Utilities User Guide

Version N-2017.09, September 2017

SYNOPSYS®

Copyright and Proprietary Information Notice

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

Contents

About This Guide

This user guide describes different utilities related to the Synopsys TCAD Sentaurus™ tools.

Related Publications

For additional information, see:

- The TCAD Sentaurus release notes, available on the Synopsys SolvNet® support site (see Accessing SolvNet on page v).
- Documentation available on SolvNet at https://solvnet.synopsys.com/DocsOnWeb.

Conventions

The following conventions are used in Synopsys documentation.

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

Customer Support

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

Accessing SolvNet

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

To access the SolvNet site:

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

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

Contacting Synopsys Support

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

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

Contacting Your Local TCAD Support Team Directly

Send an e-mail message to:

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

CHAPTER 1 The File datexcodes.txt

This chapter describes the datexcodes.txt file.

Overview

The file `datexcodes.txt` is the Synopsys configuration database for materials, doping species, and other quantities that are used in semiconductor process and device simulations. The database is referred to by various Synopsys TCAD software for different purposes. It does not contain the physical properties of materials or quantities, but the configuration properties such as names, colors, and labels. The file is divided into three sections: header, materials, and variables. Each material in the materials section and each quantity in the variables section are described by several properties that are explained here.

Header

The first four lines in the file `datexcodes.txt` consist of a header:

```
DATEX2.1
Datacode
"$Id$"
"Data codes for semiconductor process and device simulation"
```

The first two lines specify the version number and type of the file. The last two lines are strings containing comments.

Materials

A material, for example, silicon, can be specified as follows:

```
Silicon {
   label  = "Silicon"
   group  = Semiconductor
   color  = #ffb6c1
   alter1 = Si
   alter2 = 3
}
```

Table 1 describes the different fields.

Table 1 DATEX material specification

Field	Description
`alter1`	Name used for translation from and to SUPREM-4a.
`alter2`	Name used for translation from and to SUPREM-4b.
`color`	Color used for display (hexadecimal intensities for red, green, blue; default `#b0b0b0`).
`group`	Material classification (`Semiconductor`, `Conductor`, `Insulator`, or `All`; default `All`).
`label`	Name used for display purposes (default `Unknown`).

It is possible to declare multiple names (or aliases) for a material:

```
Vacuum, Gas, Ambient {
   ...
}
```

Alternative colors can be listed after the primary color, for example:

```
Silicon {
   ...
   color = #ffb6c1, #ac3320, #f18010
   ...
}
```

Variables

A variable, for example, `ElectrostaticPotential`, can be specified as follows:

```
ElectrostaticPotential {
   label     = "electrostatic potential"
   symbol    = "u"
   unit      = "V"
   factor    = 1.0e+00
   precision = 7
   interpol  = linear
   material  = All
   alter1    = v
   alter2    = 100
   property("floops") = "Potential"
}
```

Table 2 describes the different fields.

Table 2 DATEX variable specification

Field	Description
alter1	Name used for translation from and to SUPREM-4a.
alter2	Name used for translation from and to SUPREM-4b.
arsinh	Scaling factor for arsinh interpolation mode (default 10^{14}).
doping	Specification of doping species.
factor	Scaling factor (default 1).
interpol	Interpolation mode (linear, log, or asinh; default linear).
label	Name used for display purposes (default undefined).
material	Specifies the validity (domain of definition) of this quantity (Semiconductor, Conductor, Insulator, or All; default All).
parity	Symmetry property of tensors (+1 or -1; default +1).
precision	Number of significant digits (used in graphics tools; default 7).
property	Tool-specific variable properties.
symbol	Symbol used for display (default ?).
unit	Unit used for display and data exchange (default 1).

You can declare multiple names (or aliases) for a variable, for example:

```
BoronConcentration, BoronChemicalConcentration {
   ...
}
```

The following keywords are also valid for the precision field: half, single, and double. They correspond to the numeric values of 3, 7, and 14.

