sets local information level. Default: 0
-help 0 1	Prints a help screen if set to 1. Default: 0

Returns

None.

Example

```
set xs [list 1 2 3 4 5 6 7]
set ys [list 10 20 30 40 50 60 70]

ext::SubLists -out XY -x $xs -y $ys -xmin 2 -xmax 5

puts "Xs: $XY(X)"
puts "Ys: $XY(Y)"
#-> Xs: 2 3 4 5
#-> Ys: 20 30 40 50
```

Appendix F: Extraction Library

lib::SetInfoDef

lib::SetInfoDef

Sets the default information level.

Note:

Level 0: Warning, error, or status messages only.

Level 1: Echo results.

Level 2: Show progress and some debug information.

Level 3: Show all debug information.

The local information level also can be set using the `-info` keyword of the procedures in the extraction library.

Syntax

```
lib::SetInfoDef 0 | 1 | 2 | 3
```

Argument	Description
<code><info_level></code>	Sets the default information level. Default: 0

Returns

None.

Example

```
lib::SetInfoDef 2
```

References

- [1] W. H. Press *et al.*, *Numerical Recipes in C: The Art of Scientific Computing*, Cambridge: Cambridge University Press, 2nd ed., 1992.

G

Impedance Field Method Data Postprocessing Library

This appendix provides information about the impedance field method data postprocessing library.

Overview

The impedance field method in Sentaurus Device provides an accurate and efficient way to evaluate the effects of random variability on the electrical behavior of semiconductor devices (see *Sentaurus™ Device User Guide*, Chapter 23).

Within the statistical impedance field method (sIFM), Sentaurus Device generates many randomized realizations of a reference device. For example, 10000 realizations with different randomized doping distributions, different randomized gate oxide thicknesses, different randomized metal grain boundaries, and so on.

For each of these individual randomizations, Sentaurus Device computes, at each bias point, the linear current response of the randomizations with respect to the reference device.

The impedance field method (IFM) data postprocessing library helps to manage and analyze large amounts of linear current response data. For example, the IFM library allows you to conveniently apply standard statistical analysis methods to data such as computing and visualizing the distribution and comparing it to a Gaussian distribution.

The IFM library also supports the construction of the individual electrical characteristics of the randomized devices from the electrical characteristics of the reference device and the linear current response data.

The IFM library is loaded automatically when Sentaurus Visual starts. However, if you have switched off the automatic loading of extension libraries, then you can load the IFM library explicitly with the command:

```
load_library ifm
```

Syntax Conventions

The IFM library uses a unique namespace identifier (`ifm::`) for its procedures. All procedures and variables associated with this library are called with the namespace identifier prepended. For example:

```
ifm::<proc_name>
```

Each procedure has several arguments. The IFM library uses an input parser that accepts arguments of the form:

```
-keyword <value>
```

Note:

All Sentaurus Visual libraries support the standard Sentaurus Visual syntax in which keywords are preceded by a dash. For backward compatibility, all Sentaurus Visual libraries continue to support the `keyword= <value>` syntax as well. For each procedure call, you can use either the `-keyword <value>` syntax or the `keyword= <value>` syntax. However, within any one procedure call, only one type of syntax can be used. Otherwise, an error message will be generated. Only the new syntax is documented. If you want to continue using the `keyword= <value>` syntax, you also can insert space between the keyword and the equal sign, for example, `keyword = <value>`. Omitting the space between the equal sign and the value field will result in a failure if the value is a de-referenced Tcl variable. Use `keyword= $val` (*not* `keyword=$val`).

The parser accepts arguments in any order. For some arguments, default values are predefined. Such arguments can be omitted. If arguments for which no defaults are predefined are omitted, the procedure will exit with an error message. In addition, unrecognized arguments result in an error message.

Some procedures of the IFM library compute large and complex data structures. For such data structures, the standard Tcl method of using the return value of the procedure to pass results back to the calling program is not suitable. Therefore, for some datasets, the IFM library uses a *passing-by-reference* method to exchange information between the procedure and the calling program. Procedure arguments that use the passing-by-reference method are identified with `-keyword <var_name>, <list_name>, or <array_name>`.

The following conventions are used for the syntax of Tcl commands:

- Angle brackets – `<>` – indicate text that must be replaced, but they are *not* part of the syntax. In particular, the following type identifiers are used:
 - `<r>`: Replace with a real number, or a de-referenced Tcl variable that evaluates to a real number. For example: `$val`.

Appendix G: Impedance Field Method Data Postprocessing Library

Help for Procedures

- <i>: Replace with an integer, or a de-referenced Tcl variable that evaluates to an integer. For example: \$i.
 - <string>: Replace with a string, or a de-referenced Tcl variable that evaluates to a string. For example: \$file.
 - <list_of_r>: Replace with a list of real numbers, or a de-referenced Tcl variable that evaluates to a list of real numbers. For example: \$values.
 - <list_of_strings>: Replace with a list of strings, or a de-referenced Tcl variable that evaluates to a list of strings. For example: \$files.
 - <var_name>: Replace with the *name* of a local Tcl variable.
For example: val (*not* \$val).
 - <list_name>: Replace with the *name* of a local Tcl list.
For example: values (*not* \$values).
 - <array_name>: Replace with the *name* of a local Tcl array.
For example: myarray (*not* \$myarray).
- Brackets – [] – indicate that the argument is optional, but they are *not* part of the syntax.
 - A vertical bar – | – indicates options, only one of which can be specified.

Help for Procedures

To request help on a specific procedure, in Tcl mode, set the `-help` keyword to 1:

```
ifm::<procedure_name> -help 1
```

If this command is included in a Sentaurus Visual file, when Sentaurus Visual is executed in:

- Batch mode in Sentaurus Workbench, the help information is printed to the runtime output file (with the extension .out) of the corresponding Sentaurus Visual node.
- Interactive mode in Sentaurus Workbench, the help information is displayed in the Tcl Console as well as printed in the Sentaurus Visual output file.

You also can enter the command in the Tcl Console of the user interface, in which case, the help information is displayed in the Console.

Output of Procedures

All procedures of the IFM library pass the results back to the calling program by storing the results in a Tcl variable or a Tcl array. The name of this Tcl variable or array is specified as the value of the `-out` keyword.

If there are errors in the IFM library procedures, the behavior of Sentaurus Visual depends on whether it is executed in batch mode or interactive mode in Sentaurus Workbench. In batch mode, Sentaurus Visual exits and an error message is printed only in the Sentaurus Visual error file (with the extension `.err`). In interactive mode, the error message is displayed in the Tcl Console as well as printed in the Sentaurus Visual error file.

All procedures also print several messages (including warning messages). If Sentaurus Visual is executed in batch mode, the messages are printed only in the Sentaurus Visual output file; whereas, in interactive mode, the messages are displayed in the Tcl Console as well as printed in the Sentaurus Visual output file.

The amount of information printed depends on the information level specified by the procedure `lib::SetInfoDef`.

ifm::Gauss

Computes the y-value of a normalized Gaussian distribution for a given x-value:

$$y = \frac{N}{\sigma\sqrt{2\pi}} \exp\left(-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2\right) \quad (15)$$

where N is the norm of the Gaussian distribution, μ is the average, and σ is the standard deviation.

Syntax

```
ifm::Gauss -out <var_name> -x <r> -moments <array_name>
           [-help 0 | 1]
```

Argument	Description
-out <var_name>	Variable name to store the corresponding y-value of the normalized Gaussian distribution.
-x <r>	The x-value. (Real number, no default)
-moments <array_name>	Name of an array with one string-valued index, which contains the elements <code>norm</code> , <code>ave</code> , and <code>std_dev</code> . The values of these elements contain the requested norm, the average, and the standard deviation of the Gaussian. (Array name, no default)
-help 0 1	Prints a help screen if set to 1. Default: 0

Returns

None.

Example

```
set Moments(norm)      1.0
set Moments(ave)       0.0
set Moments(std_dev)   1.0
ifm::Gauss -out G -x 0.1 -moments Moments
puts "The result is $G"
```

ifm::GetDataQuantiles

Computes quantiles for a list of random variables.

This procedure sorts a list of random values and associates each value with the corresponding quantile (a value between 0 and 1).

Syntax

```
ifm::GetDataQuantiles -out <array_name> -rvs <list_of_r>
    [-help 0 | 1]
```

Argument	Description
-out <array_name>	Name of an array to store the results. The array has one string-valued index, which contains the elements x and y. The values of these elements contain the sorted random variables (x) and the corresponding quantiles (y). (Array name, no default)
-rvs <list_of_r>	List of random variables. (List of real numbers, no default)
-help 0 1	Prints a help screen if set to 1. Default: 0

Returns

None.

Example

```
set RanVals [list -1.657 0.7661 2.142 1.189 -1.919 -0.6670 -0.1915
0.3662]
ifm::GetDataQuantiles -out DataQ -rvs $RanVals
puts $DataQ(X)
#-> -1.919 -1.657 -0.6670 -0.1915 0.3662 0.7661 1.189 2.142
puts $DataQ(Y)
#-> 0.0625 0.1875 0.3125 0.4375 0.5625 0.6875 0.8125 0.9375
```

ifm::GetGaussian

Computes either a Gaussian curve:

$$G(x) = \frac{N}{\sigma\sqrt{2\pi}} \exp\left(-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2\right) \quad (16)$$

or the quantiles of a Gaussian curve:

$$Q(x) = \int_{x_{min}}^x G(x)dx \quad (17)$$

where N is the norm of the Gaussian distribution, μ is the average, and σ is the standard deviation.

Syntax

```
ifm::GetGaussian -out <array_name> -moments <array_name>
    [-nsam <i>] [-type f | q] [-xmin <r>] [-xmax <r>] [-help 0 | 1]
```

Argument	Description
-out <array_name>	Name of an array to store the results. The array has one string-valued index, which contains the elements x and y . The values of these elements represent the lists of x - and y -values of the Gaussian or quantile. (Array name, no default)
-moments <array_name>	Name of an array with one string-valued index, which contains the elements <code>norm</code> , <code>ave</code> , and <code>std_dev</code> . The values of these elements contain the requested norm, the average, and the standard deviation of the Gaussian. (Array name, no default)
-nsam <i>	Number of sample points. (Integer, default: 80)
-type f q	f: A Gaussian curve defined by the given moments is returned. q: The quantiles of this Gaussian are returned. Default: f
-xmin <r>	Starting x-value. (Real number, default: -3.0)
-xmax <r>	Ending x-value. (Real number, default: 3.0)
-help 0 1	Prints a help screen if set to 1. Default: 0

Appendix G: Impedance Field Method Data Postprocessing Library

ifm::GetGaussian

Returns

None.

Example

```
create_plot -1d -name Gaussian
select_plots Gaussian
set Moments(norm)    1.0
set Moments(ave)     0.0
set Moments(std_dev) 1.0
ifm::GetGaussian -out Gaussian -type f -moments Moments \
-nsam 100 -xmin -3.5 -xmax 3.5
create_variable -name "GX" -dataset GXY -values $Gaussian(X)
create_variable -name "GY" -dataset GXY -values $Gaussian(Y)
create_curve -name gauss -dataset GXY -axisX "GX" -axisY "GY"
```

ifm::GetHistogram

Computes x- and y-lists to be used to plot a histogram for a given list of random variables, a given plotting range, and a given number of bins.

Syntax

```
ifm::GetHistogram -out <array_name> -rvs <list_of_r>
    -xmin <r> -xmax <r>
    [-nbin <i>] [-info 0 | 1 | 2 | 3] [-help 0 | 1]
```

Argument	Description
-out <array_name>	Name of an array to store the results. The array has one string-valued index, which contains the elements x and y. The values of these elements represent the lists of x- and y-values of the histogram. (Array name, no default)
-rvs <list_of_r>	List of random variables. (List of real numbers, no default)
-xmin <r>	Starting x-value of histogram. (Real number, no default)
-xmax <r>	Ending x-value of histogram. (Real number, no default)
-nbin <i>	Number of bins for the histogram. (Integer, default: 40)
-info 0 1 2 3	Sets local information level. Default: 0
-help 0 1	Prints a help screen if set to 1. Default: 0

Returns

None.

Example

```
create_plot -1d -name Histogram
select_plots Histogram
set Rxs [list 1 1.1 2 5 5.1 5.3 6]
ifm::GetHistogram -out Histogram -rvs $Rxs -xmin 0 -xmax 6 -nbin 6
create_variable -name "X" -dataset XY -values $Histogram(X)
create_variable -name "Y" -dataset XY -values $Histogram(Y)
create_curve -name his -dataset XY -axisX "X" -axisY "Y"
```

ifm::GetMoments

Computes the norm, the average, the root mean square (rms), the standard deviation, the skewness, and the excess kurtosis for a given list of random variables:

$$\mu = \frac{1}{N} \sum_i^N x_i \quad (18)$$

$$x_{\text{rms}} = \sqrt{\frac{1}{N} \sum_i^N x_i^2} \quad (19)$$

$$\sigma = \sqrt{\frac{1}{N} \sum_i^N (x_i - \mu)^2} \quad (20)$$

$$y = \frac{1}{N\sigma^3} \sum_i^N (x_i - \mu)^3 \quad (21)$$

$$k = \frac{1}{N\sigma^4} \sum_i^N (x_i - \mu)^4 - 3 \quad (22)$$

where the norm N is given by the number of random values, the index i enumerates the random values, μ is the average, x_{rms} is the root mean square, σ is the standard deviation, y is the skewness, and k is the excess kurtosis.

Syntax

```
ifm::GetMoments -out <array_name> -rvs <list_of_r>
[-info 0 | 1 | 2 | 3] [-help 0 | 1]
```

Argument	Description
-out <array_name>	Name of a string-valued array to store the computed moments. This array contains the elements <code>norm</code> , <code>ave</code> , <code>rms</code> , <code>std_dev</code> , <code>skew</code> , and <code>kurtx</code> . The values of these elements contain the computed norm, the average, the rms, the standard deviation, the skewness, and the excess kurtosis of the list of random variables. (Array name, no default)
-rvs <list_of_r>	List of random variables. (List of real numbers, no default)
-info 0 1 2 3	Sets local information level. Default: 0
-help 0 1	Prints a help screen if set to 1. Default: 0

Appendix G: Impedance Field Method Data Postprocessing Library

ifm::GetMOSIVs

Returns

None.

Example

```
set Vs [list 1 2 3 4 5]
ifm::GetMoments -out Moments -rvs $Vs
puts $Moments(norm)
#->5
puts $Moments(ave)
#->3
puts $Moments(rms)
#->3.31662479036
puts $Moments(std_dev)
#->1.41421356237
puts $Moments(skew)
#->0.0
puts $Moments(kurtx)
#->-1.3
```

ifm::GetMOSIVs

Constructs the randomized I_d - V_g curves for MOS-type devices for one or more randomization sources.

The boundary condition that links the linear current response $\delta I_{v,d}$ to the nodal drain current $dI_{v,d}$ and the gate voltage $dV_{v,g}$ variations is given by:

$$dI_{v,d} = \delta I_{v,d} + y_{d,g} dV_{v,g} \quad (23)$$

The index v enumerates the randomizations. The given equation gives the freedom to interpret the linear current response directly as a change of the drain current:

$$dI_{v,d} = \delta I_{v,d} \quad (24)$$

Alternatively, you can interpret it as an adjustment of the gate bias:

$$dV_{v,g} = -\frac{\delta I_{v,d}}{y_{d,g}} \quad (25)$$

For the linearized system, the following two methods yield identical results:

- The gate voltage adjustment method (dV):

$$I_{v,d} = I_{ref,d}(V_{ref,g} - dV_{v,g}) \quad (26)$$

- The drain current adjustment method (dI):

$$I_{v,d} = I_{ref,d}(V_{ref,g}) + dI_{v,d} \quad (27)$$

The drain current and the gate voltage of the reference device are given by $I_{\text{ref}, d}$ and $V_{\text{ref}, g}$, respectively.

The equivalence of these two methods can be verified by expanding the two equations into a Taylor series. For a nonlinear system, the two formulations are not equivalent and, depending on the details of the nonlinearity, one or the other method might give more accurate results. To better understand the implications, consider two limiting cases:

(i) Steeply rising $I_d - V_g$ in the subthreshold and near-threshold regime

In this regime, variability effects are well approximated by a threshold voltage shift. This means that, while both $\delta I_{v, d}$ and $y_{d, g}$ increase exponentially with increasing gate bias, the ratio of the two quantities $dV_{v, g}$ remains approximately constant. While large values of $dI_{v, d}$ can result in unphysical negative output currents for some randomizations, when using the drain current adjustment method, the gate voltage adjustment method always guarantees positive and physical output currents.

(ii) Saturating $I_d - V_g$ at low drain bias and high gate bias

In this regime, the transconductance $y_{d, g}$ vanishes and, therefore, $dV_{v, g}$ diverges, while $\delta I_{v, d}$ remains approximately constant. Consequently, the large values of $dV_{v, g}$ can result in unphysical gate voltages (non-monotonous, or less than ground, or larger than the supply voltage) for some randomizations, when using the gate voltage adjustment method. The drain current adjustment method, however, always guarantees monotonous and physical input voltages.

For the $I_d - V_g$ characteristic, the transition point between the subthreshold and near-threshold regime and the saturation regime can be defined as the point of maximal transconductance and, therefore, you can apply either the gate voltage adjustment method or the drain current adjustment method, depending on the sign of the derivative of the transconductance. This observation is the foundation of the third method:

- The weighted method (`weighted`):

$$\begin{aligned} I_{v, d} &= I_{\text{ref}, d} \left(V_{\text{ref}, g} - \frac{1+W}{2} dV_{v, g} \right) \\ I_{v, d} &= I_{\text{ref}, d} (V_{\text{ref}, g}) + \frac{1-W}{2} dI_{v, d} \end{aligned} \quad (28)$$

The weights W are computed internally by calling [ifm::GetMOSWeights on page 483](#).

The weighted method switches between the gate voltage adjustment method and the drain current adjustment method to avoid artificial over-adjustments of the currents or voltages. In situations with relatively large adjustments at the transition point, discontinuities and overlaps might be observed. The `smooth` option activates a smoothening procedure to eliminate these artifacts at the transition point.

An alternative weighting scheme (`ssweighted`) switches from the `dv` method to the `dI` method when the subthreshold slope becomes larger than the user-defined threshold `ss`.

Appendix G: Impedance Field Method Data Postprocessing Library

ifm::GetMOSIVs

This method uses an error function to smooth out the transition, and you can control the smoothness by setting the normalization parameter `dss`. The weights are computed internally by calling [ifm::GetMOSWeights on page 483](#).

For I_d - V_g sweeps in the linear regime, it is recommended to use one of the weighted methods, preferably the one that gives the smoothest resulting I-V curves.

Finally, the conceptually simpler exponential method ensures nonnegative currents and also avoids gate bias overshoots. This method often gives satisfactory results, but it violates the linearity assumption:

- The exponential method (`exp`):

$$I_{v,d} = I_{\text{ref},d} \exp\left(\frac{dI_{v,d}}{I_{\text{ref},d}}\right) \quad (29)$$

Even when using the most appropriate I_d - V_g construction method, it might happen that for a certain randomization, the linear current response $dI_{v,d}$ becomes too large compared to the nominal current $I_{v,d}$ and the resulting I_d - V_g might exhibit an unexpected shape. This can result in the unreliable extraction of electrical parameters such as the subthreshold slope or the threshold voltage for these specific randomizations. You can flag and filter out such curves by looking at the maximum deviation $|dI_{v,d}/I_{v,d}|$. The maximum deviation is computed within a user-specified bias range for each constructed I_d - V_g and is accessible using the array element `maxdev` of the array that also contains the x- and y-values of the I_d - V_g curves. You can limit the bias range by specifying the `-vmin` and `-vmax` arguments.

Syntax

```
ifm::GetMOSIVs -out <array_name> -sifm <array_name> -nrow <var_name>
    -ncol <var_name> -v <list_of_r> -i <list_of_r> -y <list_of_r>
    -vmin <r> -vmax <r> -id <string>
    [-type IdVg] [-method dv | dI | weighted | SSweighted | exp]
    [-smooth yes | no] [-sgn 1 | -1] [-ss <r> -dss <r>]
    [-info 0 | 1 | 2 | 3] [-help 0 | 1]
```

Argument	Description
<code>-out <array_name></code>	Name of a two-indexed array to store the constructed I-V curves. The first index is string valued. The elements are <code>x</code> and <code>y</code> . The second index is integer valued. It represents the randomization index. The array element values contain the x- and y-value lists of the respective I-V curves. The array element <code>maxdev</code> contains the maximum deviation: $\text{maxdev} = \max dI/I $ for $v_{\text{min}} < V < v_{\text{max}}$ for the given I-V curve. It can be used to filter out curves for which <code>maxdev</code> is too big, for example, greater than 1. (Array name, no default)

Appendix G: Impedance Field Method Data Postprocessing Library

ifm::GetMOSIVs

Argument	Description
-sifm <array_name>	Name of an array that contains the sIFM data. The array has three indices: The first index is string valued. The elements are the variability source identifiers. The second index is integer valued. It represents the row or bias index. The third index is integer valued. It represents the column or randomization index. The array element values contain the sIFM linear current response. (Array name, no default)
-nrow <var_name>	Name of a variable containing the number of rows (bias points) in the sIFM data. (Variable name, no default)
-ncol <var_name>	Name of a variable containing the number of columns (randomizations) in the sIFM data. (Variable name, no default)
-v <list_of_r>	List containing the reference voltage values. (List of real numbers, no default)
-i <list_of_r>	List containing the reference current values. (List of real numbers, no default)
-y <list_of_r>	List containing the relevant reference Y-matrix values. (List of real numbers, no default)
-vmin <r>	Minimum bias for dI/I monitoring.
-vmax <r>	Maximum bias for dI/I monitoring.
-id <string>	ID of the sIFM variability source. (String, no default)
-type IdVg	Currently, only <code>IdVg</code> is supported. Default: <code>IdVg</code>
-method dv di weighted ssweighted exp	Selects the I-V construction method. Default: <code>weighted</code>
-smooth yes no	Activates I-V smoothening for the weighted method. Default: <code>no</code>
-sgn 1 -1	Set to <code>1</code> for NMOS or <code>-1</code> for PMOS. Default: <code>1</code>
-ss <r>	Subthreshold slope inflection point in mV/decade. (Real, only needed for <code>ssweighted</code>)

Appendix G: Impedance Field Method Data Postprocessing Library

ifm:::GetMOSIVs

Argument	Description
-dss <r>	Width of transition region in mV/decade. (Real, only needed for SSweighted)
-info 0 1 2 3	Sets local information level. Default: 0
-help 0 1	Prints a help screen if set to 1. Default: 0

Returns

None.

Example

```

set IDs [list rdf ift SUM]
set FILES [list]
foreach ID $IDs {
lappend FILES mos_${ID}_I_ndrain.csv
}

load_file mos_circuit_des.plt -name Data(DC)
load_file mos_ac_des.plt      -name Data(AC)

set adgs [get_variable_data "a(ndrain,ngate)" -dataset Data(AC) ]
set Vgs  [get_variable_data "v(ngate)"           -dataset Data(DC) ]
set Ids   [get_variable_data "i(mos,ndrain)"     -dataset Data(DC) ]

ifm:::ReadsIFM -out sIFM -nrow Nrow -ncol Ncol -files $FILES -ids $IDs

create_plot -1d -name RanIV
select_plots RanIV

set j 42
ifm:::GetMOSIVs -out IV_rdf -sifm sIFM -nrow Nrow -ncol Ncol \
-method "weighted" -sgn 1.0 -v $Vgs -i $Ids -y $adgs -id "rdf" -smooth
yes
create_variable -name V -dataset RanIV(rdf,$j) -values $IV_rdf(X,$j)
create_variable -name I -dataset RanIV(rdf,$j) -values $IV_rdf(Y,$j)
create_curve -name IV_rdf($j) -dataset RanIV(rdf,$j) -axisX "V" -axisY
"I"

ifm:::GetMOSIVs -out IV_ift -sifm sIFM -nrow Nrow -ncol Ncol \
-method "weighted" -sgn 1.0 -v $Vgs -i $Ids -y $adgs -id "ift" -smooth
yes
create_variable -name V -dataset RanIV(ift,$j) -values $IV_ift(X,$j)
create_variable -name I -dataset RanIV(ift,$j) -values $IV_ift(Y,$j)
create_curve -name IV_ift($j) -dataset RanIV(ift,$j) -axisX "V" -axisY
"I"

ifm:::GetMOSIVs -out IV_SUM -sifm sIFM -nrow Nrow -ncol Ncol \

```

Appendix G: Impedance Field Method Data Postprocessing Library

ifm::GetMOSWeights

```
-method "weighted" -sgn 1.0 -v $Vgs -i $Ids -y $adgs -id "SUM" -smooth  
yes  
create_variable -name V -dataset RanIV(SUM,$j) -values $IV_SUM(X,$j)  
create_variable -name I -dataset RanIV(SUM,$j) -values $IV_SUM(Y,$j)  
create_curve -name IV_SUM($j) -dataset RanIV(SUM,$j) -axisX "V" -axisY  
"I"
```

ifm::GetMOSWeights

Computes the weights for the construction of randomized MOSFET I_d - V_g curves.

This procedure computes, for each bias point, a weight between 1 and -1 to indicate that the gate-voltage adjustment method or the drain-current adjustment method is to be used.

For `-type IdVg_weighted`, the weights are set to 1 or -1 depending on the sign of the derivative of the transconductance.

For `-type IdVg_SSweighted`, the weights are computed as:

$$W = \operatorname{erf}\left(\frac{ss - SS}{dss}\right) \quad (30)$$

where SS is the subthreshold slope, and ss is the user-defined switch-over threshold. The smoothness of the transition is set by the user-defined normalization parameter dss .

Syntax

```
ifm:::GetMOSWeights -out <list_name> -y <list_of_r>  
  [-type IdVg_weighted | IdVg_SSweighted]  
  [-i <list_of_r> -ss <r> -dss <r>]  
  [-help 0 | 1]
```

Argument	Description
<code>-out <list_name></code>	Name of a list to store the computed weights. (List name, no default)
<code>-y <list_of_r></code>	List containing the relevant reference Y-matrix values. (List of real numbers, no default)
<code>-type IdVg_weighted IdVg_SSweighted</code>	Selects the method used for the computation of the weights. Default: <code>IdVg_weighted</code>
<code>-i <list_of_r></code>	List containing the reference current values. (List of real numbers, only needed for <code>IdVg_SSweighted</code>)
<code>-ss <r></code>	Subthreshold slope inflection point in mV/decade. (Real, only needed for <code>IdVg_SSweighted</code>)

Appendix G: Impedance Field Method Data Postprocessing Library

ifm::GetNoiseStdDev

Argument	Description
-dss <r>	Width of transition region in mV/decade. (Real, only needed for IdVg_SSweighted)
-help 0 1	Prints a help screen if set to 1. Default: 0

Returns

None.

Example

```
load_file mos_ac_des.plt -name Data(AC)
set adgs [get_variable_data a(ndrain,ngate) -dataset Data(AC)]
ifm::GetMOSWeights -out Ws -y $adgs
create_variable -name W -dataset Data(AC) -values $Ws
create_plot -ld -name Weights
select_plots Weights
create_curve -name W -dataset Data(AC) -axisX "v(ngate)" -axisY "W"
create_curve -name Y -dataset Data(AC) -axisX "v(ngate)" \
-axisY2 "a(ndrain,ngate)"
```

ifm::GetNoiseStdDev

Computes the drain current $\sigma(I_d)$ and the gate voltage $\sigma(V_g)$ standard deviation from the drain current noise spectral density $S_{d,d}$ (see *Sentaurus™ Device User Guide*, Chapter 23), and the gate-to-drain admittance $y_{d,g}$:

$$\sigma(I_d) = \sqrt{S_{d,d} \cdot 1\text{Hz}} \quad (31)$$

$$\sigma(V_g) = \frac{\sqrt{S_{d,d} \cdot 1\text{Hz}}}{y_{d,g}} \quad (32)$$

Syntax

```
ifm::GetNoiseStdDev -out <array_name> -s <list_of_r> -y <list_of_r>
[-help 0 | 1]
```

Argument	Description
-out <array_name>	Name of an array to store the results. The array has one string-valued index. The index contains the elements I and V. The value of the I element is a list with the current standard deviations. The value of the V element is a list with the voltage standard deviations. (Array name, no default)

Appendix G: Impedance Field Method Data Postprocessing Library

ifm::GetNoiseStdDev

Argument	Description
-s <list_of_r>	List containing the current noise spectral density values. (List of real numbers, no default)
-y <list_of_r>	List containing the relevant reference Y-matrix values. (List of real numbers, no default)
-help 0 1	Prints a help screen if set to 1. Default: 0

Returns

None.

Example

```
load_file "mos_ac_des.plt" -name Data(AC)
set adgs      [get_variable_data a(ndrain,ngate) -dataset Data(AC)]
set S_Ids(noise) [get_variable_data S_I(ndrain)      -dataset Data(AC)]
ifm:::GetNoiseStdDev -out sigIV -s $S_Ids(noise) -y $adgs
create_plot -1d -name Sig
create_variable -name sigId(noise) -dataset Data(AC) -values $sigIV(I)
create_curve -name sigId(noise) -dataset Data(AC) \
-axisX "v(ngate)" -axisY "sigId(noise)"
create_variable -name sigVg(noise) -dataset Data(AC) -values $sigIV(V)
create_curve -name sigVg(noise) -plot Sig -dataset Data(AC) \
-axisX "v(ngate)" -axisY2 "sigVg(noise)"
```

ifm::GetQQ

Compares the quantiles of a given data distribution with the quantiles of a Gaussian distribution.

For each value in the quantiles of the Gaussian distribution, the matching (interpolated) value of the data distributions is found. Then, the data x-values corresponding to this match are plotted against the normalized Gaussian x-values $(x - \mu)/\sigma$.

Syntax

```
ifm::GetQQ -out <array_name> -dq <array_name> -gq <array_name>
-moments <array_name> [-help 0 | 1]
```

Argument	Description
-out <array_name>	Name of an array to store the results. The array has one string-valued index, which contains the elements x and y. The values of these elements represent the lists of x- and y-values of a quantile-quantile comparison curve. (Array name, no default)
-dq <array_name>, -gq <array_name>	Name of arrays containing the quantiles of the given data distribution (dq) and the quantiles of a corresponding Gaussian distribution (gq). The arrays dq and gq each have one string-valued index, which contains the elements x and y. The values of these elements represent the lists of x- and y-values of the quantiles. (Array name, no default)
-moments <array_name>	Name of an array with one string-valued index, which contains the elements norm, ave, and std_dev. The values of these elements contain the requested norm, the average, and the standard deviation of the Gaussian. (Array name, no default)
-help 0 1	Prints a help screen if set to 1. Default: 0

Returns

None.

Example

```
set RanVals [list -1.657 0.7661 2.142 1.189 -1.919 -0.6670 -0.1915
0.3662]
ifm::GetDataQuantiles -out DataQ -rvs $RanVals
ifm::GetMoments -out Moments -rvs $RanVals
```

Appendix G: Impedance Field Method Data Postprocessing Library

ifm::GetsIFMStdDev

```
ifm:::GetGaussian -out GaussianQ -type q -moments Moments \
-nsam 40 -xmin -3.5 -xmax 3.5
ifm:::GetQQ -out QQ -dq DataQ -gq GaussianQ -moments Moments
create_plot -1d -name QQplot
select_plots QQplot
create_variable -name "QQX" -dataset QQXY -values $QQ(X)
create_variable -name "QQY" -dataset QQXY -values $QQ(Y)
create_curve -name qq -dataset QQXY -axisX "QQX" -axisY "QQY"
```

ifm::GetsIFMStdDev

Computes the drain current and the gate voltage standard deviation from the sIFM linear current responses and the gate-to-drain admittance.

For each bias point in the sIFM data file, this procedure reads the linear current responses and calls `ifm:::GetMoments` to compute the drain current standard deviation $\sigma(I_d)$. The gate voltage $\sigma(V_g)$ standard deviation is obtained by dividing the drain current standard deviation by the gate-to-drain admittance $y_{d,g}$.

This procedure also supports the computation of the drain-current and the gate-voltage standard deviation for contact resistance variability.

Syntax

```
ifm:::GetsIFMStdDev -out <array_name> -sIFM <string> -y <list_of_r>
[-rsig <r> -ydd <list_of_r> -i <list_of_r>] [-help 0 | 1]
```

Argument	Description
<code>-out <array_name></code>	Name of an array to store the results. The array has one string-valued index. The index contains the elements <code>I</code> and <code>V</code> . The value of the <code>I</code> element is a list with the current standard deviations. The value of the <code>V</code> element is a list with the voltage standard deviations. (Array name, no default)
<code>-sIFM <string></code>	Name of a comma-separated value (CSV) file containing the sIFM linear current responses. If this string is set to " <code>crv</code> " the contact resistance variability is computed instead. (String, no default)
<code>-y <list_of_r></code>	List containing the relevant reference Y-matrix values. (List of real numbers, no default)
<code>-rsig <r></code>	Standard deviation of contact resistance variability in Ohm. Activated when sIFM is set to " <code>crv</code> ". (Real, only needed for contact resistance variability)

Appendix G: Impedance Field Method Data Postprocessing Library

ifm::GetSNM

Argument	Description
-ydd <list_of_r>	List containing the reference Y(d,d) matrix elements. Activated when sIFM is set to "crv". (List of real numbers, only needed for contact resistance variability)
-i <list_of_r>	List containing the reference current values. Activated when sIFM is set to "crv". (List of real numbers, only needed for contact resistance variability)
-help 0 1	Prints a help screen if set to 1. Default: 0

Returns

None.

Example

```
load_file "mos_ac_des.plt" -name Data(AC)
set adgs [get_variable_data a(ndrain,ngate) -dataset Data(AC)]

ifm::GetsIFMStdDev -out sigIV -sIFM "mos_rdf_I_ndrain.csv" -y $adgs

create_plot -ld -name Sig
set_plot_prop -plot Sig -title "Standard Deviations"

create_variable -name sigId(stat) -dataset Data(AC) -values $sigIV(I)
create_curve -name sigId(stat) -dataset Data(AC) \
-axisX "v(ngate)" -axisY "sigId(stat)"
create_variable -name sigVg(stat) -dataset Data(AC) -values $sigIV(V)
create_curve -name sigVg(stat) -dataset Data(AC) \
-axisX "v(ngate)" -axisY2 "sigVg(stat)"
```

ifm::GetSNM

Computes the static noise margins (SNMs) from butterfly curves for one or more randomization sources.

This procedure takes as input the voltage transfer characteristics (VTC) curves of the left and the right inverters of an SRAM cell. One plot of all VTC curves is known as a *butterfly curve*. The left (right) SNM is defined as the axis-aligned biggest square that can be fitted into the left (right) lob of the butterfly curve. The effective SNM is defined as the smaller value of the two.

Syntax

```
ifm::GetSNM -out <array_name> -squares <array_name>
    -vtc_left <array_name> -vtc_right <array_name>
    -ncol <var_name> [-info 0 | 1 | 2 | 3] [-help 0 | 1]
```

Argument	Description
-out <array_name>	Name of a string-indexed array to store the results. The elements are <code>left</code> , <code>right</code> , and <code>eff</code> for the left, right, and effective SNMs. Each array entry contains a list of the respective SNM values for all randomizations. (Array name, no default)
-squares <array_name>	Name of a three-indexed array to store the fitted squares representing the SNM in the butterfly curve. The first index is integer valued. The elements are <code>1</code> or <code>2</code> for the square of the left or right lob of the butterfly curve. The second index is string valued. The elements are <code>x</code> or <code>y</code> for the x-values and y-values of the fitted square. The third index is integer valued and represents the randomization index. (Array name, no default)
-vtc_left <array_name>, -vtc_right <array_name>	Names of two-indexed arrays containing the left and right VTC curves. The first index is string valued. The elements are <code>Header</code> or <code>Data</code> , where the <code>Header</code> contains the names of the columns, such as <code>Vi(0)</code> . The corresponding <code>Data</code> field contains a list of voltage values. The second index is integer valued and represents the randomization index. (Array name, no default)
-ncol <var_name>	Name of a variable containing the number of randomizations in the VTC data. (Variable name, no default)
-info 0 1 2 3	Sets local information level. Default: 0
-help 0 1	Prints a help screen if set to 1. Default: 0

Returns

None.

Example

```
ifm::ReadCSV -out VTC_L -file Left_VTC.csv -ncol Ncol
ifm::ReadCSV -out VTC_R -file Right_VTC.csv -ncol Ncol
set j 42
set i_in [expr 2*$j]
set i_ot [expr 2*$j+1]

create_variable -name Vi($j) -dataset VTC(L) -values $VTC_L(Data,$i_in)
```

Appendix G: Impedance Field Method Data Postprocessing Library

ifm::GetSRAMVTC

```
create_variable -name Vo($j) -dataset VTC(L) -values $VTC_L(Data,$i_ot)
create_curve -name VTC(L,$j) -dataset VTC(L) -axisX "Vi($j)" \
-axisY "Vo($j)"

create_variable -name Vi($j) -dataset VTC(R) -values $VTC_R(Data,$i_ot)
create_variable -name Vo($j) -dataset VTC(R) -values $VTC_R(Data,$i_in)
create_curve -name VTC(R,$j) -dataset VTC(R) -axisX "Vi($j)" \
-axisY "Vo($j)"

ifm:::GetSNM -out SNM -squares SQ -vtc_left VTC_L -vtc_right VTC_R \
-ncol Ncol

create_variable -name SQ1_x($j) -dataset SQs -values $SQ(1,X,$j)
create_variable -name SQ1_y($j) -dataset SQs -values $SQ(1,Y,$j)
create_variable -name SQ2_x($j) -dataset SQs -values $SQ(2,X,$j)
create_variable -name SQ2_y($j) -dataset SQs -values $SQ(2,Y,$j)

create_curve -name SQ1($j) -dataset SQs -axisX "SQ1_x($j)" \
-axisY "SQ1_y($j)"
create_curve -name SQ2($j) -dataset SQs -axisX "SQ2_x($j)" \
-axisY "SQ2_y($j)"

puts "Left      SNM: [lindex $SNM(left)  $j]"
puts "Right     SNM: [lindex $SNM(right) $j]"
puts "Effective SNM: [lindex $SNM(eff)   $j]"
```

ifm::GetSRAMVTC

Constructs randomized VTC curves for an SRAM cell for one or more randomization sources.

To compute the randomized VTC of an inverter from an SRAM cell, a method similar to the weighted method outlined in [ifm::GetMOSIVs on page 478](#) for the single transistor is used. Unlike in a single transistor, in an SRAM cell, the output node is not connected to an external voltage source and, therefore, Kirchhoff's law requires that $dI_{v,o} = 0$.

For the extraction of SRAM SNMs, you cannot require that the output small-signal voltage variation vanishes because, during the *read* operation of the SRAM cell (access transistor is switched on) in the region of low output bias (PMOS transistor is switched off), the actual voltage of the output node is defined by the voltage divider formed by the access transistor and the NMOS transistor. The current flow fluctuations through the access transistor cannot be adequately compensated by adjusting the gate voltage of the NMOS (and PMOS) transistor. A solution for the *v*-th randomized SRAM cell for which all currents through the inverters are the same, as in the reference SRAM cell, while additionally requiring that the voltage at the output is also the same in the reference inverter would not be physical. For example, for certain bias conditions, unrealistically large gate voltage adjustments would be needed to overcompensate the random dopant fluctuation effects in the access transistor.

Appendix G: Impedance Field Method Data Postprocessing Library

ifm::GetSRAMVTC

To obtain physically relevant VTC curves, both the output and input voltages are adjusted:

$$\delta I_{v,o} = -y_{o,i} dV_{v,i} - y_{o,o} dV_{v,o} \quad (33)$$

A method, which results in physical VTC curves, consists of adjusting both $dV_{v,i}$ and $dV_{v,o}$ based on an automatic analysis of the current flow in the reference device. For this method, the voltage variations $dV_{v,i}$ and $dV_{v,o}$ are expressed in terms of voltage variations in the coordinate system that is rotated:

$$\begin{bmatrix} dV_{v,o} \\ dV_{v,i} \end{bmatrix} = \begin{bmatrix} \cos(\varphi) & \sin(\varphi) \\ -\sin(\varphi) & \cos(\varphi) \end{bmatrix} \begin{bmatrix} dV_{v,1} \\ dV_{v,2} \end{bmatrix} \quad (34)$$

In the rotated coordinate system, the boundary condition is imposed such that $dV_{v,1} = 0$, resulting in:

$$dV_{v,2} = -\frac{\delta I_{v,o}}{y_{o,i} \sin(\varphi) + y_{o,o} \cos(\varphi)} \quad (35)$$

At each bias point, an angle is selected that ensures a monotonous and physical solution.

For example, you can find out whether you are in the *hold*-like inverter regime (access transistor is closed) or in the voltage divider regime (PMOS transistor is closed) by monitoring the current flows in the reference SRAM cell. The reference current through the inverter is the sum of the currents through the PMOS and the access transistors. If the main contribution comes from the PMOS device, you are in the hold-like inverter regime, and you set φ to $\pi/2$. On the other hand, if the main contribution comes from the access transistor, you are in the voltage divider regime, and you set φ to 0. Therefore, you can use the reference current to select the appropriate angle and then compute $dV_{v,2}$.

An example of a script to compute the current-controlled angle is:

```
set Ips [get_variable_data "$pSOURCE TotalCurrent" -dataset Data(DC)]
set Ias [get_variable_data "$aDRAIN TotalCurrent" -dataset Data(DC)]
foreach Ip $Ips Ia $Ias {
    set It [expr $Ip + $Ia]
    lappend fis [expr 0.5*$pi*$Ip/$It]
}
```

Here, `pSOURCE` points to the source contact of the PMOS transistor, and `aDRAIN` points to the drain contact of the access transistor of the inverters of interest in the SRAM cell.

Syntax

```
ifm::GetSRAMVTC -out <array_name> -vin <list_of_r> -vout <list_of_r>
    -fi <list_of_r> -aoi <list_of_r> -aoe <list_of_r> -id <string>
    -sifm <array_name> -nrow <var_name> -ncol <var_name>
    [-info 0 | 1 | 2 | 3] [-help 0 | 1]
```

Appendix G: Impedance Field Method Data Postprocessing Library

ifm::GetSRAMVTC

Argument	Description
-out <array_name>	Name of a two-indexed array to store the constructed VTC curves. The first index is string valued. The elements are Header or Data, where the Header contains the names of the columns, such as vi(0). The corresponding Data field contains a list of voltage values. The second index is integer valued. It represents the randomization index. (Array name, no default)
-vin <list_of_r>	List containing the reference VTC input voltage values. (List of real numbers, no default)
-vout <list_of_r>	List containing the reference VTC output voltage values. (List of real numbers, no default)
-fi <list_of_r>	List containing the current-controlled angle values. (List of real numbers, no default)
-aoi <list_of_r>	List containing the reference ReY(out,in) matrix values. (List of real numbers, no default)
-aoo <list_of_r>	List containing the reference ReY(out,out) matrix values. (List of real numbers, no default)
-id <string>	ID of the sIFM variability source. (String, no default)
-sifm <array_name>	Name of an array that contains the sIFM data. The array has three indices: The first index is string valued. The elements are the variability source identifiers. The second index is integer valued and represents the row or bias index. The third index is integer valued and represents the column or randomization index. The array element values contain the sIFM linear current response. (Array name, no default)
-nrow <var_name>	Name of a variable containing the number of rows (bias points) in the sIFM data. (Variable name, no default)
-ncol <var_name>	Name of a variable containing the number of columns (randomizations) in the sIFM data. (Variable name, no default)

Appendix G: Impedance Field Method Data Postprocessing Library

ifm::GetSRAMVTC

Argument	Description
-info 0 1 2 3	Sets local information level. Default: 0
-help 0 1	Prints a help screen if set to 1. Default: 0

Returns

None.

Example

```
load_file sys_des.plt      -name SYSTEM
load_file SRAM_des.plt     -name DC
load_file sram_ac_des.plt -name AC
set Vins [get_variable_data v($IN)           -dataset SYSTEM]
set Vots [get_variable_data v($OUT)          -dataset SYSTEM]
set aois [get_variable_data a($OUT,$IN)       -dataset AC]
set aoos [get_variable_data a($OUT,$OUT)      -dataset AC]
set Ips  [get_variable_data "SourceP2 TotalCurrent" -dataset DC]
set Ias  [get_variable_data "DrainACC2 TotalCurrent" -dataset DC]
foreach Ip $Ips Ia $Ias {
    set It [expr $Ip + $Ia]
    lappend fis [expr 0.5*$ifm::pi*$Ip/$It]
}
ifm::ReadsIFM -out sIFM -nrow Nrow -ncol Ncol \
-files "flipL_n12_sram_rdf_I_OR.csv" -ids "rdf"
ifm::GetSRAMVTC -out VTC -id "rdf" -vin $Vins -vout $Vots \
-fi $fis -aoi $aois -ao0 $aoos -sifm sIFM -nrow Nrow -ncol Ncol
create_plot -1d -name RanVTC
select_plots RanVTC
set j 42
set i_in [expr 2*$j]
set i_ot [expr 2*$j+1]
create_variable -name Vi($j) -dataset RanVTC(rdf) -values
$VTC(Data,$i_in)
create_variable -name Vo($j) -dataset RanVTC(rdf) -values
$VTC(Data,$i_ot)
create_curve -name VTC($j) -dataset RanVTC(rdf) \
-axisX "Vi($j)" -axisY "Vo($j)"
```

ifm::ReadCSV

Reads a CSV file, and passes the read data to the calling program in the form of a Tcl array.

Note:

The CSV files are assumed to have the following format: one header line containing the names of the datasets (no spaces) followed by a number of rows containing the values in the dataset. For example:

-1.11e-12,3.92e-14,-1.66e-13,-6.09e-13,... (no spaces)

Syntax

```
ifm::ReadCSV -out <array_name> -file <string> -ncol <var_name>
[-info 0 | 1 | 2 | 3] [-help 0 | 1]
```

Argument	Description
-out <array_name>	Name of an array with two indices to store the read data. The first index is string valued and contains the elements <code>Header</code> and <code>Data</code> . The second index is integer valued and enumerates the number of columns in the CSV file. The values of the <code>Header</code> elements are the dataset names. The values of the <code>Data</code> elements contain lists with the values in the dataset. (Array name, no default)
-file <string>	Name of the CSV file to be read. (String, no default)
-ncol <var_name>	Name of a variable containing the number of columns found in the CSV file. (Variable name, no default)
-info 0 1 2 3	Sets local information level. Default: 0
-help 0 1	Prints a help screen if set to 1. Default: 0

Returns

None.

Example

```
set Ncol 3
set CSV(Header,0) "A"
set CSV(Header,1) "B"
set CSV(Header,2) "C"
set CSV(Data,0) [list 1.1 1.2 1.3 1.4]
```

Appendix G: Impedance Field Method Data Postprocessing Library

ifm::ReadsIFM

```
set CSV(Data,1) [list 2.1 2.2 2.3 2.4]
set CSV(Data,2) [list 3.1 3.2 3.3 3.4]
ifm::WriteCSV -csv CSV      -ncol Ncol      -file "my.csv"
ifm::ReadCSV  -out ReadCSV -ncol ReadNcol -file "my.csv"
for {set icol 0} {$icolumn < $ReadNcol} {incr icol} {
    puts "Column name is: $ReadCSV(Header,$icolumn)"
    puts "Column data is: $ReadCSV(Data,$icolumn)"
}
Column name is: A
Column data is: 1.1 1.2 1.3 1.4
Column name is: B
Column data is: 2.1 2.2 2.3 2.4
Column name is: C
Column data is: 3.1 3.2 3.3 3.4
```

ifm::ReadsIFM

Reads one or more sIFM CSV files containing the linear current responses, and passes the read data to the calling program in the form of a Tcl array.

This procedure also supports the computation of the linear current responses for contact resistance variability.

Note:

The CSV files are assumed to have the following format: one header line containing the names of the datasets (no spaces) followed by a number of rows containing the values in the dataset. For example:

-1.11e-12,3.92e-14,-1.66e-13,-6.09e-13,... (no spaces)

Syntax

```
ifm::ReadsIFM -out <array_name> -files <list_of_strings>
               -ids <list_of_strings> -nrow <var_name> -ncol <var_name>
               [-rsig <r> -nrand <i> -rseed <i> -ydd <list_of_r>] -i <list_of_r>
               [-info 0 | 1 | 2 | 3] [-help 0 | 1]
```

Appendix G: Impedance Field Method Data Postprocessing Library

ifm::ReadsIFM

Argument	Description
-out <array_name>	Name of an array to store the read data. The array has three indices: The first index is string valued. The elements are the variability source identifiers. The second index is integer valued and represents the row or bias index. The third index is integer valued and represents the column or randomization index. The array element values contain the sIFM linear current response. (Array name, no default)
-files <list_of_strings>	List containing the names of the sIFM CSV data files. (List of strings, no default)
-ids <list_of_strings>	List containing the variability source identifiers. If the list contains the special identifier <code>SUM</code> , the combined data from all variability sources will also be computed. (List of strings, no default)
-nrow <var_name>	Name of a variable to store the number of rows (bias points) found in the sIFM CSV file. (Variable name, no default)
-ncol <var_name>	Name of a variable to store the number of columns (randomizations) found in the sIFM CSV file. (Variable name, no default)
-rsig <r>	Standard deviation of contact resistance variability in Ohm. Activated when ids contains " <code>crv</code> ". (Real, only needed for contact resistance variability)
-nrand <i>	Number of random samples. Activated when ids contains " <code>crv</code> ". (Integer, only needed for contact resistance variability)
-rseed <i>	Random number seed. Activated when ids contains " <code>crv</code> ". (Integer between 1 and 2147483647; only needed for contact resistance variability)
-ydd <list_of_r>	List containing the reference Y(d,d) matrix elements. Activated when ids contains " <code>crv</code> ". (List of real numbers, only needed for contact resistance variability)
-i <list_of_r>	List containing the reference current values. Activated when ids contains " <code>crv</code> ". (List of real numbers, only needed for contact resistance variability)

Appendix G: Impedance Field Method Data Postprocessing Library

ifm::ReadsIFM

Argument	Description
-info 0 1 2 3	Sets local information level. Default: 0
-help 0 1	Prints a help screen if set to 1. Default: 0

Returns

None.

Example

```
set IDs [list rdf ift SUM]
set FILES [list rdf_I_ndrain.csv ift_I_ndrain.csv SUM_I_ndrain.csv]
ifm::ReadsIFM -out sIFM -nrow Nrow -ncol Ncol -files $FILES -ids $IDs
puts "The linear drain current response due to random dopant fluctuations
in the 42th randomization at 12th bias point is: sIFM(rdf,12,42) =
\$sIFM(rdf,12,42). The corresponding response to interface traps is
sIFM(ift,12,42) = \$sIFM(ift,12,42). The combined response is
sIFM(SUM,12,42) = \$sIFM(SUM,12,42)"
```

Note:

The CSV file associated with the `SUM` ID is not actually read and, therefore, it does not have to exist. The actual dataset is computed automatically by summing all previously read datasets.

ifm::WriteCSV

Writes a Tcl array to a CSV file.

Syntax

```
ifm::WriteCSV -file <string> -csv <array_name> -ncol <var_name>
    [-info 0 | 1 | 2 | 3] [-help 0 | 1]
```

Argument	Description
-file <string>	Name of a CSV file to be written. (String, no default)
-csv <array_name>	Name of an array with two indices, containing the data to be written. The first index is string valued and contains the elements <code>Header</code> and <code>Data</code> . The second index is integer valued and enumerates the number of columns in the CSV file. The values of the <code>Header</code> elements are the dataset names. The values of the <code>Data</code> elements contain lists with the values in the dataset. (Array name, no default)
-ncol <var_name>	Name of a variable containing the number of columns in the CSV data to be written. (Variable name, no default)
-info 0 1 2 3	Sets local information level. Default: 0
-help 0 1	Prints a help screen if set to 1. Default: 0

Returns

None.

Example

```
set Ncol 3
set CSV(Header,0) "A"
set CSV(Header,1) "B"
set CSV(Header,2) "C"
set CSV(Data,0) [list 1.1 1.2 1.3 1.4]
set CSV(Data,1) [list 2.1 2.2 2.3 2.4]
set CSV(Data,2) [list 3.1 3.2 3.3 3.4]
ifm::WriteCSV -csv CSV -ncol Ncol -file "my.csv"
```

lib::SetInfoDef

Sets the default information level.

Note:

Level 0: Warning, error, or status messages only.

Level 1: Echo results.

Level 2: Show progress and some debug information.

Level 3: Show all debug information.

The local information level also can be set using the `-info` keyword of the procedures in the IFM library.

Syntax

```
lib::SetInfoDef 0 | 1 | 2 | 3
```

Argument	Description
<code><info_level></code>	Sets the default information level. Default: 0

Returns

None.

Example

```
lib::SetInfoDef 2
```

H

Two-Port Network RF Extraction Library

This appendix provides information about the procedures of the RF extraction library.

Under the assumption that a transistor can be modeled by a two-port network, the procedures of the two-port network radio frequency (RF) extraction library are used to compute:

- RF parameters from AC analysis data.
- Noise parameters from noise analysis data.

The functionality of the RF extraction library includes:

- RF matrix conversion: Converting an admittance (Y-)matrix to a hybrid (h-)matrix, a scattering (S-)matrix, and an impedance (Z-)matrix.
- Plotting the small-signal data (conductance, capacitance, and RF parameters). In addition to rectangular plots, polar plots and Smith charts are supported.
- Plotting the following noise spectral densities (NSDs) or power spectral densities (PSDs) of various representations (noise equivalent circuits) of a noisy transistor:
 - Noise voltage spectral density (NVSD) for impedance representation.
 - Noise current spectral density (NISD) for admittance representation.
- RF parameter extraction:
 - Computing small-signal current gain, stability criteria such as the Rollett stability factor and the stability condition delta, various power gains such as maximum available gain (MAG), maximum stable gain (MSG), Mason's unilateral gain (MUG), and unilateral figure of merit.
 - Extracting transistor figures of merit such as cutoff frequency, maximum frequency of oscillation, and cutoff frequency for stability.

Appendix H: Two-Port Network RF Extraction Library

Syntax Conventions

- Noise parameter extraction:
 - Computing PSDs of the equivalent input noise generators in chain representation of a noisy transistor.
 - Computing the noise figure of a transistor.
 - Extracting various noise parameters of a transistor such as the minimum noise figure, the equivalent noise resistance and conductance, and the optimum source admittance and impedance.
- Complex arithmetic support (both scalar and vectorial versions).
- Exporting data in comma-separated value (CSV) file format.

The RF extraction library is loaded with the command:

```
load_library rfx
```

Syntax Conventions

The RF extraction library uses a unique namespace identifier (`rfx::`) for its procedures. All procedures and variables associated with this library are called with the namespace identifier prepended. For example:

```
rfx::<proc_name>
```

Each procedure has several arguments. The RF extraction library uses an input parser that accepts arguments of the form:

```
-keyword <value>
```

Note:

All Sentaurus Visual libraries support the standard Sentaurus Visual syntax in which keywords are preceded by a dash. For backward compatibility, all Sentaurus Visual libraries continue to support the `keyword= <value>` syntax as well. For each procedure call, you can use either the `-keyword <value>` syntax or the `keyword= <value>` syntax. However, within any one procedure call, only one type of syntax can be used. Otherwise, an error message will be generated. Only the new syntax is documented. If you want to continue using the `keyword= <value>` syntax, you also can insert space between the keyword and the equal sign, for example, `keyword = <value>`. Omitting the space between the equal sign and the value field will result in a failure if the value is a de-referenced Tcl variable. Use `keyword= $val` (*not* `keyword=$val`).

The parser accepts arguments in any order. For some arguments, default values are predefined. Such arguments can be omitted. If arguments for which no defaults are

Appendix H: Two-Port Network RF Extraction Library

Syntax Conventions

predefined are omitted, the procedure will exit with an error message. In addition, unrecognized arguments result in an error message.

Instead of using the standard Tcl method of using the return value of the procedure to pass results back to the calling program, the RF extraction library uses a *passing-by-reference* method to return the results to the calling program. The procedure keyword `-out` is used to pass the results back to the calling program:

```
-out <var_name>, <list_name>, or <array_name>
```

The following conventions are used for the syntax of Tcl commands:

- Angle brackets – `<>` – indicate text that must be replaced, but they are *not* part of the syntax. In particular, the following type identifiers are used:
 - `<r>`: Replace with a real number, or a de-referenced Tcl variable that evaluates to a real number. For example: `$val`.
 - `<i>`: Replace with an integer, or a de-referenced Tcl variable that evaluates to an integer. For example: `$i`.
 - `<string>`: Replace with a string, or a de-referenced Tcl variable that evaluates to a string. For example: `$file`.
 - `<list_of_r>`: Replace with a list of real numbers, or a de-referenced Tcl variable that evaluates to a list of real numbers. For example: `$values`.
 - `<list_of_strings>`: Replace with a list of strings, or a de-referenced Tcl variable that evaluates to a list of strings. For example: `$files`.
 - `<var_name>`: Replace with the *name* of a local Tcl variable.
For example: `val` (*not* `$val`).
 - `<list_name>`: Replace with the *name* of a local Tcl list.
For example: `values` (*not* `$values`).
 - `<array_name>`: Replace with the *name* of a local Tcl array.
For example: `myarray` (*not* `$myarray`).
 - `<dataName>`: Replace with the *name* of a dataset.
 - `<fileName>`: Replace with the *name* of a file, or a de-referenced Tcl variable that evaluates to the name of a file.
 - `<plotName>`: Replace with the *name* of a plot.
- Brackets – `[]` – indicate that the argument is optional, but they are *not* part of the syntax.
- A vertical bar – `|` – indicates options, only one of which can be specified.

Help for Procedures

To request help on a specific procedure, in Tcl mode, set the `-help` keyword to 1:

```
rfx::<procedure_name> -help 1
```

If this command is included in a Sentaurus Visual file, when Sentaurus Visual is executed in:

- Batch mode in Sentaurus Workbench, the help information is printed to the runtime output file (with the extension `.out`) of the corresponding Sentaurus Visual node.
- Interactive mode in Sentaurus Workbench, the help information is displayed in the Tcl Console as well as printed in the Sentaurus Visual runtime output file.

You also can enter the command in the Tcl Console of the user interface, in which case, the help information is displayed in the Console.

Output of Procedures

As discussed in [Syntax Conventions on page 501](#), all procedures of the RF extraction library pass the results back to the calling program by storing the results in a Tcl variable. The name of this Tcl variable is specified as the value of the `-out` keyword.

If there are errors in the RF extraction library procedures, the behavior of Sentaurus Visual depends on whether it is executed in batch mode or interactive mode in Sentaurus Workbench. In batch mode, Sentaurus Visual exits and an error message is printed only in the Sentaurus Visual error file (with the extension `.err`). In interactive mode, the error message is displayed in the Tcl Console as well as printed in the Sentaurus Visual runtime error file.

All procedures also print several messages (including warning messages). If Sentaurus Visual is executed in batch mode, the messages are printed only in the Sentaurus Visual output file; whereas, in interactive mode, the messages are displayed in the Tcl Console as well as printed in the Sentaurus Visual runtime output file.

The amount of information printed depends on the information level specified by the procedure `lib::SetInfoDef`.

Overview of RF Extraction Library Procedures

For the simulation of RF characteristics and the extraction of RF parameters, small-signal (AC) analysis is performed in Sentaurus Device by varying the bias at a contact and performing a frequency sweep at each bias point. The Sentaurus Device AC data file contains the conductance values a_{ij} and capacitance values c_{ij} at each bias and frequency

Appendix H: Two-Port Network RF Extraction Library

Overview of RF Extraction Library Procedures

point for all contact-to-contact combinations included in the small-signal analysis (see [A-Matrix, C-Matrix, and Y-Matrix on page 506](#)). The RF extraction library assumes that the transistor can be modeled by a two-port network as shown in [Figure 141](#).

The functionality of the RF extraction library and the corresponding procedures are:

- Loading the Sentaurus Device AC data file, and creating a Y-matrix and PSD matrices:

The Sentaurus Device AC data file is loaded in Sentaurus Visual using the `rfx::Load` procedure, which creates the Tcl array `rfx::AC` containing the conductance and capacitance values (see [rfx::Load on page 547](#)). This data also is converted to admittance or Y-parameters, and the Y-parameters or the Y-matrix are stored in the Tcl array `rfx::Y`. The `rfx::Load` procedure also creates the PSD Tcl arrays (corresponding to PSD matrices) and other variables that are summarized in [Table 37 on page 523](#). For details about these arrays, see [A-Matrix, C-Matrix, and Y-Matrix on page 506](#) and [Power Spectral Density Matrices on page 507](#).

- Converting a Y-matrix to other matrices:

The Y-matrix is converted to either an h-matrix, an S-matrix, or a Z-matrix (see [Matrix Conversions on page 513](#)) using the matrix conversion procedures `rfx::Y2H` (see [rfx::Y2H on page 556](#)), `rfx::Y2S` (see [rfx::Y2S on page 557](#)), and `rfx::Y2Z` (see [rfx::Y2Z on page 558](#)), respectively. All these matrices are complex and the matrix conversion procedures internally use the complex arithmetic procedures (see [Complex Arithmetic Support on page 559](#)). The RF parameters Y_{ij} , h_{ij} , S_{ij} , and Z_{ij} are the elements of the Y-, h-, S-, and Z-matrix, respectively.

- Creating Sentaurus Visual datasets containing small-signal data and noise analysis data:

Sentaurus Visual datasets containing the small-signal data (a_{ij} , c_{ij} , Y_{ij} , h_{ij} , S_{ij} , or Z_{ij}) as a function of frequency or bias can be created using the procedure `rfx::CreateDataset` (see [rfx::CreateDataset on page 526](#)). The datasets corresponding to the RF parameters contain the real and imaginary parts, as well as the absolute value and the phase of the RF parameters. The absolute value of these parameters also can be computed in units of decibel (dB). In addition, Sentaurus Visual datasets containing noise analysis data (S_V^{ij} and S_I^{ij}) can be created using the `rfx::CreateDataset` procedure.

- Plotting small-signal data and noise analysis data:

The datasets created using the `rfx::CreateDataset` procedure can be used to visualize the small-signal data and noise analysis data as a function of frequency or bias. The RF parameters can be visualized using the following types of plot:

- Rectangular plots using Sentaurus Visual commands
- Polar plots using the `rfx::PolarBackdrop` procedure (see [rfx::PolarBackdrop on page 551](#))

Appendix H: Two-Port Network RF Extraction Library

Overview of RF Extraction Library Procedures

- Smith charts using the `rfx::SmithBackdrop` procedure (see [rfx::SmithBackdrop on page 555](#))
- Computing power gains and stability criteria:

Various power gains and stability criteria are computed using S-parameters (see [Gains, Amplifier Stability, and Unilateralization on page 514](#)) using the `rfx::GetPowerGain` procedure (see [rfx::GetPowerGain on page 544](#)). This procedure also can be used to create datasets containing the power gains and stability criteria, either as a function of frequency or bias.

- Extracting transistor figures of merit:

The transistor figures of merit such as cutoff frequency, maximum frequency of oscillation, and cutoff frequency for stability (see [Transistor Figures of Merit on page 516](#)) can be extracted using the procedures `rfx::GetFt` (see [rfx::GetFt on page 536](#)), `rfx::GetFmax` (see [rfx::GetFmax on page 534](#)), and `rfx::GetFK1` (see [rfx::GetFK1 on page 532](#)). These procedures can be used to create the datasets containing the extracted figures of merit as a function of bias. The first two procedures also can be used to create datasets containing the derivative of the gain as a function of frequency.

- Extracting transistor noise parameters:

Transistor noise parameters such as the minimum noise figure, the equivalent noise resistance and conductance, and the optimum source admittance and impedance, along with other noise parameters and noise figures (see [Noise Figure of a Linear Two-Port Network on page 519](#)), can be extracted using the `rfx::GetNoiseFigure` procedure (see [rfx::GetNoiseFigure on page 540](#)). This procedure can create datasets containing the extracted noise parameters as a function of frequency or bias.

- Exporting small-signal data:

The small-signal data can be exported to a CSV file using the `rfx::Export` procedure (see [rfx::Export on page 531](#)).

Appendix H: Two-Port Network RF Extraction Library

Equations Used in RF Extraction Library

Equations Used in RF Extraction Library

This section discusses the equations used in the RF extraction library.

A-Matrix, C-Matrix, and Y-Matrix

The Sentaurus Device AC data file contains the conductance matrix (A-matrix, A) and the capacitance matrix (C-matrix, C) for each bias (v) and frequency (f) point for all contact-to-contact combinations included in the small-signal analysis. The rows and columns of these matrices are given by the nodes included in the small-signal analysis.

For a 3D device, the Sentaurus Device AC data file contains the following for each frequency and bias point:

- a_{ij} , the coefficients or elements of the A-matrix
- c_{ij} , the coefficients or elements of the C-matrix

The A-matrix and C-matrix are converted to an admittance matrix (Y-matrix, Y) using:

$$Y = A + j\omega C = A + jB \quad (36)$$

where:

- j is the imaginary unit.
- $\omega = 2\pi f$ is the angular frequency.
- Matrix B is the susceptance matrix, with coefficients b_{ij} .

The RF extraction library assumes that the transistor can be modeled by a two-port network as shown in [Figure 141](#). Therefore, the RF extraction library reads only a 2×2 matrix, corresponding to a two-port network setup. If other ports are present, they are ignored.

Figure 141 Two-port network schematic



Appendix H: Two-Port Network RF Extraction Library

Equations Used in RF Extraction Library

For a two-port network, the complex Y-matrix at a particular frequency f and bias point v is represented by:

$$Y = \begin{bmatrix} Y_{11}(f, v) & Y_{12}(f, v) \\ Y_{21}(f, v) & Y_{22}(f, v) \end{bmatrix} \quad (37)$$

where the elements of the Y-matrix, Y_{ij} , are the admittance (Y-)parameters. The real and imaginary parts of the complex Y-parameters are given by:

$$\Re(Y_{ij}(f, v)) = a_{ij}(f, v) \quad (38)$$

$$\Im(Y_{ij}(f, v)) = b_{ij}(f, v) = \omega c_{ij}(f, v) \quad (39)$$

Tcl Arrays rfx::AC and rfx::Y

When the Sentaurus Device AC data file is loaded in Sentaurus Visual using the procedure `rfx::Load`, two Tcl arrays `rfx::AC` and `rfx::Y` are created (see [Table 38 on page 528](#)). Both these arrays are two dimensional and have the same form:

```
rfx::AC($ReIm,$P1,$P2,$if,$iv)  
rfx::Y($ReIm,$P1,$P2,$if,$iv)
```

where:

- `ReIm`: 0 (real part) or 1 (imaginary part)
 - For `rfx::AC`, 0 corresponds to $a_{ij}(f, v)$ and 1 corresponds to $c_{ij}(f, v)$.
 - For `rfx::Y`, 0 corresponds to $a_{ij}(f, v)$ and 1 corresponds to $b_{ij}(f, v)$ ([Equation 39](#)).
- `P1, P2`: 1 or 2 (port number)
- `if`: 0-(`rfx::i_freqend`) frequency index
- `iv`: 0-(`rfx::i_biasend`) bias point index

Therefore, the `rfx::AC` array contains the coefficients $a_{ij}(f, v)$ and $c_{ij}(f, v)$ for all frequency and bias points. The `rfx::Y` array contains the coefficients $a_{ij}(f, v)$ and $b_{ij}(f, v)$.

To access the small-signal data or the RF parameter for a given bias or frequency, the appropriate array indices (frequency index and bias point index) must be given.

Power Spectral Density Matrices

The effect of a noisy electronic device on a circuit can be analyzed using either a small-signal model of the device or a two-port network approach. RF extraction library implements noise parameter extraction using the two-port network approach.

Appendix H: Two-Port Network RF Extraction Library

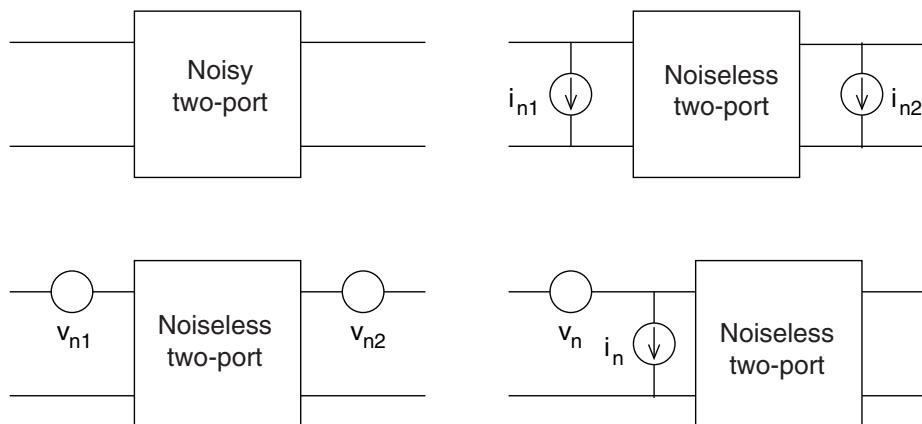
Equations Used in RF Extraction Library

A noisy device such as a transistor can be represented by a two-port network with internal noise sources (see [Figure 142](#)). For small signals, any noisy two-port network can be replaced by a noise equivalent circuit consisting of a noiseless two-port network and two external equivalent noise sources added to the terminals of the two-port (see [Figure 142](#)). The external noise sources are either noise voltage sources or noise current sources, and they produce the same noise voltages at the circuit terminals as the internal noise sources.

Power Spectral Densities

As shown in [Figure 142](#), there are several equivalent representations of a noisy two-port network depending on the type of the external noise sources and their arrangement relative to the noiseless two-port [1].

Figure 142 (Upper left) Two-port network with internal noise sources, and noise equivalent circuits of a two-port network: (upper right) admittance representation, (lower left) impedance representation, and (lower right) chain representation



The most commonly used representations are:

- Impedance representation: Noise voltage sources v_{n1} and v_{n2} are placed in series with the input and output terminals (see [Figure 142](#), lower-left image).
- Admittance representation: Noise current sources i_{n1} and i_{n2} are placed in parallel with the input and output terminals (see [Figure 142](#), upper-right image).
- Chain representation (equivalent input noise representation): Input noise voltage source v_n and input noise current source i_n are placed at the input terminals (see [Figure 142](#), lower-right image).

Appendix H: Two-Port Network RF Extraction Library

Equations Used in RF Extraction Library

The noise sources are characterized by a mean square value and a PSD:

- A noise voltage source v_n is characterized by the mean square value $\overline{v_n^2}$ and the NVSD S_{v_n} .
- A noise current source i_n is characterized by the mean square value $\overline{i_n^2}$ and the NISD S_{i_n} .

The NVSD determines the mean square value of the noise voltage source, and the NISD determines the mean square value of the current voltage source within a frequency interval of width 1 Hz [2]. Therefore, the units of NVSD are V^2/Hz or $V^2\text{s}$, and the units of NISD are A^2/Hz or $A^2\text{s}$. The PSD gives the average power P_{av} that the noise source contributes in a 1 Hz bandwidth around frequency f . The PSD spectrum shows how much average power the noise source contributes at each frequency [3].

For a noise voltage source, its average power and mean square voltage over a frequency interval $[f_1, f_2]$ are related to its NVSD by:

$$P_{\text{av}} = \overline{v_n^2} = \int_{f_1}^{f_2} S_{v_n} df \quad (40)$$

For a contact pair (i, j) (or for nodes i and j), the NVSD is denoted by S_V^{ij} , and the NISD is denoted by S_I^{ij} . Autocorrelation PSD corresponds to the case when both terminals are the same ($i = j$); whereas, the cross-correlation PSD corresponds to the case when both terminals are different ($i \neq j$).

The following PSDs can be defined for the various representations of the noisy two-port (see [PSDs Computed by Sentaurus Device and RF Extraction Library on page 510](#)):

- S_V^{11} : Noise voltage spectral density at the input port in the impedance representation.
- S_V^{22} : Noise voltage spectral density at the output port in the impedance representation.
- S_V^{12} and S_V^{21} : Cross-correlation spectral density between the input and output voltage noise sources (impedance representation).
- S_I^{11} : Noise current spectral density at the input port in the admittance representation.
- S_I^{22} : Noise current spectral density at the output port in the admittance representation.
- S_I^{12} and S_I^{21} : Cross-correlation spectral density between the input and output current noise sources (admittance representation).
- S_{v_n} : Noise voltage spectral density of the input noise voltage source v_n in the chain representation.
- S_{i_n} : Noise current spectral density of the input noise current source i_n in the chain representation.
- $S_{v_n i_n}$ and $S_{i_n v_n}$: Cross-correlation spectral density of the equivalent input noise voltage and noise current source (chain representation).

Appendix H: Two-Port Network RF Extraction Library

Equations Used in RF Extraction Library

The NISDs in the admittance representation are used to compute the noise correlation coefficient between the input and output noise current sources C_{i_n} :

$$C_{i_n} = \frac{S_I^{12}}{\sqrt{S_I^{11} S_I^{22}}} \quad (41)$$

PSDs Computed by Sentaurus Device and RF Extraction Library

As a result of noise analysis on a two-port network containing nodes i ($i = 1$, input port) and j ($j = 2$, output port), Sentaurus Device computes the autocorrelation and cross-correlation spectral densities for both impedance representation (S_V^{11} , S_V^{21} , and S_V^{22}) and admittance representation (S_I^{11} , S_I^{21} , and S_I^{22}), and writes them in the Sentaurus Device AC data file.

The RF extraction library computes the spectral densities S_V^{12} and S_I^{12} , as well as the spectral densities characterizing the external noise sources in the chain representation (S_{v_n} , S_{i_n} , $S_{v_n i_n}$, and $S_{i_n v_n}$) (see [rfx::GetNoiseFigure on page 540](#)).

The RF extraction library converts the PSD data in the AC data file to PSD Tcl arrays using the `rfx::Load` procedure (see [rfx::Load on page 547](#)) and to Sentaurus Visual datasets using the `rfx::CreateDataset` procedure (see [rfx::CreateDataset on page 526](#)). [Table 40 on page 529](#) gives examples of PSD matrix coefficients and the corresponding names of PSD variables in the AC data file, the PSD Tcl array name, and the dataset variable name.

The NISDs are computed for the current through the selected circuit nodes, assuming a fixed voltage at these nodes. The NVSDs are computed for the voltages at these nodes, assuming the net current to these nodes is fixed [4].

The NISD is saved as `s_I` and the NVSD is saved as `s_v` in the AC data file. In the case of the autocorrelation coefficient for node i , the NISD is denoted by `s_I(i)`. The cross-correlation coefficients have both real and imaginary parts. The real part of the NISD for nodes i and j is denoted by `ReS_Ixi(i, j)`. The imaginary part is denoted by `ImS_Ixi(i, j)`. Similar conventions apply to the NVSD.

In addition to the previously mentioned NISD and NVSD, Sentaurus Device writes several partial noise spectral densities that describe the contribution of specific noise sources.

For example, `s_v_ee` is the NVSD due to electrons and `s_v_eediff` is the electron NVSD due to diffusion noise. For a list of all these spectral densities, see the *Sentaurus™ Device User Guide* [4].

Power Spectral Density Tcl Arrays

For the impedance representation of a noisy two-port network (see [Power Spectral Densities on page 508](#)), the complex S_V -matrix (NVSD matrix) at a particular frequency f and bias point v is represented by:

$$S_V = \begin{bmatrix} S_V^{11}(f, v) & S_V^{12}(f, v) \\ S_V^{21}(f, v) & S_V^{22}(f, v) \end{bmatrix} \quad (42)$$

Similarly, the complex S_I -matrix (NISD matrix) at a particular frequency f and bias point v is represented by:

$$S_I = \begin{bmatrix} S_I^{11}(f, v) & S_I^{12}(f, v) \\ S_I^{21}(f, v) & S_I^{22}(f, v) \end{bmatrix} \quad (43)$$

The coefficients $S_V^{ij}(f, v)$ and $S_I^{ij}(f, v)$ are defined in [Power Spectral Densities on page 508](#).

When the Sentaurus Device AC data file is loaded in Sentaurus Visual using the `rfx::Load` procedure, the PSD Tcl arrays (see [Table 38 on page 528](#)) are also created if the file contains PSD data. A Tcl array is created for each PSD data stored in the AC data file. For example, the Tcl arrays `rfx::SV` and `rfx::SI` are created and contain the coefficients $S_V^{ij}(f, v)$ and $S_I^{ij}(f, v)$ (see [Table 40 on page 529](#)).

The form of these PSD Tcl arrays is the same as the Tcl arrays `rfx::AC` and `rfx:::Y`. For example, the arrays `rfx::SV` and `rfx::SI` have the form:

```
rfx::SV($ReIm,$P1,$P2,$if,$iv)
rfx::SI($ReIm,$P1,$P2,$if,$iv)
```

where:

- For `rfx::SV`, 0 corresponds to $\Re(S_V^{ij}(f, v))$ and 1 corresponds to $\Im(S_V^{ij}(f, v))$ ([Equation 42](#)).
- For `rfx::SI`, 0 corresponds to $\Re(S_I^{ij}(f, v))$ and 1 corresponds to $\Im(S_I^{ij}(f, v))$ ([Equation 43](#)).

Each PSD array contains the autocorrelation coefficients (both ports are the same, $i = j$) as well as the cross-correlation coefficients (both ports are different, $i \neq j$). Since the autocorrelation values are real, the imaginary part is set to 0.

The PSD arrays for the local noise source (LNS) are created depending on the specific noise models activated in the Sentaurus Device command file. For example, `rfx::SVeeDiff` is created only if the diffusion LNS is specified in the Sentaurus Device command file.

Appendix H: Two-Port Network RF Extraction Library

Equations Used in RF Extraction Library

If there is a named noise specification in the Sentaurus Device command file, this name is prefixed to the name of the PSD Tcl array: `<name>_rfx::SV`. For example, if the name of the noise specification is `diff`, examples of array names are `diff_rfx::SVeeDiff` and `diff_rfx::SV`.

Note:

Only one noise specification is supported per simulation. It can be either named or unnamed.

Device Width Scaling for 2D Structures

A 3D device homogeneous in one of the directions (for example, the z-direction) can be analyzed by using a two-dimensional (2D) device structure. In this case, device simulation can be performed on the 2D device structure and the results for the 3D device can be obtained by multiplying the 2D simulation results (terminal currents, conductance, and capacitance) by the device width in the z-direction, W , and in [Equation 36](#), you replace:

$$A = WA^{2D} \quad (44)$$

$$B = WB^{2D} \quad (45)$$

where A^{2D} and B^{2D} are the conductance matrix and the susceptance matrix of the 2D device, respectively.

W can be set in one of the following ways:

- In the Sentaurus Device input file using the keyword `AreaFactor` in the `Physics` section.
- During postprocessing using the `rfx::Load` procedure by specifying the `-devicewidth` keyword (see [rfx::Load on page 547](#)).

The default value of `AreaFactor` as well as the `-devicewidth` keyword is 1 μm .

Note:

Avoid applying the scaling twice by using only one of the scaling methods.

Some of the parameters such as h_{21} scale trivially with the device width. For other parameters such as S-parameters or the unilateral figure of merit, the device width is important.

Note:

Not all RF quantities scale linearly with the device width. You must use the keyword `AreaFactor` in the Sentaurus Device command file to take into account the device width scaling for 2D structures.

Matrix Conversions

As discussed in [Overview of RF Extraction Library Procedures on page 503](#), the Y-matrix is converted to either an h-matrix, an S-matrix, or a Z-matrix using the matrix conversion formulas summarized here [5][6].

Converting Y-Matrix to h-Matrix

The complex Y-matrix is converted to the complex h-matrix using the formulas:

$$\begin{aligned} h_{11} &= \frac{1}{Y_{11}} \\ h_{12} &= \frac{-Y_{12}}{Y_{11}} \\ h_{21} &= \frac{Y_{21}}{Y_{11}} \\ h_{22} &= \frac{D_y}{Y_{11}} \end{aligned} \tag{46}$$

with $D_y = Y_{11}Y_{22} - Y_{12}Y_{21}$.

Converting Y-Matrix to S-Matrix

The complex Y-matrix is converted to the complex S-matrix using the formulas:

$$\begin{aligned} S_{11} &= \frac{(1 - \bar{Y}_{11})(1 + \bar{Y}_{22}) + \bar{Y}_{12}\bar{Y}_{21}}{N_y} \\ S_{12} &= \frac{-2\bar{Y}_{12}}{N_y} \\ S_{21} &= \frac{-2\bar{Y}_{21}}{N_y} \\ S_{22} &= \frac{(1 - \bar{Y}_{22})(1 + \bar{Y}_{11}) + \bar{Y}_{12}\bar{Y}_{21}}{N_y} \end{aligned} \tag{47}$$

with:

$$\bar{Y}_{ij} = Z_o Y_{ij} \tag{48}$$

where Z_o is the characteristic impedance and $N_y = (1 + \bar{Y}_{11})(1 + \bar{Y}_{22}) - \bar{Y}_{12}\bar{Y}_{21}$.

Converting Y-Matrix to Z-Matrix

The complex Y-matrix is converted to the complex Z-matrix using the formulas:

$$\begin{aligned} Z_{11} &= \frac{Y_{22}}{D_y} \\ Z_{12} &= \frac{-Y_{12}}{D_y} \\ Z_{21} &= \frac{-Y_{21}}{D_y} \\ Z_{22} &= \frac{Y_{11}}{D_y} \end{aligned} \quad (49)$$

with $D_y = Y_{11}Y_{22} - Y_{12}Y_{21}$.

Gains, Amplifier Stability, and Unilateralization

This section discusses gains, amplifier stability, and unilateralization.

Small-Signal Current Gain

The short-circuit small-signal current gain h_{21} of a transistor is given by:

$$h_{21} = \left| \frac{Y_{21}}{Y_{11}} \right| \quad (50)$$

Amplifier Stability

The Rollett stability factor K is computed from the S-parameters using the formula [6][7]:

$$K = \frac{1 - |S_{11}|^2 - |S_{22}|^2 + |\Delta|^2}{2|S_{12} \cdot S_{21}|} \quad (51)$$

where:

$$|\Delta| = |S_{11} \cdot S_{22} - S_{12} \cdot S_{21}| \quad (52)$$

The necessary and sufficient conditions for unconditional stability for an amplifier are [6]:

$$\begin{aligned} K &> 1 \\ |\Delta| &< 1 \end{aligned} \quad (53)$$

Unconditional stability indicates conjugate matching between output and input loads. For $K < 1$, an amplifier is conditionally stable or potentially unstable and must be stabilized.

Maximum Stable Gain and Maximum Available Gain

The maximum stable gain (MSG) G_{ms} of a two-port network is given by [7][8]:

$$G_{ms} = \left| \frac{S_{21}}{S_{12}} \right| \quad (54)$$

The maximum available gain (MAG) G_{ma} depends on the stability of the two-port network. For an unconditionally stable two-port network, that is, if both $K > 1$

and $|\Delta| < 1$:

$$G_{ma}(K > 1, |\Delta| < 1) = G_{ms} \cdot (K - \sqrt{K^2 - 1}) \quad (55)$$

For $K \leq 1$ or $|\Delta| > 1$, MAG is set to MSG [9]:

$$G_{ma}(K \leq 1, |\Delta| > 1) = G_{ms} \quad (56)$$

Unilateral Amplifier Design

Mason's unilateral gain (MUG) U is computed from the S-parameters using the formula [8]:

$$U = \frac{\left| \frac{S_{21}}{S_{12}} - 1 \right|^2}{2K \left| \frac{S_{21}}{S_{12}} \right|^2 - 2 \cdot \Re(\frac{S_{21}}{S_{12}})} \quad (57)$$

where $\Re(z)$ denotes the real part of the complex number z .

The unilateral figure of merit U_f is given by [6]:

$$U_f = \frac{|S_{12}| |S_{21}| |S_{22}| |S_{11}|}{(1 - |S_{11}|^2)(1 - |S_{22}|^2)} \quad (58)$$

For a unilateral amplifier design approach, U_f must be as small as possible.

Converting Gain Units to Decibels

The short-circuit current gain, $|h_{21}|$, can be expressed in units of decibel (dB) using:

$$|h_{21}|[\text{dB}] = 20 \log |h_{21}| \quad (59)$$

The power gain, P , is expressed in units of dB using:

$$P[\text{dB}] = 10 \log |P| \quad (60)$$

Transistor Figures of Merit

This section discusses transistor figures of merit.

ft and fmax

The frequency dependency of the magnitude of the current gain $|h_{21}|$ is given by [2]:

$$|h_{21}| \approx \frac{\beta}{\sqrt{1 + \left(\frac{f}{f_\beta}\right)^2}} \quad (61)$$

where β is the current gain at low frequency, and f_β is the β cutoff frequency or the 3 dB frequency.

The short-circuit current gain cutoff frequency or the cutoff frequency f_t is defined as the frequency at which $|h_{21}| = 1$ (unit gain point):

$$f_t \equiv f(|h_{21}| = 1) = 0[\text{dB}] \quad (62)$$

f_t is related to f_β :

$$f_t = \beta f_\beta \quad (63)$$

[Equation 61](#) shows that for $f \ll f_\beta$ (low frequencies):

$$|h_{21}| \approx \beta \quad (64)$$

and for $f \gg f_\beta$ (high frequencies):

$$|h_{21}| \approx \frac{f_t}{f} \quad (65)$$

Converting the unit of $|h_{21}|$ to dB (using [Equation 59](#)), [Equation 65](#) can be written as:

$$|h_{21}|[\text{dB}] = 20\log f_t - 20\log f \quad (66)$$

Therefore, the $|h_{21}|$ (in units of dB) versus $\log f$ curve (current gain curve) is flat at low frequencies, reduces by 3 dB at f_β , and falls off linearly with a slope of -20 dB/decade with increasing frequencies.

Let $(\log f_0, |h_{21,0}|[\text{dB}])$ be a high frequency point at which the -20 dB/decade slope is fully established. From [Equation 66](#):

$$20\log \frac{f_t}{f_0} = |h_{21,0}|[\text{dB}] \quad (67)$$

or:

$$f_t = f_0 \cdot 10^{|h_{21,0}|[\text{dB}]/20} \quad (68)$$

Appendix H: Two-Port Network RF Extraction Library

Equations Used in RF Extraction Library

Therefore, [Equation 66](#) implies that f_t can also be determined by linear extrapolation from a high frequency point ($\log f_0, |h_{21,0}|[\text{dB}]$) on the current gain curve, using [Equation 68](#) [9].

The frequency dependency of MUG or MAG at high frequencies is given by [9]:

$$G \approx \frac{f_{\max}^2}{f^2} \quad (69)$$

where G is either MUG or MAG, and f_{\max} is the maximum frequency of oscillation. f_{\max} can be extracted using either MUG or MAG [9].

f_{\max} is defined as the frequency at which $U = 1$ (unit gain point):

$$f_{\max} \equiv f(U = 1 = 0[\text{dB}]) \quad (70)$$

or the frequency at which $G_{\max} = 1$:

$$f_{\max} \equiv f(G_{\max} = 1 = 0[\text{dB}]) \quad (71)$$

f_{\max} is the maximum frequency at which power gain can be extracted from an amplifier. It is also the maximum frequency of oscillation of an oscillator made from an amplifier with power gain. If $U > 1$, the transistor is active and f_{\max} is extracted. If $U \leq 1$, the transistor is passive and f_{\max} is not extracted [10].

Similar to f_t , f_{\max} also can be determined by linear extrapolation from a point ($\log f_0, U_0[\text{dB}]$) on the power gain curve (U in units of dB versus $\log f$) with a slope of -20 dB/decade , using [2]:

$$f_{\max} = f_0 \cdot 10^{(U_0[\text{dB}])/20} \quad (72)$$

A similar equation is used to extract f_{\max} from a G_{\max} versus f curve:

$$f_{\max} = f_0 \cdot 10^{(G_{\max,0}[\text{dB}])/20} \quad (73)$$

Extraction Methods for f_t and f_{\max}

f_t and f_{\max} are extracted from the corresponding gain curves by the following RF extraction library procedures:

- `rfx::GetFt` extracts f_t from the $|h_{21}|$ versus frequency curve.
- `rfx::GetFmax` extracts f_{\max} from the U versus frequency curve, or the G_{\max} versus frequency curve. For brevity, either of these power gains is denoted by G .

Both procedures use three different methods of extraction: `unit-gain-point`, `extract-at-dBPoint`, and `extract-at-frequency`. The last two methods are extrapolation methods and assume the ideal frequency dependency of gain (flat at low frequencies, and falling off linearly at higher frequencies with a slope of -20 dB/decade). None of these methods checks the validity of these assumptions.

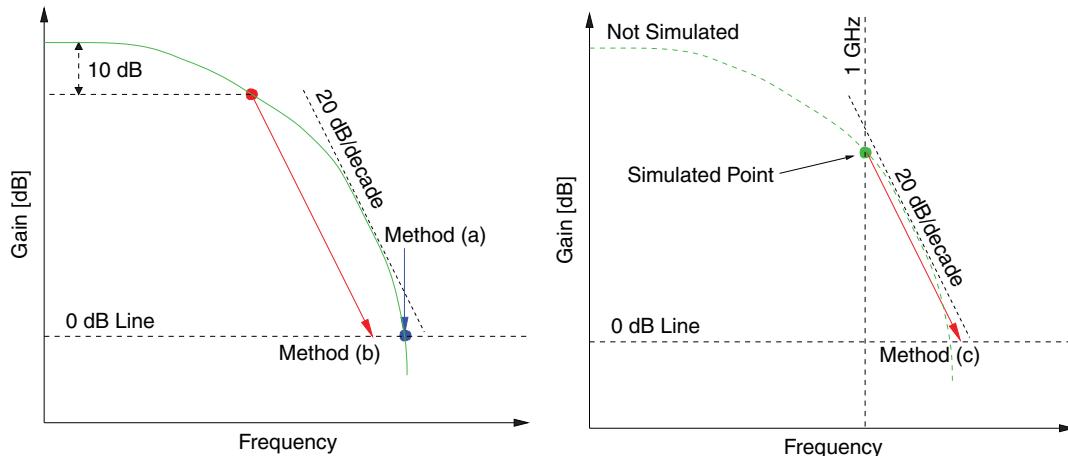
Appendix H: Two-Port Network RF Extraction Library

Equations Used in RF Extraction Library

The extraction methods are (see Figure 143):

- (a) The *unit gain point method* uses the definition of f_t (Equation 62) and f_{\max} (Equation 70 or Equation 71). It searches directly for the unit gain point but might give inappropriate results if the gain curves deviate from the -20 dB/decade slope near the unit gain point.
- (b) The *extract at dB point method* looks for the gain point ($(\log f_0, |h_{21,0}|[\text{dB}])$ or $(\log f_0, G_0[\text{dB}])$) where the gain ($|h_{21}|$ or G) has fallen by a certain number of decibels from its value at the start of the gain curve. This difference in decibels is called the *dB point* and is specified using the keyword `-parameter` in units of dB. Assuming a -20 dB/decade slope, the gain point ($(\log f_0, |h_{21,0}|[\text{dB}])$ or $(\log f_0, G_0[\text{dB}])$) is used to compute f_t using Equation 68 or f_{\max} using Equation 72 or Equation 73. This method might give inappropriate results if the -20 dB/decade slope is not fully established at the gain point. Often, the results can be improved by adjusting the dB point.
- (c) The *extract at frequency method* is the same as method (b), but the frequency corresponding to the unit gain point is extrapolated from a specified frequency f_0 . The frequency f_0 is specified using the keyword `-parameter`. The corresponding gain ($|h_{21,0}|[\text{dB}]$ or $G_0[\text{dB}]$) is computed, and f_t or f_{\max} is computed using Equation 68 or Equation 72 and Equation 73. This method might give inappropriate results if the -20 dB/decade slope is not fully established at this gain point. The optimal frequency for this method can depend on the control bias or current, and might be different for G and $|h_{21}|$.

Figure 143 Different extraction methods and the circumstances under which they might return inappropriate results



Unfortunately, no single extraction method of f_t and f_{\max} is appropriate under all circumstances. Technically, method (a), the direct search for the unit gain point, should be the most appropriate method. However, at high frequencies, additional parasitic elements might become dominant and alter the -20 dB/decade slope near the unit gain point. Furthermore, in experiments, it is sometimes difficult to trace the gain curve to high-enough frequencies to see the unit gain point directly. Therefore, extrapolation of experimental data

Appendix H: Two-Port Network RF Extraction Library

Equations Used in RF Extraction Library

to the unit gain point is common. In this case, the experiment might not ‘see’ the altered slope due to the parasitics (also some parasitics might not be included in the simulation). For a comparison with experimental results, therefore, method (b) or method (c) might be better.

The extraction methods assume that, for each value of the control bias or current, a full frequency sweep is performed. Ideally, this sweep should start at a frequency where the gain is flat (low-frequency regime) and should end beyond the unit gain point. If the frequency sweep does not go beyond the unit gain point, method (a) sets the value of f_t to zero. If the frequency sweep does not start in the flat region, methods (b) and (c) still return a (nonzero) value. However, you must ensure that, at the selected dB point (for method (b)) or the frequency point (for method (c)), the –20 dB/decade slope is established.

The transition between the flat low-frequency region of the gain curves to the –20 dB/decade slope at higher frequencies can be wide. Sometimes, a clear –20 dB/decade slope is never reached. In this case, methods (b) and (c) might give incorrect results (often, the results can be improved by adjusting the dB point used for the extrapolation).

The simulation of a full frequency sweep in Sentaurus Device can be time consuming, especially for large structures and if many equations are solved. If it is known beforehand that, at a given frequency, the slope of the gain curves is –20 dB/decade, it is sufficient to simulate the small-signal response at this single frequency only and to apply method (c). However, the band of frequencies for which the –20 dB/decade slope assumption holds true can be very narrow and can depend on the bias conditions.

This discussion shows that using solely one method might give inappropriate results. Therefore, it is recommended to always use all three methods concurrently. If the f_t or f_{\max} curves for all three methods agree well, the results can be trusted with a high level of confidence. If the results are very different, most likely, the form of the gain curve prevents a meaningful automatic extraction of f_t and f_{\max} . In this case, it is suggested to examine the underlying gain curves and the slope of the gain curves. In most such cases, it is clear that the assumptions on which the extractions are based are not fulfilled. That is, the gain curve does not fall off with a clean –20 dB/decade slope at high frequencies.

Cutoff Frequency for Stability

The cutoff frequency for stability f_{K1} is defined as the frequency point at which $K = 1$ (the boundary between the unconditionally stable and conditionally stable region):

$$f_{K1} \equiv f(K = 1) \quad (74)$$

Noise Figure of a Linear Two-Port Network

The noise factor F of a linear two-port network is defined as the signal-to-noise ratio at the input port divided by the signal-to-noise ratio at the output port.

In the RF extraction library, the y-parameters of the noisy two-port network and the PSDs of the admittance representation are used to compute the PSDs of the chain representation,

Appendix H: Two-Port Network RF Extraction Library

Equations Used in RF Extraction Library

which are then used to compute the noise figure and the noise parameters of the two-port network.

As discussed in [Power Spectral Densities on page 508](#), in the chain representation, the noisy two-port network consists of two noise sources, v_n and i_n , placed at the input port. The two-port is driven by a sinusoidal source of either internal admittance Y_s (source admittance) or internal impedance Z_s (source impedance). The noise figure NF ([Equation 94](#)) of the noisy two-port network is the noise factor F expressed in units of dB ([Equation 92](#)). It is determined by the source admittance and the noise parameters of the two-port [\[9\]](#). The noise parameters are:

- Minimum noise figure, NF_{\min} ([Equation 93](#))
- Equivalent noise resistance R_n of the noise voltage source ([Equation 76](#))
- Optimum source admittance Y_{opt} ([Equation 87](#) and [Equation 88](#))

The optimum source admittance is the value of the source admittance at which the noise figure has its minimum value NF_{\min} . R_n determines the sensitivity of the noise figure to deviations from Y_{opt} .

An alternative set of noise parameters is [\[11\]](#):

- Minimum noise figure, NF_{\min} ([Equation 93](#))
- Equivalent noise conductance G_n of the noise voltage source ([Equation 82](#))
- Optimum source impedance Z_{opt} ([Equation 90](#))

The above admittances and impedances can be normalized by dividing by the characteristic impedance Z_o , and these values are called normalized admittances and impedances (r_n , y_{opt} , g_n , and z_{opt}).

The equations used to compute the noise figure, the noise parameters, and various other quantities using the `rfx::NoiseFigure` procedure (see [rfx::NoiseFigure on page 549](#)) are discussed here [\[2\]](#)[\[11\]](#)[\[12\]](#)[\[13\]](#)[\[14\]](#).

The spectral density of the equivalent input noise voltage source S_{v_n} is computed using:

$$S_{v_n} = \frac{S_I^{22}}{|Y_{21}|} \quad (75)$$

The equivalent noise resistance R_n of the noise voltage source is computed using:

$$R_n = \frac{S_{v_n}}{4k_B T_o} \quad (76)$$

where:

- k_B is the Boltzmann constant.

Appendix H: Two-Port Network RF Extraction Library

Equations Used in RF Extraction Library

- $T_o = 290\text{ K}$ is the standard noise temperature.

The normalized equivalent noise resistance r_n is computed using:

$$r_n = \frac{R_n}{Z_o} \quad (77)$$

The equivalent noise conductance G_u and the normalized equivalent noise conductance g_u of the uncorrelated noise current component are given by:

$$G_u = \frac{1}{4k_B T_o} \left(S_I^{11} - \frac{|S_I^{12}|^2}{S_I^{22}} \right) \quad (78)$$

$$g_u = \frac{G_u}{Z_o} \quad (79)$$

The correlation admittance Y_{cor} is given by:

$$Y_{\text{cor}} = G_{\text{cor}} + jB_{\text{cor}} = Y_{11} - Y_{21} \frac{S_I^{12}}{S_I^{22}} \quad (80)$$

where:

- G_{cor} is the correlation conductance.
- B_{cor} is the correlation susceptance.

The spectral density of the equivalent input noise current source S_{i_n} is given by:

$$S_{i_n} = 4k_B T_o (|Y_{\text{cor}}|^2 R_n + G_u) \quad (81)$$

The equivalent noise conductance G_n and the normalized equivalent noise conductance g_n of the input noise current source are computed using:

$$\begin{aligned} G_n &= \frac{S_{i_n}}{4k_B T_o} \\ g_n &= \frac{G_n}{Z_o} \end{aligned} \quad (82)$$

The cross-correlation spectral densities $S_{v_n \bar{i}_n}$ and $S_{i_n v_n}$ of the equivalent input noise voltage and noise current sources are given by:

$$\begin{aligned} S_{v_n \bar{i}_n} &= \overline{Y_{\text{cor}} S_{v_n}} \\ S_{i_n v_n} &= \overline{S_{v_n \bar{i}_n}} \end{aligned} \quad (83)$$

Appendix H: Two-Port Network RF Extraction Library

Equations Used in RF Extraction Library

The noise correlation coefficient between the equivalent input noise voltage and noise current source is computed using:

$$C_{i_n v_n} = \frac{S_{i_n v_n}}{\sqrt{S_{i_n} S_{v_n}}} \quad (84)$$

The source admittance and the source impedance are defined as:

$$\begin{aligned} Y_s &= G_s + jB_s \\ Z_s &= R_s + jX_s = \frac{1}{Y_s} \end{aligned} \quad (85)$$

where:

- G_s is the source conductance.
- B_s is the source susceptance.
- R_s is the source resistance.
- X_s is the source reactance.

The optimum source admittance Y_{opt} is defined as:

$$Y_{\text{opt}} = G_{\text{opt}} + jB_{\text{opt}} \quad (86)$$

The optimum source conductance G_{opt} is computed using:

$$G_{\text{opt}} = \sqrt{\frac{G_u + R_n G_{\text{cor}}^2}{R_n}} \quad (87)$$

The optimum source susceptance B_{opt} is computed using:

$$B_{\text{opt}} = -B_{\text{cor}} \quad (88)$$

The normalized optimum source admittance y_{opt} is computed using:

$$y_{\text{opt}} = \frac{Y_{\text{opt}}}{Z_o} \quad (89)$$

The optimum source impedance Z_{opt} and the normalized optimum source impedance z_{opt} are defined as:

$$\begin{aligned} Z_{\text{opt}} &= R_{\text{opt}} + jX_{\text{opt}} = \frac{1}{Y_{\text{opt}}} \\ z_{\text{opt}} &= \frac{Z_{\text{opt}}}{Z_o} \end{aligned} \quad (90)$$

The minimum noise factor F_{min} is given by:

$$F_{\text{min}} = 1 + 2R_n(G_{\text{cor}} + G_{\text{opt}}) \quad (91)$$

Appendix H: Two-Port Network RF Extraction Library

rfx Namespace Variables

and the noise factor F is computed using:

$$F = F_{\min} + \frac{R_n}{G_s} [(G_s - G_{\text{opt}})^2 + (B_s - B_{\text{opt}})^2] \quad (92)$$

The minimum noise figure NF_{\min} [dB] is computed using:

$$NF_{\min} = 10 \log_{10} F_{\min} \quad (93)$$

The noise figure NF [dB] is computed using:

$$NF = 10 \log_{10} F \quad (94)$$

rfx Namespace Variables

Many RF extraction library procedures use the variables summarized in [Table 37](#). These variables are created by the `rfx::Load` procedure.

Note:

If there is a named noise specification in the Sentaurus Device command file, this name is prefixed to the name of the PSD Tcl array, for example, `<name>_rfx::SV`.

Table 37 rfx namespace variables

Variable name	Description
<code>rfx::AC</code>	AC array (see A-Matrix, C-Matrix, and Y-Matrix on page 506).
<code>rfx::Y</code>	Y-matrix (see A-Matrix, C-Matrix, and Y-Matrix on page 506).
<code>rfx::nfreq</code>	Number of frequencies.
<code>rfx::freq</code>	List of frequencies.
<code>rfx::i_freqstart</code>	Index of the first element in the list of frequencies, <code>rfx::freq</code> .
<code>rfx::i_freqend</code>	Index of the last element in the list of frequencies, <code>rfx::freq</code> .
<code>rfx::nbias</code>	Number of bias points.
<code>rfx::bias</code>	List of bias points.
<code>rfx::i_biasstart</code>	Index of the first element in the list of bias points, <code>rfx::bias</code> .
<code>rfx::i_biasend</code>	Index of the last element in the list of bias points, <code>rfx::bias</code> .
<code>rfx::z0</code>	Characteristic impedance in units of Ω . Default: 50 Ω .

Appendix H: Two-Port Network RF Extraction Library

rfx Namespace Variables

Table 37 rfx namespace variables (Continued)

Variable name	Description
<code>rfx:::zs</code>	Source impedance [Ω] seen by the two-port network. Default: [$\$rfx:::z0\ 0$]
<code>rfx:::T0</code>	Standard noise temperature.
Noise or power spectral density arrays	
<code>rfx:::SV</code>	NVSD matrix (S_V^{11} , S_V^{12} , S_V^{21} , and S_V^{22}).
<code>rfx:::SI</code>	NISD matrix (S_I^{11} , S_I^{12} , S_I^{21} , and S_I^{22}).
Partial noise spectral density arrays	
<code>rfx:::SVee</code>	Electron NVSD matrix.
<code>rfx:::SVhh</code>	Hole NVSD matrix.
<code>rfx:::SIEe</code>	Electron NISD matrix.
<code>rfx:::SIhh</code>	Hole NISD matrix.
<code>rfx:::SVeeDiff</code>	Electron NVSD matrix due to diffusion LNS.
<code>rfx:::SVhhDiff</code>	Hole NVSD matrix due to diffusion LNS.
<code>rfx:::SVeeMonoGR</code>	Electron NVSD matrix due to monopolar generation–recombination (GR) LNS.
<code>rfx:::SVhhMonoGR</code>	Hole NVSD matrix due to monopolar GR LNS.
<code>rfx:::SVeeFlickerGR</code>	Electron NVSD matrix due to flicker GR LNS.
<code>rfx:::SVhhFlickerGR</code>	Hole NVSD matrix due to flicker GR LNS.

Characteristic Impedance and Source Impedance

The RF extraction library uses the characteristic impedance Z_o to:

- Convert Y-parameters to S-parameters (see [Equation 47 on page 513](#) and [rfx::Y2S on page 557](#)).
- Compute the normalized values of various noise parameter-related admittances and impedances (see [Noise Figure of a Linear Two-Port Network on page 519](#)).

The RF extraction library also uses the source impedance Z_s to compute the noise figure (see [Equation 94 on page 523](#) and [rfx::NoiseFigure on page 549](#)).

In the RF extraction library, Z_o and Z_s are represented by the `rfx::z0` and `rfx::zs` variables, respectively (see [Table 37 on page 523](#)).

Z_o defaults to 50Ω . To change the characteristic impedance to 100Ω , for example, use the following command *after* loading the RF extraction library:

```
set rfx::z0 100.0
```

Z_s defaults to $50 + j0 \Omega$. To change the source impedance to $100 + j0 \Omega$, for example, use the following command *after* loading the RF extraction library:

```
set rfx::zs [list 100.0 0.0]
```

rfx::CreateDataset

Creates a Sentaurus Visual dataset corresponding to an RF matrix or a PSD matrix as a function of frequency or bias.

Syntax

```
rfx::CreateDataset -dataset <dataName>
  [-rfmatrix "AC" | "Y" | "H" | "Z" | "S" | "SV" | "SI"]
  [-dB 0 | 10 | 20]
  [-noisename <string>] [-xaxis "frequency" | "bias"]
  [-info 0 | 1 | 2 | 3] [-help 0 | 1]
```

Argument	Description
-dataset <dataName>	Name of a Sentaurus Visual dataset. The dataset contains variables for all the RF parameters in the RF matrix specified using the keyword <code>-rfmatrix</code> , as a function of frequency for each bias point (<code>-xaxis "frequency"</code>) or bias for each frequency (<code>-xaxis "bias"</code>). For <code>-xaxis "frequency"</code> , the variables are created for all the bias point indices (0 to <code>rfx::i_biasend</code>). For <code>-xaxis "bias"</code> , the variables are created for all the frequency point indices (0 to <code>rfx::i_freqend</code>). For example, for <code>-rfmatrix "Y"</code> and <code>-xaxis "frequency"</code> , the dataset contains the Y-parameters as a function of frequency for each bias point. Therefore, the variables " <code>bias_<i> y<ij>_Re</code> " and " <code>bias_<i> frequency</code> " are created. Here, <code>i</code> is the bias point index and <code>ij</code> refers to the port numbers (11, 12, 21, 22). If <code>-dB 10</code> (or 20) is specified, 10dB (or 20dB) is appended to the name of the variable corresponding to the absolute value of the RF parameter. The variables are summarized in Table 38 on page 528 . Similar variables are created for other RF parameters. For <code>-rfmatrix "SV"</code> or <code>-rfmatrix "SI"</code> , variables for all the power spectral densities are created. These are summarized in Table 39 on page 528 . In addition, the variables for all of the Y-parameters are created. (String, no default)
-rfmatrix "AC" "Y" "H" "Z" "S" "SV" "SI"	Name of the RF or PSD matrix. (String, default: "AC")
-dB 0 10 20	Decibel level for computing the absolute value of an RF parameter. For <code>-dB 0</code> , the absolute value is computed on linear scale. Default: 0

Appendix H: Two-Port Network RF Extraction Library

rfx::CreateDataset

Argument	Description
-noisename <string>	Name of the noise specification. Required only for -rfmatrix "SV" or -rfmatrix "SI", and in the case when a named noise specification was used to perform noise analysis. (String, default: "")
-xaxis "frequency" "bias"	Selects the x-axis as either frequency or bias. Selects the sorting order. For example, for -xaxis "frequency", the data is created using a loop with frequency as the <i>inner variable</i> and bias as the <i>outer variable</i> . (String, default: "frequency")
-info 0 1 2 3	Sets local information level. Default: 0
-help 0 1	Prints a help screen if set to 1. Default: 0

Returns

None.

Example

```
# Ex 1: Create AC matrix and Y-matrix
rfx::Load -dataset "ACPLT_nMOS" -file "DATA/nMOS_ac_des.plt" \
    -port1 1 -port2 2 -biasport "v(1)" -devicewidth 0.9

# Create Y-parameter dataset as a function of bias
rfx::CreateDataset -dataset "Y_bias" -xaxis "bias" -rfmatrix "Y"

# Create AC parameter dataset as a function of frequency
rfx::CreateDataset -dataset "AC_freq" -xaxis "frequency" -rfmatrix "AC"

# Ex 2: Create AC, Y-matrix, and PSD matrices
rfx::Load -dataset "ACPLT_noise_nMOS" \
    -file "DATA/nMOS_noise_ac_des.plt" \
    -port1 1 -port2 2 -biasport "v(1)"

# Create NVSD dataset as a function of frequency
rfx::CreateDataset -dataset "2port_PSD_freq" -xaxis "frequency" \
    -rfmatrix "SV"
```

Appendix H: Two-Port Network RF Extraction Library

`rfx::CreateDataset`

Table 38 Dataset variable names for -xaxis "frequency" and -rfmatrix "Y"

Dataset variable name	Description
<code>bias_<i> frequency</code>	List of frequencies.
<code>bias_<i> y<ij>_Re</code>	List of the real parts of the Y-parameter, Y_{ij} .
<code>bias_<i> y<ij>_Im</code>	List of the imaginary parts of the Y-parameter, Y_{ij} .
<code>bias_<i> y<ij>_Abs</code>	List of the absolute values of the Y-parameter, Y_{ij} . This variable is created if the keyword <code>-dB</code> is not specified or <code>-dB 0</code> is specified.
<code>bias_<i> y<ij>_Abs_10dB</code>	List of the absolute values of the Y-parameter, Y_{ij} , on a 10 dB scale. This variable is created only if <code>-dB 10</code> is specified.
<code>bias_<i> y<ij>_Abs_20dB</code>	List of the absolute values of the Y-parameter, Y_{ij} , on a 20 dB scale. This variable is created only if <code>-dB 20</code> is specified.
<code>bias_<i> y<ij>_Phase</code>	List of the phases of the Y-parameter, Y_{ij} .

Here, ij refers to the port numbers (11, 12, 21, 22); i is the bias point index; iv varies from `rfx::i_biasstart` to `rfx::i_biasend`. These variables are created for all the Y-parameters: Y_{11} , Y_{12} , Y_{21} , and Y_{22} .

Table 39 Dataset variable names for -xaxis "frequency" and -rfmatrix "SV"

Dataset variable name	Description
<code>bias_<i> frequency</code>	List of frequencies.
<code>bias_<i> sv<ij></code>	List of NVSD autocorrelation coefficients (S_V^{ij} , $i = j$).
<code>bias_<i> sv<ij>_Re</code>	List of the real parts of the NVSD cross-correlation coefficients (S_V^{ij} , $i \neq j$).
<code>bias_<i> sv<ij>_Im</code>	List of the imaginary parts of the NVSD cross-correlation coefficients (S_V^{ij} , $i \neq j$).

Appendix H: Two-Port Network RF Extraction Library

`rfx::CreateDataset`

Table 39 Dataset variable names for -xaxis "frequency" and -rfmatrix "SV" (Continued)

Dataset variable name	Description
<code>bias_<i> sv<ij>_Abs</code>	List of the absolute values of the NVSD cross-correlation coefficients, (S_V^{ij} , $i \neq j$). This variable is created if the keyword <code>-dB</code> is not specified or <code>-dB 0</code> is specified.
<code>bias_<i> sv<ij>_Phase</code>	List of the phases of the cross-correlation coefficients, (S_V^{ij} , $i \neq j$).

Here, ij refers to the port numbers (11, 12, 21, 22); i is the bias point index; iv varies from `rfx::i_biasstart` to `rfx::i_biasend`. These variables are created for all of the coefficients of the S_V -matrix: S_V^{11} , S_V^{12} , S_V^{21} , and S_V^{22} .

Note:

[Table 39](#) lists the variables corresponding to S_V^{ij} . Similar variables are created for S_I^{ij} . For the partial noise spectral densities that describe the contribution of specific noise sources, the name of the specific noise source is included in parentheses, for example, `svij(ee)`, `svij(eeMonoGR)`, and `svij_Re(eeMonoGR)`. If there is a named noise specification, this name is prefixed to the name of the variable. For example, if the name of the noise specification is `diff`, examples of variable names are `diff_svij(ee)`, `diff_svij(eeDiff)`, and `diff_svij_Re(eeDiff)`.

[Table 40 on page 529](#) lists examples of PSD matrix coefficients and the corresponding names of PSD variables in the AC data file, the PSD Tcl array name and element, and the name of the corresponding dataset variables.

Table 40 PSD data in AC data file, PSD Tcl array element, and dataset variables

Coefficient of PSD matrix	PSD variable in AC data file	PSD Tcl array element	Dataset variables
NVSD matrix S_V			
$S_V^{11}(f, v)$	<code>S_V(1)</code>	<code>rfx:::SV(0,1,1,\$if, \$iv)</code>	<code>sv11</code>
$S_V^{12}(f, v)$	–	<code>rfx:::SV(0,1,2,\$if, \$iv)</code> <code>rfx:::SV(1,1,2,\$if, \$iv)</code>	<code>sv12_Re</code> <code>sv12_Im</code> <code>sv12_Abs</code> <code>sv12_Phase</code>

Appendix H: Two-Port Network RF Extraction Library

`rfx::CreateDataset`

Table 40 PSD data in AC data file, PSD Tcl array element, and dataset variables

Coefficient of PSD matrix	PSD variable in AC data file	PSD Tcl array element	Dataset variables
$S_V^{21}(f, v)$	<code>ReS_VXV(2,1)</code> <code>ImS_VXV(2,1)</code>	<code>rfx::SV(0,2,1,\$if, \$iv)</code> <code>rfx::SV(1,2,1,\$if, \$iv)</code>	<code>sv21_Re</code> <code>sv21_Im</code> <code>sv21_Abs</code> <code>sv21_Phase</code>
$S_V^{22}(f, v)$	<code>S_V(2)</code>	<code>rfx::SV(2,2)</code>	<code>sv22</code>
The matrix $S_{V,n}^{\text{GR}}$ with coefficients corresponding to electron NVSD due to monopolar GR LNS (a partial PSD)			
$S_{V,n}^{\text{GR}, 11}(f, v)$	<code>S_V_eeMonoGR(1)</code>	<code>rfx::SVeeMonoGR(0,1,1,\$if,\$iv)</code>	<code>sv11(eeMonoGR)</code>
$S_{V,n}^{\text{GR}, 12}(f, v)$	-	<code>rfx::SVeeMonoGR(0,1,2,\$if,\$iv)</code> <code>rfx::SVeeMonoGR(1,1,2,\$if,\$iv)</code>	<code>sv12_Re(eeMonoGR)</code> <code>sv12_Im(eeMonoGR)</code> <code>sv12_Abs(eeMonoGR)</code> <code>sv12_Phase(eeMonoGR)</code>
$S_{V,n}^{\text{GR}, 21}(f, v)$	<code>ReS_VXV_eeMonoGR(2,1)</code> <code>ImS_VXV_eeMonoGR(2,1)</code>	<code>rfx::SVeeMonoGR(0,2,1,\$if,\$iv)</code> <code>rfx::SVeeMonoGR(1,2,1,\$if,\$iv)</code>	<code>sv21_Re(eeMonoGR)</code> <code>sv21_Im(eeMonoGR)</code> <code>sv21_Abs(eeMonoGR)</code> <code>sv21_Phase(eeMonoGR)</code>
$S_{V,n}^{\text{GR}, 22}(f, v)$	<code>S_V_eeMonoGR(2)</code>	<code>rfx::SVeeMonoGR(0,2,2,\$if,\$iv)</code>	<code>sv22(eeMonoGR)</code>

rfx::Export

Exports the AC array, or the Y-, h-, Z-, or S-matrix, to a CSV file.

Syntax

```
rfx::Export -rfmatrix "AC" | "Y" | "H" | "Z" | "S"  
  [-file <fileName>] [-xaxis "frequency" | "bias"]  
  [-info 0 | 1 | 2 | 3] [-help 0 | 1]
```

Argument	Description
-rfmatrix "AC" "Y" "H" "Z" "S"	Name of the RF matrix. (String, no default)
-file <fileName>	Name of the CSV file. Default: "rfmatrix.csv"
-xaxis "frequency" "bias"	Specifies either the frequency or the bias as the axis. Selects the sorting order. For example, for -xaxis "frequency", the data is printed using a loop with frequency as the <i>inner variable</i> and bias as the <i>outer variable</i> . Default: "frequency"
-info 0 1 2 3	Sets local information level. Default: 0
-help 0 1	Prints a help screen if set to 1. Default: 0

Returns

None.

Example

```
# Create AC matrix and Y-matrix.  
rfx::Load -dataset "ACPLT_nMOS" -file "DATA/nMOS_ac_des.plt" \  
  -port1 1 -port2 2 -biasport "v(1)" -devicewidth 0.9  
  
rfx::Export -file "Sparam_freq.csv" -rfmatrix "S" -xaxis "frequency"
```

Note:

The CSV file can be loaded into any spreadsheet application. It contains a header that lists the number of bias and frequency points as well as the value of the first and last bias and frequency points. The header is followed by a table, which contains the frequency, the bias, and the real and imaginary parts of the elements of the RF matrix.

Appendix H: Two-Port Network RF Extraction Library

rfx::GetFK1

Two versions of the CSV file can be written. One is sorted by frequencies and the other is sorted by bias points. The keyword `-xaxis` specifies whether the parameters should be sorted by frequency first (`-xaxis "frequency"`), with bias being the secondary parameter, or by bias first (`-xaxis "bias"`) with frequency being the secondary parameter. For example, the CSV file `Sparam_freq.csv` generated by the `rfx::Export` command in the above example contains the following (for `-xaxis "bias"`, the first two columns are reversed):

```
# of bias pts. : 21, first bias: 0, last bias: 1
# of frequencies: 25, first freq: 1e+08, last freq: 1e+12
bias,freq,S11_Re,S11_Im,S12_Re,S12_Im,S21_Re,S21_Im,S22_Re,S22_Im
0,1e+08,0.9991,-2.38e-05,0.00089,1.08e-05,0.00084,1.08e-05,0.999,
-1.9774e-05, 0,1.47e+08,0.9991,-3.50e-05,0.000899,1.58e-05,0.00084,
1.58e-05,0.999,-2.90e-05
0,2.15e+08,0.999,-5.14e-05,0.000899,2.32e-05,0.00084,
2.32e-05,0.9991,-4.26e-05
...
...
```

rfx::GetFK1

Computes the cutoff frequency for stability f_{K1} (see [Cutoff Frequency for Stability on page 519](#)) at all bias points from the Rollett stability factor versus frequency curves. Creates the corresponding datasets if the keyword `-dataset` is specified.

Note:

If f_{K1} is not found, then the procedure returns 0.

Syntax

```
rfx::GetFK1 -out <array_name> [-dataset <dataName>]
[-xscale "lin" | "log"] [-scale <r>] [-target <r>]
[-occurrence <i>] [-info 0 | 1 | 2 | 3] [-help 0 | 1]
```

Argument	Description
<code>-out <array_name></code>	Name of an array to store the results. The array has one string-valued index. The index contains the elements <code>fK1</code> and <code>bias</code> . The values of the <code>fK1</code> element and the <code>bias</code> element are lists of f_{K1} and bias, respectively. (Array name, no default)
<code>-dataset <dataName></code>	Name of Sentaurus Visual dataset containing the variables <code>bias</code> and <code>fK1</code> . The variable <code>bias</code> contains a list of bias values, and the variable <code>fK1</code> contains a list of cutoff frequencies for stability. These variables can be used to plot a f_{K1} versus bias curve. The dataset is created only if this keyword is specified. (String, no default)

Appendix H: Two-Port Network RF Extraction Library

rfx::GetFK1

Argument	Description
-xscale "lin" "log"	Specifies whether the values on the x-axis are linearly or logarithmically distributed. Default: "log"
-scale <r>	Computed f_{K1} is divided by this scaling factor. Use to convert, for example, the f_{K1} value to GHz. (Real number, default: 1.0)
-target <r>	Selects the value of K that should be looked for. (Real number, default: 1.0)
-occurrence <i>	Specifies the n -th interpolated f_{K1} value to be extracted. Use this if multiple frequencies have the same K-value (specified using the keyword -target) at a bias point. (Integer, default: 1)
-info 0 1 2 3	Sets local information level. Default: 0
-help 0 1	Prints a help screen if set to 1. Default: 0

Returns

None.

Example

```
# Create AC matrix and Y-matrix
rfx::Load -dataset "ACPLT_nMOS" -file "DATA/nMOS_ac_des.plt" \
    -port1 1 -port2 2 -biasport "v(1)" -devicewidth 0.9

# Compute fK1 versus bias and corresponding dataset
rfx::GetFK1 -out FK -dataset "fK_bias" -xscale "log" \
    -target 1.0 -occurrence 1 -scale 1e9

puts "Bias Points: $FK(bias)"
puts "Cutoff frequencies for stability \[GHz\]: $FK(fK1)"

#-> Bias Points: 0 0.05 0.1 0.15 0.20 0.25 ...
#-> Cutoff frequencies for stability [GHz]: 0.0 0.0 0.0 0.0 14.53 8.04
...
```

rfx::GetFmax

Computes the maximum frequency of oscillation f_{\max} at all bias points from the power gain (MUG or MAG) versus frequency curves at each bias point. Creates the corresponding datasets if the keywords `dataset` and `slopedataset` are specified.

Note:

This procedure can compute f_{\max} using three different methods (see [Extraction Methods for ft and fmax on page 517](#)). If f_{\max} is not found, or the power gain is less than or equal to 1 (device is passive), the procedure returns 0.

Syntax

```
rfx:::GetFmax -out <array_name> -parameter <r> [-method <string>]
[-dataset <dataName>] [-slopedataset <dataName>]
[-powergain "MUG" | "MAG"] [-xscale "lin" | "log"] [-scale <r>]
[-info 0 | 1 | 2 | 3] [-help 0 | 1]
```

Argument	Description
<code>-out <array_name></code>	Name of an array to store the results. The array has one string-valued index. The index contains the elements <code>fmax</code> and <code>bias</code> . The values of the <code>fmax</code> element and the <code>bias</code> element are lists of f_{\max} and <code>bias</code> , respectively. (Array name, no default)
<code>-parameter <r></code>	The dB point for method "b", specified in units of dB, or the frequency point in units of Hz for method "c". This is a mandatory argument if method "b" or "c" is used. (Real number, no default)
<code>-method <string></code>	Specifies the method to use for computing f_{\max} (see Extraction Methods for ft and fmax on page 517): <ul style="list-style-type: none">"a" or "unit-gain-point" extracts f_{\max} as the frequency at which power gain equals one."b" or "extract-at-dbpoint" extracts f_{\max} by extrapolating the power gain from the point at which it has fallen by a certain number of decibels (called the dB point) from its initial value."c" or "extract-at-frequency" is the same as "b", but the power gain is extrapolated from the specified frequency point. (String, default: "a")

Appendix H: Two-Port Network RF Extraction Library

`rfx::GetFmax`

Argument	Description
<code>-dataset <dataName></code>	Name of Sentaurus Visual dataset containing the variables <code>bias</code> and <code>fmax</code> . The variable <code>bias</code> contains a list of bias values, and the variable <code>fmax</code> contains a list of f_{max} values. These variables can be used to plot a f_{max} versus bias curve. The dataset is created only if <code>-dataset</code> is specified. (String, no default)
<code>-slopedataset <dataName></code>	Name of Sentaurus Visual dataset containing the variables <code>"bias_<i> frequency"</code> and <code>"bias_<i> dMUG"</code> (<code>-powergain "MUG"</code>) or <code>"bias_<i> dMAG"</code> (<code>-powergain "MAG"</code>) corresponding to the bias point <code>bias_<i>.i</code> . <code>i</code> is the bias point index. These variables are created for all the bias point indices (0 to <code>rfx:::i_biasend</code>). The variable <code>frequency</code> contains a list of frequency values, and the variables <code>dMUG</code> and <code>dMAG</code> contain a list of derivatives of MUG and MAG, respectively as a function of frequency. The unit of the power gain derivatives is dB/decade. These variables are used to plot the power gain derivative versus frequency curve for various bias points. The dataset is created only if <code>-slopedataset</code> is specified. (String, no default)
<code>-powergain "MUG" "MAG"</code>	Selects the power gain used for extracting f_{max} . (String, default: "MUG")
<code>-xscale "lin" "log"</code>	Specifies whether the values on the x-axis are linearly or logarithmically distributed. Default: "log"
<code>-scale <r></code>	Computed f_{max} is divided by this scaling factor. Use to convert, for example, the f_{max} value to GHz. (Real number, default: 1.0)
<code>-info 0 1 2 3</code>	Sets local information level. Default: 0
<code>-help 0 1</code>	Prints a help screen if set to 1. Default: 0

Returns

None.

Example

```
# Create AC matrix and Y-matrix
rfx::Load -dataset "ACPLT_nMOS" -file "DATA/nMOS_ac_des.plt" \
    -port1 1 -port2 2 -biasport "v(1)" -devicewidth 0.9

# Compute fmax versus bias. Create dMUG versus frequency
# and fmax versus bias datasets
rfx::GetFmax -out Fmax0 -method "unit-gain-point" -scale 1e9 \
    -xscale "log" -dataset "fmax0_bias" -slopedataset "MUG_slope_freq"
```

Appendix H: Two-Port Network RF Extraction Library

rfx::GetFt

```
puts "Bias Points: $Fmax0(bias)"
puts "Max frequency of oscillation: $Fmax0(fmax)"

#-> Bias Points: 0 0.05 0.10 0.15 0.20 0.25 0.30 ...
#-> Max frequency of oscillation: 0.0 0.0 0.0 0.0 172.24 417.31 640.89
...
```

rfx::GetFt

Computes the cutoff frequency f_t at all bias points from the current gain $|h_{21}|$ versus frequency curves at each bias point. Creates the corresponding datasets if the keywords `dataset` and `slopedataset` are specified.

Note:

This procedure can compute f_t using three different methods (see [Extraction Methods for ft and fmax on page 517](#)). If f_t is not found or $|h_{21}| \leq 1$, the procedure returns 0.

Syntax

```
rfx::GetFt -out <array_name> -parameter <r> [-method <string>]
[-dataset <dataName>] [-slopedataset <dataName>]
[-xscale "lin" | "log"] [-scale <r>]
[-info 0 | 1 | 2 | 3] [-help 0 | 1]
```

Argument	Description
<code>-out <array_name></code>	Name of an array to store the results. The array has one string-valued index. The index contains the elements <code>ft</code> and <code>bias</code> . The values of the <code>ft</code> element and the <code>bias</code> element are lists of f_t and <code>bias</code> , respectively. (Array name, no default)
<code>-parameter <r></code>	The dB point for method " <code>b</code> ", specified in units of dB, or the frequency point in units of Hz for method " <code>c</code> ". This is a mandatory argument if method " <code>b</code> " or " <code>c</code> " is used. (Real number, no default)

Appendix H: Two-Port Network RF Extraction Library

rfx::GetFt

Argument	Description
-method <string>	Specifies the method to use for computing f_t (see Extraction Methods for ft and fmax on page 517): <ul style="list-style-type: none">"a" or "unit-gain-point" extracts f_t as the frequency at which $h_{21} = 0 \text{ dB}$."b" or "extract-at-dbpoint" extracts f_t by extrapolating h_{21} from the point at which h_{21} has fallen by a certain number of decibels (called the DB point) from its initial value."c" or "extract-at-frequency" is the same as "b", but h_{21} is extrapolated from the specified frequency point. (String, default: "a")
-dataset <dataName>	Name of Sentaurus Visual dataset containing the variables <code>bias</code> and <code>ft</code> . The variable <code>bias</code> contains a list of bias values, and the variable <code>ft</code> contains a list of f_t values. These variables can be used to plot a f_t versus bias curve. The dataset is created only if <code>-dataset</code> is specified. (String, no default)
-slopedataset <dataName>	Name of Sentaurus Visual dataset containing the variables " <code>bias_<i></code> <code>frequency</code> " and " <code>bias_<i></code> <code>dh21</code> " corresponding to the bias point <code>bias_<i></code> . i is the bias point index. These variables are created for all the bias point indices (0 to <code>rfx::i_biasend</code>). The variable <code>frequency</code> contains a list of frequency values, and the variable <code>dh21</code> contains a list of the derivatives of $ h_{21} $ as a function of frequency. The unit of <code>dh21</code> is dB/decade. It can be used to plot the derivatives of $ h_{21} $ versus frequency curve at various bias points. The slope dataset is created only if <code>-slopedataset</code> is specified. (String, no default)
-xscale "lin" "log"	Specifies whether the values on the x-axis are linearly or logarithmically distributed. Default: "log"
-scale <r>	Computed f_t is divided by this scaling factor. Use to convert, for example, the f_t value to GHz. (Real number, default: 1.0)
-info 0 1 2 3	Sets local information level. Default: 0
-help 0 1	Prints a help screen if set to 1. Default: 0

Returns

None.

Appendix H: Two-Port Network RF Extraction Library

rfx::GetNearestIndex

Example

```
# Create AC matrix and Y-matrix
rfx::Load -dataset "ACPLT_nMOS" -file "DATA/nMOS_ac_des.plt" \
    -port1 1 -port2 2 -biasport "v(1)" -devicewidth 0.9

# Compute ft versus bias
# Create dh21 versus frequency and ft versus bias datasets
rfx::GetFt -out Ft0 -method "unit-gain-point" -scale 1e9 -xscale "log" \
    -dataset "ft0_bias" -slopedataset "h21_slope_freq"

puts "Cutoff frequencies: $Ft0(ft)"
puts "Bias Points: $Ft0(bias)"

#-> Cutoff frequencies: 0 0.05 0.10 0.15 0.2 0.25 ...
#-> Bias Points: 0.0 0.0 0.0 4.30 25.84 74.21 ...
```

rfx::GetNearestIndex

Finds the index of the entry in an ordered numeric list that is closest to the given target. For example, this procedure can find the index of a frequency or bias point closest to a frequency or bias point of interest (see [Tcl Arrays rfx::AC and rfx::Y on page 507](#)).

Note:

If the target is outside the range of values in the numeric list, this procedure returns the index of the first element or the last element in the numeric list.

Therefore, if the frequency or bias point is outside the range of frequency or bias values, this procedure returns the index of the first element or the last element in the list of frequency or bias values.

Syntax

```
rfx::GetNearestIndex -out <var_name> -target <r> -list <list_of_r>
    [-info 0 | 1 | 2 | 3] [-help 0 | 1]
```

Argument	Description
-out <var_name>	Variable name to store the index.
-target <r>	Target value. (Real number, no default)
-list <list_of_r>	An ordered numeric list. The list can be in ascending or descending order. (List of real numbers, no default)
-info 0 1 2 3	Sets local information level. Default: 0
-help 0 1	Prints a help screen if set to 1. Default: 0

Appendix H: Two-Port Network RF Extraction Library

rfx::GetNearestIndex

Returns

None.

Example

```
# Create AC matrix and Y-matrix
rfx::Load -dataset "ACPLT_nMOS" -file "DATA/nMOS_ac_des.plt" \
    -port1 1 -port2 2 -biasport "v(1)"

# Ex 1: Get nearest bias index corresponding to 0.025 V
rfx::GetNearestIndex -out i_bias -target 0.025 -list $rfx::bias
puts "Nearest bias point index: $i_bias"
set BiasPoint [lindex $rfx::bias $i_bias]
puts "Corresponding bias point: $BiasPoint"

#-> Nearest bias point index: 0
#-> Corresponding bias point: 0

# Ex 2: Get nearest frequency index corresponding to 1e9 Hz
rfx::GetNearestIndex -out i_freq -target 1e9 -list $rfx::freq
puts "Nearest frequency index: $i_freq"
set Frequency [lindex $rfx::freq $i_freq]
puts "Corresponding frequency: $Frequency"

#-> Nearest frequency index: 6
#-> Corresponding frequency: 1.00e+09
```

rfx::GetNoiseFigure

Computes the noise figure, the noise parameters, the input-referred spectral densities, and several other parameters as a function of frequency or bias:

- Noise figure NF (using [Equation 94 on page 523](#)) and minimum noise figure NF_{\min} (using [Equation 93 on page 523](#))
- Noise factor F (using [Equation 92 on page 523](#)) and minimum noise factor F_{\min} (using [Equation 91 on page 522](#))
- Equivalent noise resistance R_n (using [Equation 76 on page 520](#)) and conductance G_n (using [Equation 82 on page 521](#))
- Optimum source admittance Y_{opt} (using [Equation 87](#) and [Equation 88 on page 522](#)) and impedance Z_{opt} (using [Equation 90 on page 522](#))
- Equivalent noise conductance G_u (using [Equation 78 on page 521](#)) and correlation admittance Y_{cor} (using [Equation 80 on page 521](#))
- Normalized quantities r_n , g_n , y_{opt} , z_{opt} , and g_u
- Input-referred spectral densities S_{v_n} (using [Equation 75 on page 520](#)), $S_{v_n i_n}$ (using [Equation 83 on page 521](#)), $S_{i_n v_n}$ (using [Equation 83](#)), and S_{i_n} (using [Equation 81 on page 521](#))
- Noise correlation coefficients C_{i_n} (using [Equation 41 on page 510](#)) and $C_{i_n v_n}$ (using [Equation 84 on page 522](#))

Note:

This procedure uses the `rfx` namespace variables `rfx:::zo` and `rfx:::zs` (see [Characteristic Impedance and Source Impedance on page 525](#)).

Syntax

```
rfx:::GetNoiseFigure -out <array_name> [-xaxis "frequency" | "bias"]
    (-target <r> | -index <i>) [-dataset <dataName>]
    [-info 0 | 1 | 2 | 3] [-help 0 | 1]
```

Argument	Description
<code>-out <array_name></code>	Name of an array to store the results. The array has one string-valued index. The index contains the elements summarized in Table 41 , which also summarizes the values of these elements. The index also contains the element freq (for <code>-xaxis "frequency"</code>) or bias (for <code>-xaxis "bias"</code>). (Array name, no default)

Appendix H: Two-Port Network RF Extraction Library

rfx::GetNoiseFigure

Argument	Description
-xaxis "frequency" "bias"	Specifies either the frequency or bias as the axis. Default: "frequency"
-target <r>	Bias point (for -xaxis "frequency") or frequency point (for -xaxis "bias"). Specify only one of the keywords -target or -index. (Real number, no default)
-index <i>	Index of the bias point (for -xaxis "frequency") or frequency point (for -xaxis "bias"). Specify only one of the keywords -target or -index. (Integer, no default)
-dataset <dataName>	Name of Sentaurus Visual dataset containing the variables summarized in Table 41 . The dataset also contains the variable frequency (for -xaxis "frequency") or bias (for -xaxis "bias"). (String, no default)
-info 0 1 2 3	Sets local information level. Default: 0
-help 0 1	Prints a help screen if set to 1. Default: 0

Table 41 Elements of array index, dataset variable names, and their values

Elements of array index	Dataset variable name	Value
NF, F, NFmin, Fmin	NF, F, NFmin, Fmin	Noise figure, noise factor, minimum noise figure, and minimum noise factor.
Rn, Gn, rn, gn	Rn, Gn, rn, gn	Equivalent noise resistance R_n and conductance G_n , and their normalized values r_n and g_n .
Gopt, Bopt, Ropt, Xopt, gopt, bopt, ropt, xopt	Gopt, Bopt, Ropt, Xopt, gopt, bopt, ropt, xopt	Optimum source conductance, susceptance, resistance, and reactance, and their normalized values.
Gcor, Bcor	Gcor, Bcor	Correlation conductance and susceptance.
Gu, gu	Gu, gu	Equivalent conductance G_u and its normalized values g_u .
ReCi, ImCi, AbsCi, PhaseCi	Ci_Re, Ci_Im, Ci_Abs, Ci_Phase	Real and imaginary parts, absolute value, and phase of C_{i_n} .

Appendix H: Two-Port Network RF Extraction Library

`rfx::GetNoiseFigure`

Table 41 Elements of array index, dataset variable names, and their values

Elements of array index	Dataset variable name	Value
Sv, ReSvi, ImSvi, ReSiv, ImSiv, Si	Sv, Svi_Re, Svi_Im, Siv_Re, Siv_Im, Si	S_{v_n} , real and imaginary parts of $S_{v_n \bar{i}_n}$, real and imaginary parts of $S_{i_n v_n}$, and S_{i_n} .
ReCiv, ImCiv, AbsCiv, PhaseCiv	Civ_Re, Civ_Im, Civ_Abs, Civ_Phase	Real and imaginary parts, absolute value, and phase of $C_{i_n v_n}$.

Returns

None.

Example

```
# Create AC-matrix, Y-matrix and PSD matrices
rfx::Load -dataset "ACPLT_nMOS" -file "DATA/nMOS_noise_ac_des.plt" \
    -port1 1 -port2 2 -biasport "v(1)"

# Compute noise parameters, noise figure and input-referred
# spectral densities at 0 bias, as a function of frequency
rfx::GetNoiseFigure -out NFparam -dataset "NF_freq" \
    -xaxis "frequency" -target 0.0

puts "frequency: $NFparam(freq)"
puts "NF: $NFparam(NF)"
puts "NFmin: $NFparam(NFmin)"
puts "Rn: $NFparam(Rn)"
puts "Gopt: $NFparam(Gopt)"
puts "Bopt: $NFparam(Bopt)"
puts "Sv: $NFparam(Sv)"
puts "Si: $NFparam(Si)"
puts "ReSvi: $NFparam(ReSvi)"
puts "ImSvi: $NFparam(ImSvi)"

#-> frequency: 0.1 0.215 0.464 ...
#-> NF: 113.501 113.501 113.501...
#-> NFmin: 5.310e-07 1.144e-06 2.465e-06 ...
#-> Rn: 1.120e13 1.120e13 1.120e13 ...
#-> Gopt: 5.460e-21 1.176e-20 2.534e-20 ...
#-> Bopt: -2.198e-16 -4.736e-16 -1.020e-15 ...
#-> Sv: 1.793e-07 1.793e-07 1.793e-07 ...
#-> Si: 8.666e-39 4.022e-38 1.867e-37 ...
#-> ReSvi: -1.083e-36 -5.029e-36 -2.334e-35 ...
#-> ImSvi: -3.942e-23 -8.493e-23 -1.830e-22 ...
```

Appendix H: Two-Port Network RF Extraction Library

rfx::GetParsAtPoint

rfx::GetParsAtPoint

Accesses the RF parameters of an RF matrix at a given bias and frequency point.

Syntax

```
rfx::GetParsAtPoint -out <array_name>
    -rfmatrix "AC" | "Y" | "H" | "Z" | "S"
    -biaspoint <r> -freqpoint <r>
    [-info 0 | 1 | 2 | 3] [-help 0 | 1]
```

Argument	Description
-out <array_name>	Name of an array to store the results. The array has one string-valued index. The index contains the elements bias, freq, 11, 12, 21, and 22. The values of the bias element and the freq element are the bias and the frequency point, respectively. The 11, 12, 21, and 22 elements are the complex RF parameters (list containing real and imaginary parts). (Array name, no default)
-rfmatrix "AC" "Y" "H" "Z" "S"	Name of the RF matrix. (String, no default)
-biaspoint <r>	Bias point. (Real number, no default)
-freqpoint <r>	Frequency point. (Real number, no default)
-info 0 1 2 3	Sets local information level. Default: 0
-help 0 1	Prints a help screen if set to 1. Default: 0

Returns

None.

Example

```
# Create AC- and Y-matrix
rfx::Load -dataset "ACPLT_nMOS" -file "DATA/nMOS_ac_des.plt" \
    -port1 1 -port2 2 -biasport "v(1)" -devicewidth 0.9

# Compute AC parameters at first bias point and frequency
set BiasPoint [lindex $rfx::bias 0]
set Frequency [lindex $rfx::freq 0]
rfx::GetParsAtPoint -out ReImData -rfmatrix "AC" \
    -biaspoint $BiasPoint -freqpoint $Frequency

puts "Bias Point: $ReImData(bias)"
puts "Freq Point: [format %.2e $ReImData(freq)]"
```

Appendix H: Two-Port Network RF Extraction Library

rfx::GetPowerGain

```
puts "ac11: $ReImData(11)"
puts "ac12: $ReImData(12)"
puts "ac21: $ReImData(21)"
puts "ac22: $ReImData(22)"

#-> Bias Point: 0
#-> Freq Point: 1.00e+08
#-> ac11: 9.00e-06 3.80e-16
#-> ac12: -9.00e-06 -1.72e-16
#-> ac21: -8.41e-06 -1.72e-16
#-> ac22: 9.08e-06 3.15e-16
```

rfx::GetPowerGain

Computes the following as a function of frequency (at a fixed bias point) or bias (at a fixed frequency point):

- Rollett stability factor (using [Equation 51](#)) and stability condition delta (using [Equation 52](#))
- MSG (using [Equation 54](#)) (linear scale as well as 10 dB scale)
- MAG (using [Equation 55](#) and [Equation 56](#)) (linear scale as well as 10 dB scale)
- MUG (using [Equation 57](#)) (linear scale as well as 10 dB scale)
- Unilateral figure of merit (using [Equation 58](#))

Note:

If the denominator in [Equation 51](#), [Equation 54](#), [Equation 57](#), or [Equation 58](#) is 0, the procedure returns a value of 10^{-20} .

If MUG, MSG, or MAG is 0, the procedure returns a value of 10^{-20} .

Syntax

```
rfx::GetPowerGain -out <array_name> [-xaxis "frequency" | "bias"]
  (-target <r> | -index <i>) [-dataset <dataName>]
  [-powergain "all" | "MUG" | "MSG" | "MAG"]
  [-info 0 | 1 | 2 | 3] [-help 0 | 1]
```

Appendix H: Two-Port Network RF Extraction Library

rfx::GetPowerGain

Argument	Description
-out <array_name>	Name of an array to store the results. The array has one string-valued index. The index contains the elements K, delta, and freq (for -xaxis "frequency") or bias (for -xaxis "bias"). The values of the K, delta, freq, and bias elements are the Rollett stability factor, the stability condition delta, the frequency, and the bias, respectively. In addition, for -powergain "all", the index contains the elements MUG, MUG_dB, MSG, MSG_dB, MAG, MAG_dB, and U. These are MUG (linear scale), MUG (10 dB scale), MSG (linear scale), MSG (10 dB scale), MAG (linear scale), MAG (10 dB scale), and U_f , respectively. For -powergain "MUG", only the MUG, MUG_dB, and U elements are created. For -powergain "MSG", only the MSG and MSG_dB elements are created. For -powergain "MAG", only the MAG and MAG_dB elements are created. (Array name, no default)
-xaxis "frequency" "bias"	Specifies either the frequency or bias as the axis. Default: "frequency"
-target <r>	Bias point (for -xaxis "frequency") or frequency point (for -xaxis "bias"). Specify only one of the keywords -target or -index. (Real number, no default)
-index <i>	Index of the bias point (for -xaxis "frequency") or frequency point (for -xaxis "bias"). Specify only one of the keywords -target or -index. (Integer, no default)
-dataset <dataName>	Name of Sentaurus Visual dataset containing the variables K, delta, and frequency (for -xaxis "frequency") or bias (for -xaxis "bias"). In addition, for -powergain "all", the variables MSG, MSG_dB, MAG, MAG_dB, MUG, MUG_dB, and U are created. For -powergain "MUG", the variables MUG, MUG_dB, and U are created. For -powergain "MSG", the variables MSG and MSG_dB are created. For -powergain "MAG", the variables MAG and MAG_dB are created. (String, no default)

Appendix H: Two-Port Network RF Extraction Library

rfx::GetPowerGain

Argument	Description
-powergain "all" "MUG" "MSG" "MAG"	Specifies the power gains to compute: <ul style="list-style-type: none">• For -powergain "all", MSG, MAG, MUG, and U are computed.• For -powergain "MUG", only MUG and U are computed.• For -powergain "MSG", only MSG is computed.• For -powergain "MAG", only MAG is computed. Power gains are computed on both the linear scale and 10 dB scale. (String, default: "all")
-info 0 1 2 3	Sets local information level. Default: 0
-help 0 1	Prints a help screen if set to 1. Default: 0

Returns

None.

Example

```
# Create AC matrix and Y-matrix
rfx::Load -dataset "ACPLT_nMOS" -file "DATA/nMOS_trunc_ac_des.plt" \
    -port1 1 -port2 2 -biasport "v(1)" -devicewidth 0.9

# Compute stability factor and power gain at 1e8 Hz,
# as a function of bias
rfx::GetPowerGain -out gain -dataset "Pgain_freq" -powergain "all" \
    -xaxis "bias" -target 0.0

puts "Bias Points: $gain(bias)"
puts "K: $gain(K)"
puts "delta: $gain(delta)" puts "MSG: $gain(MSG)"
puts "MSG_dB: $gain(MSG_dB)"
puts "MAG: $gain(MAG)"
puts "MAG_dB: $gain(MAG_dB)"
puts "MUG: $gain(MUG)"
puts "MUG_dB: $gain(MUG_dB)"
puts "U: $gain(U)"

#-> Bias Points: 0 0.1 ...
#-> K: 0.18019 0.0161 ...
#-> delta: 0.99999 0.99992 ...
#-> MSG: 5.55723 62.92755 ...
#-> MSG_dB: 7.44858 17.98841 ...
#-> MAG: 5.55723 62.92755 ...
#-> MAG_dB: 7.44858 17.98841 ...
#-> MUG: -102686.50144 -1218108.32141 ...
#-> MUG_dB: 50.11513 60.85686 ...
#-> U: 66383.57685 134797.56995 ...
```

rfx::Load

Loads a Sentaurus Device AC data file and creates a Sentaurus Visual dataset. It also creates the Tcl arrays, `rfx:::AC` and `rfx:::Y` (see [Tcl Arrays rfx:::AC and rfx:::Y on page 507](#)), along with several `rfx` namespace variables summarized in [Table 37 on page 523](#). The Tcl arrays for the power spectral densities are created only if the AC data file contains PSD data (see [Power Spectral Density Tcl Arrays on page 511](#)).

Syntax

```
rfx:::Load -file <fileName>
    [-biasport <stringValue>] [-biassportsign +1 | -1]
    [-dataset <dataName>] [-devicewidth <r>]
    [-port1 <integer | stringValue>]
    [-port2 <integer | stringValue>]
    [-info 0 | 1 | 2 | 3] [-help 0 | 1]
```

Argument	Description
<code>-file <fileName></code>	Name of the Sentaurus Device AC data file. (String, no default)
<code>-biasport <stringValue></code>	Name of biased port. For example, " <code>v(1)</code> " for voltage on port or node 1, or " <code>i(vc,2)</code> " for current flowing out of the voltage source <code>vc</code> at port or node 2. (String, default: " <code>v(1)</code> ")
<code>-biassportsign +1 -1</code>	Sign of the bias port. Used to reverse the polarity of the bias. Default: +1
<code>-dataset <dataName></code>	Name of the created dataset. It contains the data from the Sentaurus Device AC data file. (String, default: " <code>ACPLT</code> ")
<code>-devicewidth <r></code>	Device width multiplier (device width in the z-direction, L_z ; see Device Width Scaling for 2D Structures on page 512) in μm . (Real number, default: 1.0)
<code>-port1 <integer stringValue></code>	Name of the input port of the two-port network. The port name must agree with the node names defined in the Sentaurus Device System section. (Integer or string, default: 1)
<code>-port2 <integer stringValue></code>	Name of the output port of the two-port network. The port name must agree with the node names defined in the Sentaurus Device System section. (Integer or string, default: 2)
<code>-info 0 1 2 3</code>	Sets local information level. Default: 0
<code>-help 0 1</code>	Prints a help screen if set to 1. Default: 0

Appendix H: Two-Port Network RF Extraction Library

rfx::Load

Returns

None.

Example

```
# Create AC matrix and Y-matrix
rfx::Load -dataset "ACPLT_nMOS" -file "DATA/nMOS_ac_des.plt" \
    -port1 1 -port2 2 -biasport "v(1)" -devicewidth 0.9

# rfx namespace variables
puts "Number of bias points: $rfx::nbias"
puts "Bias points: $rfx::bias"
puts "Number of frequencies: $rfx::nfreq"
puts "Frequencies: $rfx::freq"

# Print the Y-matrix
set biases [get_variable_data v(1) -dataset ACPLT]
foreach P1 {1 2} {
    foreach P2 {1 2} {
        foreach Bias $biases {
            set if 0                      ;# Frequency index
            set iv 0                      ;# Bias point index
            puts "Frequency: [lindex $rfx::freq $if]"
            puts "Bias point: [lindex $rfx::bias $iv]"
            puts "Y(0,$P1,$P2,$if,$iv): $rfx::Y(0,$P1,$P2,$if,$iv)"
            puts "Y(1,$P1,$P2,$if,$iv): $rfx::Y(0,$P1,$P2,$if,$iv)"
            if { $iv < [expr $rfx::nbias-1] } {
                incr iv
            } elseif { $if < [expr $rfx::nfreq-1] } {
                incr if
                set iv 0
            }
        }
    }
}
#-> Number of bias points: 21
#-> Bias points: 0 0.05 0.1 0.15 ...
#-> Number of frequencies: 25
#-> Frequencies: 1e+08 1.47e+08 2.15e+08 ...

#-> Frequency: 1e+08
#-> Bias point: 0
#-> Y(0,1,1,0,0): 9.00e-06
#-> Y(1,1,1,0,0): 9.00e-06
#-> Frequency: 1e+08
#-> Bias point: 0.05
#-> Y(0,1,1,0,1): 9.00e-06
#-> Y(1,1,1,0,1): 9.00e-06
...
...
```

Appendix H: Two-Port Network RF Extraction Library

rfx::NoiseFigure

Note:

It is assumed that the Sentaurus Device AC data file contains one or more frequency sweeps with a voltage bias or current bias as the control variable.

The keyword `-biasport` defines which port is biased and whether the biasing is a voltage or a current condition. For a current condition, the syntax for `biasport` is more complex. For example, if the name of the current source is `vc` and the name of the port is `2`, `-biasport` is specified as `-biasport "i(vc,2)"`.

In this case, however, Sentaurus Device must also be instructed to include the current at this node through this device in the Sentaurus Device AC data file. This is performed in the `System` section of the Sentaurus Device input file with. For example:

```
System {
    HBT hbt (base=1 collector=2 emitter=0)
    Vsource_pset vb ( 1 0 ){ dc = 0 }
    Vsource_pset vc ( 2 0 ){ dc = 0 }
    ACPlot(v(1) v(2) i(vb 1) i(vc 2))
}
```

Internally, the arrays `rfx::AC` and `rfx::Y` use the port numbers `1` and `2` corresponding to `port1` and `port2` as array indices (see [Tcl Arrays rfx::AC and rfx::Y on page 507](#)). This convention is also valid for the PSD Tcl arrays.

rfx::NoiseFigure

Computes the noise figure, the noise parameters, the power spectral densities, and other parameters at a fixed frequency and bias point:

- Noise figure NF (using [Equation 94 on page 523](#)) and minimum noise figure NF_{\min} (using [Equation 93 on page 523](#))
- Noise factor F (using [Equation 92 on page 523](#)) and minimum noise factor F_{\min} (using [Equation 91 on page 522](#))
- Equivalent noise resistance R_n (using [Equation 76 on page 520](#)) and conductance G_n (using [Equation 82 on page 521](#))
- Optimum source admittance Y_{opt} (using [Equation 87](#) and [Equation 88 on page 522](#)) and impedance Z_{opt} (using [Equation 90 on page 522](#))
- Equivalent noise conductance G_u (using [Equation 78 on page 521](#)) and correlation admittance Y_{cor} (using [Equation 80 on page 521](#))
- Normalized quantities r_n , g_n , y_{opt} , z_{opt} , and g_u

Appendix H: Two-Port Network RF Extraction Library

rfx::NoiseFigure

- Input-referred spectral densities S_{v_n} (using [Equation 75 on page 520](#)), S_{v_i} (using [Equation 83 on page 521](#)), $S_{i_n v_n}$ (using [Equation 83](#)), and S_{i_n} (using [Equation 81 on page 521](#))
- Noise correlation coefficients C_{i_n} (using [Equation 41 on page 510](#)) and $C_{i_n v_n}$ (using [Equation 84 on page 522](#))

Note:

This procedure uses the `rfx` namespace variables `rfx:::zo` and `rfx:::zs` (see [Characteristic Impedance and Source Impedance on page 525](#)).

Syntax

```
rfx:::NoiseFigure SI11 SI12 SI22 Y11 Y21
```

Argument	Description
SI11 SI12 SI22	The coefficients S_I^{11} , S_I^{12} , and S_I^{22} of the complex NISD matrix for a fixed frequency and bias point in the form of three lists. Each list contains the real and imaginary parts. (List of real numbers, no default)
Y11 Y21	The complex elements Y_{11} and Y_{21} of the Y-matrix for a fixed frequency and bias point in the form of two lists. Each list contains the real and imaginary parts. (List of real numbers, no default)

Returns

An array having one string-valued index. The index contains elements summarized in [Table 41 on page 541](#), which also summarizes the values of these elements.

Example

```
set SI11 [list 8.24593987e-23 0.0]
set SI12 [list -3.91285654e-23 -1.08922612e-23]
set SI22 [list 4.57789473e-23 0.0]
set Y11 [list 0.00613324387 0.0170027533]
set Y21 [list -0.00506706586 -0.0060744537]

rfx:::NoiseFigure $SI11 $SI12 $SI22 $Y11 $Y21

puts "NF: $NFparameters(NF)"
puts "NFmin: $NFparameters(NFmin)"
puts "Rn: $NFparameters(Rn)"
puts "Gopt: [lindex $NFparameters(Yopt) 0]"
puts "Bopt: [lindex $NFparameters(Yopt) 1]"
puts "Sv: $NFparameters(Sv)"
puts "Si: $NFparameters(Si)"
puts "ReSvi: [lindex $NFparameters(Svi) 0]"
puts "ImSvi: [lindex $NFparameters(Svi) 1]"
```

Appendix H: Two-Port Network RF Extraction Library

rfx::PolarBackdrop

```
#-> NF: 4.21
#-> NFmin: 3.19
#-> Rn: 45.68
#-> Gopt: 0.0086
#-> Bopt: -0.011
#-> Sv: 7.32e-19
#-> Si: 1.36e-22
#-> ReSvi: 2.38e-21
#-> ImSvi: -7.76e-21
```

rfx::PolarBackdrop

Creates a ruled background on which RF parameters in polar coordinates are plotted.
Creates two families of curves: a set of concentric circles and a set of angular lines.

Syntax

```
rfx::PolarBackdrop -plot <plotName> -r <list_of_r> -phi <list_of_r>
  [-dataset <dataName>] [-color <stringValue>]
  [-linestyle <stringValue>] [-linewidth <r>]
  [-info 0 | 1 | 2 | 3] [-help 0 | 1]
```

Argument	Description
-plot <plotName>	Name of the polar plot. (String, no default)
-r <list_of_r>	A list of increasing radial values. (List of real numbers, no default)
-phi <list_of_r>	A list of increasing angular values. (List of real numbers, no default)
-dataset <dataName>	Name of the dataset used to create the polar background. (String, default: "PolarBackdrop")
-color <stringValue>	Sets the color of the curves of the polar background. (String, default: "black")
-linestyle <stringValue>	Sets the style of the curve lines of the polar background. (String, default: "dash")
-linewidth <r>	Sets the line width of the curve lines of the polar background. (Real number, default: 2.0)
-info 0 1 2 3	Sets local information level. Default: 0
-help 0 1	Prints a help screen if set to 1. Default: 0

Appendix H: Two-Port Network RF Extraction Library

rfx::PowerGain

Returns

None.

Example

```
set Rs [list 0.25 0.5 0.75 1.0 1.25]
set Phis [list 0 30 60 90 120 150]
rfx::PolarBackdrop -plot Plot_Polar -r $Rs -phi $Phis
```

rfx::PowerGain

Computes the following at a fixed bias and frequency point:

- Rollett stability factor (using [Equation 51](#)) and stability condition delta (using [Equation 52](#))
- MSG (using [Equation 54](#)) (linear scale)
- MAG (using [Equation 55](#) and [Equation 56](#)) (linear scale)
- MUG (using [Equation 57](#)) (linear scale)
- Unilateral figure of merit (using [Equation 58](#))

Syntax

```
rfx::PowerGain S11 S12 S21 S22
```

Argument	Description
S11 S12 S21 S22	The complex S-matrix for a fixed frequency and bias point in the form of four lists. Each list contains the real and imaginary parts of an element S_{ij} of the S-matrix. (List of real numbers, no default)

Returns

The Rollett stability factor, the stability condition delta, MSG, MAG, MUG, and U_f at a fixed frequency and bias point in the form of a list.

Example

```
set S11 [list 0.999999877967 -8.36457327936e-05]
set S12 [list -1.20947562066e-07 1.09859319808e-05]
set S21 [list -0.0536108305232 4.08872504539e-05]
set S22 [list 0.997240784695 -2.17611978854e-05]
set Pgain [rfx::PowerGain $S11 $S12 $S21 $S22]

puts "K: [lindex $Pgain 0]"
```

Appendix H: Two-Port Network RF Extraction Library

rfx::RFCLList

```
puts "delta: [lindex $Pgains 1]"
puts "MSG: [lindex $Pgains 2]"
puts "MAG: [lindex $Pgains 3]"
puts "MUG: [lindex $Pgains 4]"
puts "U: [lindex $Pgains 5]"

#-> K: -0.009
#-> delta: 0.997
#-> MSG: 4879.658
#-> MAG: 4879.658
#-> MUG: -116265.652
#-> U: 449.598
```

rfx::RFCLList

Accesses a slice of the Y-, h-, Z-, S-matrix, or PSD matrices, typically, to generate a curve of an RF parameter or PSD as a function of bias or frequency.

Syntax

```
rfx::RFCLList -out <array_name> -rfparameter <string> -index <i>
[-noisename <string>]
[-noisesource "ee" | "hh" | "eeDiff" | "hhDiff" | "eeMonoGR" |
 "hhMonoGR" | "eeFlickerGR" | "hhFlickerGR"]
[-xaxis "frequency" | "bias"] [-info 0 | 1 | 2 | 3] [-help 0 | 1]
```

Argument	Description
-out <array_name>	Name of an array to store the results. The array has one string-valued index. The index contains the elements Re and Im. The values of the Re element and the Im element are lists of the real part and imaginary part, respectively, of the RF parameter. (Array name, no default)
-rfparameter <string>	Parameter identifier. Use, for example, the notation h21 for the h-matrix element h_{21} . For PSDs, use sv<ij> or si<ij>. Here, ij refers to the port numbers (11, 12, 21, 22). In addition, for partial noise spectral density, use the -noisesource keyword. (String, no default)
-index <i>	Selects the index of the slice. (Integer, no default)
-noisename <string>	Name of the noise specification. Required only for -rfparameter sv<ij> or si<ij>, and when a named noise specification was used to perform noise analysis. (String, default: "")

Appendix H: Two-Port Network RF Extraction Library

rfx::RFCLList

Argument	Description
-noisesource "ee" "hh" "eeDiff" "hhDiff" "eeMonoGR" "hhMonoGR" "eeFlickerGR" "hhFlickerGR"	Name of the noise source. Only used for accessing the matrices of partial PSDs. For -rfparameter si<ij>, the only allowed values of this keyword are "ee" and "hh". (String, no default)
-xaxis "frequency" "bias"	Specifies either the frequency or bias as the axis. Default: "frequency"
-info 0 1 2 3	Sets local information level. Default: 0
-help 0 1	Prints a help screen if set to 1. Default: 0

Returns

None.

Example

```
# For v=0.5, return real and imaginary parts of z11 as a function
# of frequency

# Create AC matrix and Y-matrix
rfx::Load -dataset "ACPLT_nMOS" -file "DATA/nMOS_ac_des.plt" \
    -port1 1 -port2 2 -biasport "v(1)" -devicewidth 0.9

rfx::GetNearestIndex -out i_bias -target 0.5 -list $rfx::bias
puts "Nearest bias point index: $i_bias"
set BiasPoint [lindex $rfx::bias $i_bias]
puts "Corresponding bias point: $BiasPoint"

rfx::RFCLList -out ReImz11 -rfparameter "z11" -xaxis "frequency" \
    -index $i_bias
puts "Re(ReImz11): $ReImz11(Re)"
puts "Im(ReImz11): $ReImz11(Im)"

#-> Nearest bias point index: 10
#-> Corresponding bias point: 0.5
#-> Re(ReImz11): 10748.32 10746.14 10741.47 ...
#-> Re(ReImz11): -138.16 -202.74 -297.45 ...
```

rfx::SmithBackdrop

Creates a Smith chart–ruled background on which RF parameters are plotted. Creates two families of curves: the normalized resistance circles and the normalized reactance (capacitive or inductive) arcs.

Syntax

```
rfx::SmithBackdrop -plot <plotName> -r <list_of_r> -x <list_of_x>
  [-dataset <dataName>] [-color <stringValue>]
  [-linestyle <stringValue>] [-linewidth <r>]
  [-info 0 | 1 | 2 | 3] [-help 0 | 1]
```

Argument	Description
-plot <plotName>	Name of the Smith chart. (String, no default)
-r <list_of_r>	A list of normalized resistance values. The list must contain positive values, which monotonically increase. (List of real numbers, no default)
-x <list_of_x>	A list of normalized reactance values. The list must contain nonzero positive values, which monotonically increase. (List of real numbers, no default)
-dataset <dataName>	Name of dataset used to create the Smith chart background. (String, default: "SmithBackdrop")
-color <stringValue>	Sets the color of the curves of the Smith chart background. (String, default: "black")
-linestyle <stringValue>	Sets the style of the curve lines of the Smith chart background. (String, default: "dash")
-linewidth <r>	Sets the line width of the curve lines of the Smith chart background. (Real, default: 2.0)
-info 0 1 2 3	Sets local information level. Default: 0
-help 0 1	Prints a help screen if set to 1. Default: 0

Returns

None.

Appendix H: Two-Port Network RF Extraction Library

rfx::Y2H

Example

```
set Rs [list 0 0.3333 1.0 3.0]
set Xs [list 0.268 0.575 1 1.73 3.75]
rfx::SmithBackdrop -plot Plot_Smith -r $Rs -x $Xs
```

rfx::Y2H

Converts Y-parameters to h-parameters at a fixed frequency and bias point using [Equation 46 on page 513](#).

Syntax

```
rfx::Y2H Y11 Y12 Y21 Y22
```

Argument	Description
Y11 Y12 Y21 Y22	The complex Y-matrix for a fixed frequency and bias point in the form of four lists. Each list contains the real and imaginary parts of an element Y_{ij} of the Y-matrix. (List of real numbers, no default)

Returns

The complex h-matrix at the fixed frequency and bias point in the form of four lists. Each list contains the real and imaginary parts of an element h_{ij} of the h-matrix.

Example

```
set Y11 [list 1.21582e-09 8.33508513295e-07]
set Y12 [list 1.21693e-09 -1.10011034906e-07]
set Y21 [list 0.000536849 -3.81214675594e-07]
set Y22 [list 2.76303e-05 2.15260671987e-07]

set H [rfx::Y2H $Y11 $Y12 $Y21 $Y22]

puts "h11= [lindex $H 0]"
puts "h12= [lindex $H 1]"
puts "h21= [lindex $H 2]"
puts "h22= [lindex $H 3]"

#-> h11= 1750.04 -1199745.24
#-> h12= 0.13 0.0017
#-> h21= 0.48 -644.08
#-> h22= 9.85e-05 1.05e-06
```

rfx::Y2S

Converts Y-parameters to S-parameters at a fixed frequency and bias point using [Equation 47 on page 513](#). The S-parameters are computed using a characteristic impedance value that defaults to $50\ \Omega$.

Note:

This procedure uses the variable `rfx::z0` (see [Characteristic Impedance and Source Impedance on page 525](#)).

Syntax

```
rfx::Y2S Y11 Y12 Y21 Y22
```

Argument	Description
<code>Y11 Y12 Y21 Y22</code>	The complex Y-matrix for a fixed frequency and bias point in the form of four lists. Each list contains the real and imaginary parts of an element Y_{ij} of the Y-matrix. (List of real numbers, no default)

Returns

The complex S-matrix in the form of four lists at a fixed frequency and bias point. Each list contains the real and imaginary parts of an element S_{ij} of the S-matrix.

Example

```
set rfx::z0 100.0
set Y11 [list 1.21582e-09 8.33508513295e-07]
set Y12 [list 1.21693e-09 -1.10011034906e-07]
set Y21 [list 0.000536849 -3.81214675594e-07]
set Y22 [list 2.76303e-05 2.15260671987e-07]

set S [rfx::Y2S $Y11 $Y12 $Y21 $Y22]

puts "S11= [lindex $S 0]"
puts "S12= [lindex $S 1]"
puts "S21= [lindex $S 2]"
puts "S22= [lindex $S 3]"

#-> S11= 0.9999997 -0.00017
#-> S12= -2.40e-07 2.19e-05
#-> S21= -0.11 8.73e-05
#-> S22= 0.99 -4.40e-05
```

rfx::Y2Z

Converts Y-parameters to Z-parameters at a fixed frequency and bias point using [Equation 49 on page 514](#).

Syntax

```
rfx:::Y2Z Y11 Y12 Y21 Y22
```

Argument	Description
Y11 Y12 Y21 Y22	The complex Y-matrix for a fixed frequency and bias point in the form of four lists. Each list contains the real and imaginary parts of an element Y_{ij} of the Y-matrix. (List of real numbers, no default)

Returns

The complex Z-matrix in the form of four lists at a fixed frequency and bias point. Each list contains the real and imaginary parts of an element Z_{ij} of the Z-matrix.

Example

```
set Y11 [list 1.21582e-09 8.33508513295e-07]
set Y12 [list 1.21693e-09 -1.10011034906e-07]
set Y21 [list 0.000536849 -3.81214675594e-07]
set Y22 [list 2.76303e-05 2.15260671987e-07]

set Z [rfx:::Y2Z $Y11 $Y12 $Y21 $Y22]

puts "Z11= [lindex $Z 0]"
puts "Z12= [lindex $Z 1]"
puts "Z21= [lindex $Z 2]"
puts "Z22= [lindex $Z 3]"

#-> Z11= -482.36 -336580.47
#-> Z12= 1340.15 2.46
#-> Z21= 64960.88 6539151.68
#-> Z22= 10152.58 -108.46
```

Complex Arithmetic Support

This section describes procedures for performing complex arithmetic. For most procedures, the RF extraction library contains two versions of a procedure: scalar and vectorial. The name of the scalar version of a procedure ends with *c*, and the name of the corresponding vectorial version ends with *v*.

The scalar version operates on a single complex number or two complex numbers. A complex number is specified using a Tcl list containing the real and imaginary parts of the complex number. For example, the absolute value of the complex number, $z = 4 + 3i$ can be computed using the procedure `rfx::Abs_c` as follows:

```
set z [list 4 3]
puts [rfx::Abs_c $z]
#-> 5.0
```

The vectorial version operates on either a single list of complex numbers or two lists of complex numbers. The list of complex numbers is specified using arrays that have one string-valued index. The index contains the elements `Re` and `Im`. The values of the `Re` element and the `Im` element are the real and imaginary parts, respectively, of the list of complex numbers. For example, the absolute values of the complex numbers $z_1 = 2 + 3i$ and $z_2 = -1 + i$ can be computed using the procedure `rfx::Abs_v` as follows:

```
set Z(Re) [list 2 -1]
set Z(Im) [list 3 1]
rfx::Abs_v -out absvals -z Z
puts "abs values = $absvals"
#-> abs values = 3.606 1.414
```

All the procedures except `rfx::Polar2Cart_c` and `rfx::Polar2Cart_v` operate on complex numbers specified in Cartesian coordinates. The `rfx::Polar2Cart` procedures operate on a complex number or a list of complex numbers specified in polar coordinates, which are represented by a Tcl list or an array, respectively. For example, the complex number $\sqrt{2}\angle 45^\circ$ is specified using:

```
set z [list 1.414 45]
```

rfx::Abs_c

Computes the absolute value of a complex number.

Syntax

```
rfx::Abs_c z
```

Argument	Description
z	A list containing the real and imaginary parts of a complex number. (List of real numbers, no default)

Returns

A single value, the absolute value of a complex number.

Example

```
set z [list 4 3]
puts [rfx::Abs_c $z]
#-> 5.0
```

rfx::Abs_v

Computes the absolute values of a list of complex numbers, and also computes the absolute values in units of 10 dB or 20 dB.

Syntax

```
rfx::Abs_v -out <list_name> -z <array_name> [-dB 0 | 10 | 20]
[-info 0 | 1 | 2 | 3] [-help 0 | 1]
```

Argument	Description
-out <list_name>	Name of a list to store the list of absolute values. (List name, no default)
-z <array_name>	Name of an array containing a list of complex numbers. The array has one string-valued index. The index contains the elements Re and Im. The values of the Re element and the Im element are the real and imaginary parts, respectively, of the complex numbers. (Array name, no default)
-dB 0 10 20	Specifies the decibel level for absolute values. If -dB is not specified or -dB 0 is specified, absolute values are computed on the linear scale. Default: 0
-info 0 1 2 3	Sets local information level. Default: 0
-help 0 1	Prints a help screen if set to 1. Default: 0

Returns

None.

Example

```
set Z(Re) [list 1 1 0 -1 -1 -1 0 1]
set Z(Im) [list 0 1 1 1 0 -1 -1 -1]
rfx::Abs_v -out absvals -z Z
puts "abs values = $absvals"
#-> abs values = 1.0 1.414 1.0 1.414 1.0 1.414 1.0 1.414

set Z(Re) [list 0 0 0 0 0]
set Z(Im) [list 1e0 1e1 1e2 1e3 1e4]
rfx::Abs_v -out dBs -z Z -dB 10

puts "dBs= $dBs"
#-> 0.0 10.0 20.0 30.0 40.0
```

Appendix H: Two-Port Network RF Extraction Library

Complex Arithmetic Support

rfx::Abs2_c

Computes the square of the absolute value of a complex number.

Syntax

```
rfx::Abs2_c z
```

Argument	Description
z	A list containing the real and imaginary parts of a complex number. (List of real numbers, no default)

Returns

A single value, the square of the absolute value of a complex number.

Example

```
set z [list 4 3]
puts [rfx::Abs2_c $z]
#-> 25
```

rfx::Abs2_v

Computes the square of the absolute value of a list of complex numbers.

Syntax

```
rfx::Abs2_v -out <list_name> -z <array_name>
[-info 0 | 1 | 2 | 3] [-help 0 | 1]
```

Argument	Description
-out <list_name>	Name of a list to store the list of square of absolute values. (List name, no default)
-z <array_name>	Name of an array containing list of complex numbers. The array has one string-valued index. The index contains the elements Re and Im. The values of the Re element and the Im element are the real and imaginary parts, respectively, of the complex numbers. (Array name, no default)
-info 0 1 2 3	Sets local information level. Default: 0
-help 0 1	Prints a help screen if set to 1. Default: 0

Appendix H: Two-Port Network RF Extraction Library

Complex Arithmetic Support

Returns

None.

Example

```
set Z(Re) [list 1 1 0 -1 -1 -1 0 1]
set Z(Im) [list 0 1 1 1 0 -1 -1 -1]
rfx::Abs2_v -out abs2 -z Z
puts "square of abs values = $abs2"
#-> square of abs values = 1.0 2.0 1.0 2.0 1.0 2.0 1.0 2.0
```

rfx::Add_c

Adds two complex numbers.

Syntax

```
rfx::Add_c z1 z2
```

Argument	Description
z1 z2	Two lists, each containing the real and imaginary parts of a complex number. (List of real numbers, no default)

Returns

A list containing the real and imaginary parts of the sum (a complex number).

Example

```
set z1 [list 1 0]
set z2 [list 0 1]
puts [rfx::Add_c $z1 $z2]
#-> 1 1
```

rfx::Add_v

Adds two lists of complex numbers.

Syntax

```
rfx::Add_v -out <array_name> -z1 <array_name> -z2 <array_name>
[-info 0 | 1 | 2 | 3] [-help 0 | 1]
```

Argument	Description
-out <array_name>	Name of an array to store the results. The array has one string-valued index. The index contains the elements Re and Im. The values of the Re element and the Im element are the real and imaginary parts, respectively, of the sum of the list of complex numbers specified using the keywords -z1 and -z2. (Array name, no default)
-z1 <array_name>	Name of an array containing a list of complex numbers. The array has one string-valued index. The index contains the elements Re and Im. The values of the Re element and the Im element are the real and imaginary parts of the complex numbers, respectively. (Array name, no default)
-z2 <array_name>	Name of an array containing a list of complex numbers. The array has one string-valued index. The index contains the elements Re and Im. The values of the Re element and the Im element are the real and imaginary parts of the complex numbers, respectively. (Array name, no default)
-info 0 1 2 3	Sets local information level. Default: 0
-help 0 1	Prints a help screen if set to 1. Default: 0

Returns

None.

Example

```
set Z1(Re) [list 0 1]
set Z1(Im) [list 1 2]
set Z2(Re) [list 1 2]
set Z2(Im) [list 3 4]
rfx::Add_v -out Z -z1 Z1 -z2 Z2
puts "Z(Re)= $Z(Re)"
puts "Z(Im)= $Z(Im)"
#-> Z(Re)= 1 3
#-> Z(Im)= 4 6
```

rfx::Cart2Polar_c

Converts a complex number from Cartesian to polar coordinates.

Syntax

```
rfx::Cart2Polar_c z
```

Argument	Description
z	A list containing the real and imaginary parts of a complex number. (List of real numbers, no default)

Returns

A list containing the absolute value and the phase of the complex number.

Example

```
set z [list 1 1]
set polar [rfx::Cart2Polar_c $z]
puts "abs value = [format %.3f [lindex $polar 0]]"
puts "phase = [lindex $polar 1]"
#-> abs value = 1.414
#-> phase = 45.0
```

rfx::Cart2Polar_v

Converts a list of complex numbers from Cartesian to polar coordinates.

Syntax

```
rfx::Cart2Polar_v -out <array_name> -z <array_name>
[-info 0 | 1 | 2 | 3] [-help 0 | 1]
```

Argument	Description
-out <array_name>	Name of an array to store the results. The array has one string-valued index. The index contains the elements Abs and Phase. The values of the Abs element and the Phase element are the absolute value and the phase, respectively, of the list complex numbers specified using the keyword -z. (Array name, no default)

Appendix H: Two-Port Network RF Extraction Library

Complex Arithmetic Support

Argument	Description
-z <array_name>	Name of an array containing a list of complex numbers. The array has one string-valued index. The index contains the elements Re and Im. The values of the Re element and the Im element are the real and imaginary parts, respectively, of the list of complex numbers. (Array name, no default)
-info 0 1 2 3	Sets local information level. Default: 0
-help 0 1	Prints a help screen if set to 1. Default: 0

Returns

None.

Example

```
set Z(Re) [list 1 1 0 -1 -1 -1 0 1]
set Z(Im) [list 0 1 1 1 0 -1 -1 -1]
rfx::Cart2Polar_v -out polar -z Z
puts "abs values = $polar(Abs)"
puts "phases = $polar(Phase)"
#-> abs values = 1.0 1.414 1.0 1.414 1.0 1.414 1.0 1.414
#-> phases = 0.0 45.0 89.999 135.0 180.0 -135.0 -89.999 -45.0
```

rfx::Conj_c

Computes the complex conjugate of a complex number.

Syntax

```
rfx::Conj_c z
```

Argument	Description
z	A list containing the real and imaginary parts of a complex number. (List of real numbers, no default)

Returns

A list containing the real and imaginary parts of the complex conjugate.

Appendix H: Two-Port Network RF Extraction Library

Complex Arithmetic Support

Example

```
set z [list 1 1]
puts [rfx::Conj_c $z]
#-> 1 -1
```

rfx::Conj_v

Computes the complex conjugate of a list of complex numbers.

Syntax

```
rfx::Conj_v -out <array_name> -z <array_name>
[-info 0 | 1 | 2 | 3] [-help 0 | 1]
```

Argument	Description
-out <array_name>	Name of an array to store the results. The array has one string-valued index. The index contains the elements Re and Im. The values of the Re element and the Im element are the real and imaginary parts, respectively, of the complex conjugate of the list of complex numbers specified using the keyword -z. (Array name, no default)
-z <array_name>	Name of an array containing a list of complex numbers. The array has one string-valued index. The index contains the elements Re and Im. The values of the Re element and the Im element are the real and imaginary parts, respectively, of the complex numbers. (Array name, no default)
-info 0 1 2 3	Sets local information level. Default: 0
-help 0 1	Prints a help screen if set to 1. Default: 0

Returns

None.

Example

```
set Z(Re) [list 1 1]
set Z(Im) [list -1 1]
rfx::Conj_v -out Conj -z Z
puts "Real part of complex conjugate= $Conj(Re)"
puts "Imaginary part of complex conjugate= $Conj(Im)"
#-> Real part of complex conjugate= 1 1
#-> Imaginary part of complex conjugate= 1 -1
```

rfx::Div_c

Divides two complex numbers.

Syntax

```
rfx::Div_c z1 z2
```

Argument	Description
z1 z2	Two lists, each containing the real and imaginary parts of a complex number. (List of real numbers, no default)

Returns

A list containing the real and imaginary parts of the quotient (a complex number).

Example

```
set z1 [list 4 0]
set z2 [list 0 2]
puts [rfx::Div_c $z1 $z2]
#-> 0.0 -2.0
```

rfx::Div_v

Divides two lists of complex numbers.

Syntax

```
rfx::Div_v -out <array_name> -z1 <array_name> -z2 <array_name>
[-info 0 | 1 | 2 | 3] [-help 0 | 1]
```

Argument	Description
-out <array_name>	Name of an array to store the results. The array has one string-valued index. The index contains the elements Re and Im. The values of the Re element and the Im element are the real and imaginary parts, respectively, of the quotient of the list of complex numbers specified using the keywords -z1 and -z2. (Array name, no default)
-z1 <array_name>	Name of an array containing a list of complex numbers. The array has one string-valued index. The index contains the elements Re and Im. The values of the Re element and the Im element are the real and imaginary parts of the complex numbers, respectively. (Array name, no default)

Appendix H: Two-Port Network RF Extraction Library

Complex Arithmetic Support

Argument	Description
-z2 <array_name>	Name of an array containing a list of complex numbers. The array has one string-valued index. The index contains the elements Re and Im. The values of the Re element and the Im element are the real and imaginary parts of the complex numbers, respectively. (Array name, no default)
-info 0 1 2 3	Sets local information level. Default: 0
-help 0 1	Prints a help screen if set to 1. Default: 0

Returns

None.

Example

```
set Z1(Re) [list 4 4]
set Z1(Im) [list 7 2]
set Z2(Re) [list 1 3]
set Z2(Im) [list -3 -1]
rfx::Div_v -out Z -z1 Z1 -z2 Z2
puts "Z(Re)= $Z(Re)"
puts "Z(Im)= $Z(Im)"
#-> Z(Re)= -1.7 1.0
#-> Z(Im)= 1.9 1.0
```

rfx::Im_c

Computes the imaginary part of a complex number.

Syntax

```
rfx::Im_c z
```

Argument	Description
z	A list containing the real and imaginary parts of a complex number. (List of real numbers, no default)

Returns

A single value, the imaginary part of a complex number.

Appendix H: Two-Port Network RF Extraction Library

Complex Arithmetic Support

Example

```
set z [list 1 2]
puts [rfx::Im_c $z]
#-> 2
```

rfx::Mul_c

Multiplies two complex numbers.

Syntax

```
rfx::Mul_c z1 z2
```

Argument	Description
z1 z2	Two lists, each containing the real and imaginary parts of a complex number. (List of real numbers, no default)

Returns

A list containing the real and imaginary parts of the product (a complex number).

Example

```
set z1 [list 1 0]
set z2 [list 0 1]
puts [rfx::Mul_c $z1 $z2]
#-> 0 1
```

rfx::Mul_v

Multiplies two lists of complex numbers.

Syntax

```
rfx::Mul_v -out <array_name> -z1 <array_name> -z2 <array_name>
[-info 0 | 1 | 2 | 3] [-help 0 | 1]
```

Argument	Description
-out <array_name>	Name of an array to store the results. The array has one string-valued index. The index contains the elements Re and Im. The values of the Re element and the Im element are the real and imaginary parts, respectively, of the product of the list of complex numbers specified using the keywords -z1 and -z2. (Array name, no default)

Appendix H: Two-Port Network RF Extraction Library

Complex Arithmetic Support

Argument	Description
-z1 <array_name>	Name of an array containing a list of complex numbers. The array has one string-valued index. The index contains the elements Re and Im. The values of the Re element and the Im element are the real and imaginary parts of the complex numbers, respectively. (Array name, no default)
-z2 <array_name>	Name of an array containing a list of complex numbers. The array has one string-valued index. The index contains the elements Re and Im. The values of the Re element and the Im element are the real and imaginary parts of the complex numbers, respectively. (Array name, no default)
-info 0 1 2 3	Sets local information level. Default: 0
-help 0 1	Prints a help screen if set to 1. Default: 0

Returns

None.

Example

```
set Z1(Re) [list 4 4]
set Z1(Im) [list 7 2]
set Z2(Re) [list 1 3]
set Z2(Im) [list -3 -1]
rfx::Mul_v -out Z -z1 Z1 -z2 Z2
puts "Z(Re)= $Z(Re)"
puts "Z(Im)= $Z(Im)"
#-> Z(Re)= 25 14
#-> Z(Im)= -5 2
```

rfx::Mulsc_c

Multiplies a scalar and a complex number.

Syntax

```
rfx::Mulsc_c c z
```

Argument	Description
c	A real number. (Real number, no default)
z	A list containing the real and imaginary parts of a complex number. (List of real numbers, no default)

Appendix H: Two-Port Network RF Extraction Library

Complex Arithmetic Support

Returns

A list containing the real and imaginary parts of the product (a complex number) of the scalar and the complex number.

Example

```
set c 5.0
set z [list 2.0 3.0]
puts [rfx::Mulsc_c $c $z]
#-> 10.0 15.0
```

rfx::Phase_c

Computes the principal value of the phase (in degrees, in the interval $(-180^\circ, 180^\circ]$) of a complex number specified in Cartesian coordinates.

Syntax

```
rfx::Phase_c z
```

Argument	Description
z	A list containing the real and imaginary parts of a complex number. (List of real numbers, no default)

Returns

A single value, the phase of a complex number.

Example

```
set z [list 1 1]
puts [rfx::Phase_c $z]
#-> 45.0
```

rfx::Phase_v

Computes the principal value of the phase (in degrees, in the interval $(-180^\circ, 180^\circ]$) of a list of complex numbers specified in Cartesian coordinates.

Syntax

```
rfx::Phase_v -out <list_name> -z <array_name>
[-info 0 | 1 | 2 | 3] [-help 0 | 1]
```

Argument	Description
-out <list_name>	Name of a list to store the list of phases. (List name, no default)
-z <array_name>	Name of an array containing a list of complex numbers. The array has one string-valued index. The index contains the elements Re and Im. The values of the Re element and the Im element are the real and imaginary parts, respectively, of the complex numbers. (Array name, no default)
-info 0 1 2 3	Sets local information level. Default: 0
-help 0 1	Prints a help screen if set to 1. Default: 0

Returns

None.

Example

```
set Z(Re) [list 1 1 0 -1 -1 -1 0 1]
set Z(Im) [list 0 1 1 1 0 -1 -1 -1]
rfx::Phase_v -out phases -z Z
puts "phases = $phases"
#-> phases = 0.0 45.0 89.999 135.0 180.0 -135.0 -89.999 -45.0
```

rfx::Polar2Cart_c

Converts a complex number from polar to Cartesian coordinates.

Syntax

```
rfx::Polar2Cart_c z
```

Argument	Description
z	A list containing the absolute value and the phase of a complex number. (List of real numbers, no default)

Returns

A list containing the real and imaginary parts of a complex number.

Example

```
set z [list 1.414 45]
set ReIm [rfx::Polar2Cart_c $z]
puts "Real part = [lindex $ReIm 0]"
puts "Imaginary part = [lindex $ReIm 1]"
#-> Real part = 0.999
#-> Imaginary part = 0.999
```

rfx::Polar2Cart_v

Converts a list of complex numbers from polar to Cartesian coordinates.

Syntax

```
rfx::Polar2Cart_v -out <array_name> -z <array_name>
[-info 0 | 1 | 2 | 3] [-help 0 | 1]
```

Argument	Description
-out <array_name>	Name of an array to store the results. The array has one string-valued index. The index contains the elements Re and Im. The values of the Re element and the Im element are the real and imaginary parts, respectively, of the list of complex numbers specified using the keyword -z. (Array name, no default)

Appendix H: Two-Port Network RF Extraction Library

Complex Arithmetic Support

Argument	Description
-z <array_name>	Name of an array containing a list of complex numbers. The array has one string-valued index. The index contains the elements Abs and Phase. The values of the Abs element and the Phase element are the absolute value and the phase (in degrees), respectively, of the list of complex numbers. (Array name, no default)
-info 0 1 2 3	Sets local information level. Default: 0
-help 0 1	Prints a help screen if set to 1. Default: 0

Returns

None.

Example

```
set Z(Abs) [list 1.414 1]
set Z(Phase) [list 45 60]
rfx::Polar2Cart_v -out ReIm -z Z
puts "Real part = $ReIm(Re)"
puts "Imaginary part = $ReIm(Im)"
#-> Real part = 0.999 0.5
#-> Imaginary part = 0.999 0.866
```

rfx::Re_c

Computes the real part of a complex number.

Syntax

```
rfx::Re_c z
```

Argument	Description
z	A list containing the real and imaginary parts of a complex number. (List of real numbers, no default)

Returns

A single value, the real part of a complex number.

Example

```
set z [list 1 2]
puts [rfx::Re_c $z]
#-> 1
```

rfx::Sign

Computes the sign of a real number.

Syntax

```
rfx::Sign r1
```

Argument	Description
r1	A real value. (Real number, no default)

Returns

A single value, the sign of a real number.

Example

```
puts [rfx::Sign -2]
#-> -1.0
```

rfx::Sub_c

Subtracts two complex numbers.

Syntax

```
rfx::Sub_c z1 z2
```

Argument	Description
z1 z2	Two lists, each containing the real and imaginary parts of a complex number. (List of real numbers, no default)

Returns

A list containing the real and imaginary parts of the difference (a complex number).

Example

```
set z1 [list 1 0]
set z2 [list 0 1]
puts [rfx::Sub_c $z1 $z2]
#-> 1 -1
```

rfx::Sub_v

Subtracts two lists of complex numbers.

Syntax

```
rfx::Sub_v -out <array_name> -z1 <array_name> -z2 <array_name>
[-info 0 | 1 | 2 | 3] [-help 0 | 1]
```

Argument	Description
-out <array_name>	Name of an array to store the results. The array has one string-valued index. The index contains the elements Re and Im. The values of the Re element and the Im element are the real and imaginary parts, respectively, of the difference of the list of complex numbers specified using the keywords -z1 and -z2. (Array name, no default)
-z1 <array_name>	Name of an array containing a list of complex numbers. The array has one string-valued index. The index contains the elements Re and Im. The values of the Re element and the Im element are the real and imaginary parts of the complex numbers, respectively. (Array name, no default)

Appendix H: Two-Port Network RF Extraction Library

Complex Arithmetic Support

Argument	Description
-z2 <array_name>	Name of an array containing a list of complex numbers. The array has one string-valued index. The index contains the elements Re and Im. The values of the Re element and the Im element are the real and imaginary parts of the complex numbers, respectively. (Array name, no default)
-info 0 1 2 3	Sets local information level. Default: 0
-help 0 1	Prints a help screen if set to 1. Default: 0

Returns

None.

Example

```
set Z1(Re) [list 1 1]
set Z1(Im) [list 0 2]
set Z2(Re) [list 0 2]
set Z2(Im) [list 1 4]
rfx::Sub_v -out Z -z1 Z1 -z2 Z2
puts "Z(Re)= $Z(Re)"
puts "Z(Im)= $Z(Im)"
#-> Z(Re)= 1 -1
#-> Z(Im)= -1 -2
```

lib::SetInfoDef

Sets the default information level.

Note:

Level 0: Warning, error, or status messages only.

Level 1: Echo results.

Level 2: Show progress and some debug information.

Level 3: Show all debug information.

The local information level also can be set using the `-info` keyword of the procedures in the RF extraction library.

Syntax

```
lib::SetInfoDef 0 | 1 | 2 | 3
```

Argument	Description
<code><info_level></code>	Sets the default information level. Default: 0

Returns

None.

Example

```
lib::SetInfoDef 2
```

References

- [1] H. Hillbrand and P. H. Russer, "An Efficient Method for Computer Aided Noise Analysis of Linear Amplifier Networks," *IEEE Transactions on Circuits and Systems*, vol. CAS-23. no. 4, pp. 235–238, 1976.
- [2] M. Reisch, *High-Frequency Bipolar Transistors: Physics, Modeling, Applications*, Berlin: Springer, 2003.
- [3] B. Razavi, *Design of Analog CMOS Integrated Circuits*, Boston: McGraw-Hill, 2001.
- [4] *Sentaurus™ Device User Guide*, Version T-2022.03, Mountain View, California: Synopsys, Inc., 2022.
- [5] R. S. Carson, *High-Frequency Amplifiers*, New York: John Wiley & Sons, 2nd ed., 1982.

Appendix H: Two-Port Network RF Extraction Library

References

- [6] R. Ludwig and G. Bogdanov, *RF Circuit Design: Theory and Applications*, Upper Saddle River, New Jersey: Prentice Hall, 2nd ed., 2009.
- [7] W. Liu, *Handbook of III-V Heterojunction Bipolar Transistors*, New York: John Wiley & Sons, 1998.
- [8] M. S. Gupta, "Power Gain in Feedback Amplifiers, a Classic Revisited," *IEEE Transactions on Microwave Theory and Techniques*, vol. 40. no. 5, pp. 864–879, 1992.
- [9] J. D. Cressler and G. Niu, *Silicon-Germanium Heterojunction Bipolar Transistors*, Boston: Artech House, 2003.
- [10] A. M. Niknejad, *Electromagnetics for High-Speed Analog and Digital Communication Circuits*, Cambridge: Cambridge University Press, 2007.
- [11] G. Gonzalez, *Microwave Transistor Amplifiers: Analysis and Design*, Upper Saddle River, New Jersey: Prentice Hall, 2nd ed., 1997.
- [12] V. Rizzoli and A. Lipparini, "Computer-Aided Noise Analysis of Linear Multiport Networks of Arbitrary Topology," *IEEE Transactions on Microwave Theory and Techniques*, vol. MTT-33. no. 12, pp. 1507–1512, 1985.
- [13] M. E. Mokari and W. Patience, "A New Method of Noise Parameter Calculation Using Direct Matrix Analysis," *IEEE Transactions on Circuits and Systems—I: Fundamental Theory and Applications*, vol. 39. no. 9, pp. 767–771, 1992.
- [14] F. Bonani and G. Ghione, *Noise in Semiconductor Devices, Modeling and Simulation*, Berlin: Springer, 2001.

PhysicalConstants Library

This appendix provides information about the PhysicalConstants library.

Major Physical Constants

This library defines a set of variables of major physical constants [1] (see [Table 42](#)).

Table 42 Variables defined in PhysicalConstants library

Name of variable	Value	Unit
AtomicMassConstant	1.660540210e-27	kg
AvogadroConstant	6.022136736e23	mol ⁻¹
BohrMagneton	9.274015431e-24	J/T
BoltzmannConstant	1.38065812e-23	J/K
ElectronMass	9.109389754e-31	kg
ElectronVolt	1.6021773349e-19	J
ElementaryCharge	1.6021773349e-19	C
FaradayConstant	9.648530929e4	C/mol
FineStructureConstant	7.2973530833e-3	1
FreeSpaceImpedance	376.730313462	Ω
GravitationConstant	6.6725985e-11	m ³ /kg/s ²
MagneticFluxQuantum	2.0678346161e-15	Wb

Appendix I: PhysicalConstants Library

Major Physical Constants

Table 42 Variables defined in PhysicalConstants library (Continued)

Name of variable	Value	Unit
MolarVolume	22.4141019e-3	m ³ /mol
Permeability	12.566370614e-7	H/m
Permittivity	8.854187817e-12	F/m
Pi	3.141592653589793	1
PlanckConstant	6.626075540e-34	Js
ProtonMass	1.672623110e-27	kg
RydbergConstant	1.097373153413e7	m ⁻¹
SpeedOfLight	299792458	m/s
StefanBoltzmannConstant	5.6705119e-8	W/m ² /K ⁴
kT300	0.0258521592446	V

To load the library, use the command:

```
load_library physicalconstants
```

The PhysicalConstants library uses a unique namespace identifier (`const::`) for its variables. All variables associated with this library are accessed with the namespace identifier prepended:

```
const::<var_name>
```

For example:

```
load_library physicalconstants
puts "c=$const::SpeedOfLight"
#-> c=299792458
```

The following conventions are used for the syntax of Tcl commands:

- Angle brackets – `<>` – indicate text that must be replaced, but they are *not* part of the syntax.

References

- [1] G. Woan, *The Cambridge Handbook of Physics Formulas*, Cambridge: Cambridge University Press, 2000.