The field material can specify multiple materials or material groups:

```
material = Semiconductor, Insulator
material = Silicon, PolySilicon, Germanium
```

Doping Specification

Doping species are identified by the `doping` field:

```
CarbonDoping {
   doping = acceptor (
      active = CarbonActiveDoping
      ionized = CarbonIonizedDoping
      material = GaN
   )
   doping = donor (
      active = CarbonActiveDoping
      ionized = CarbonIonizedDoping
      material = SiliconGermanium, Silicon
   )
}
```

The `doping` field indicates whether the variable is an acceptor or a donor. The `active` field links a chemical doping concentration to its corresponding active doping concentration. Similarly, the `ionized` field links a chemical doping concentration to its corresponding ionized concentration.

The definition of a doping species may be limited to a list of substrate materials by the `material` field. By default, a doping species is defined for all substrate materials.

See Sentaurus™ Device User Guide, Doping Specification on page 11 for more information.

Search Strategy

Multiple `datexcodes.txt` files can be used. The following search strategy is observed:

- `$STROOT_LIB/datexcodes.txt` or `$STROOT/tcad/$STRELEASE/lib/datexcodes.txt` if the environment variable `STROOT_LIB` is not defined (lowest priority).
- `$HOME/datexcodes.txt` (medium priority).
- `datexcodes.txt` in local directory (highest priority).

Definitions in later files replace or add to the definitions in the earlier files. In this way, the local file only needs to contain materials or variables that you want to add or modify.

CHAPTER 2 TCAD Logfile Browser

This chapter discusses the TCAD Logfile Browser.

Overview

The TCAD Logfile Browser enables you to access information in a TCAD log file in a highly efficient manner. It displays the structure of the log file content in the form of an interactive Table of Contents, in which you can expand or collapse each subsection to see more details or to gain an overview.

The Table of Contents is shown in the lower-left panel as a tree of section tags, where each section tag is a button. When you click a section tag button, the log file content corresponding to this section is displayed in the main panel.

Immediately above the Table of Contents, the buttons of the Info Level Selector panel affect the content shown in the main panel. The buttons allow you to filter out all log file content that belongs to an information level higher than the one noted on the button.

The Active Tags panel shows an ordered list of all section tags contained in the section tag that is chosen in the Table of Contents.

Launching the TCAD Logfile Browser

The TCAD Logfile Browser is available for log files generated using Sentaurus Process, Sentaurus Interconnect, and Sentaurus Device, Version K-2015.06 or later, if the command-line option `--xml` is used.

In this case, these TCAD Sentaurus tools write the version of the file that contains XML-like tags. The marked-up version of the log file has the `*.xml` extension.

For example, the following call to Sentaurus Process creates the marked-up log file `n1_fps_log.xml`:

```
> sprocess --xml n1_fps.cmd
```

Launch the TCAD Logfile Browser with, for example:

```
> logbrowser n1_fps_log.xml
```

The XML tag information is preprocessed and optimized for efficient display in a Web browser. The TCAD Logfile Browser calls the Web browser that you selected in the Sentaurus Workbench preferences.

NOTE Depending on the size of the log file and the number of tags it contains, this preprocessing stage may take a few seconds.

During the preprocessing stage, the TCAD Logfile Browser provides information about the progress of the preprocessing and about the action taken. This information is written to the *standard output* pipe. If you launched the TCAD Logfile Browser directly from the command line of a terminal, the output is shown in that terminal. If you launched it from Sentaurus Workbench, the output is shown in the terminal from which Sentaurus Workbench was launched.

You can control the verbosity level of the output during the preprocessing stage with the `-info` option. The value of this option can be either 0, 1, 2, or 3. For example, to increase the verbosity to level 1, use:

```
> logbrowser -info 1 n1_fps_log.xml
```

NOTE The `-info` option controls only the output of the TCAD Logfile Browser during preprocessing and does not influence the content of the log file itself.

After the preprocessing is completed, the default Web browser launches automatically. The preprocessed marked-up version of the log file is saved with the `*.html` extension. If you want to reload a log file that has been preprocessed already, you can call the Web browser directly with, for example:

```
> firefox n1_fps_log.html
```

Graphical User Interface

The graphical user interface of the TCAD Logfile Browser consists of different areas (see Figure 1).

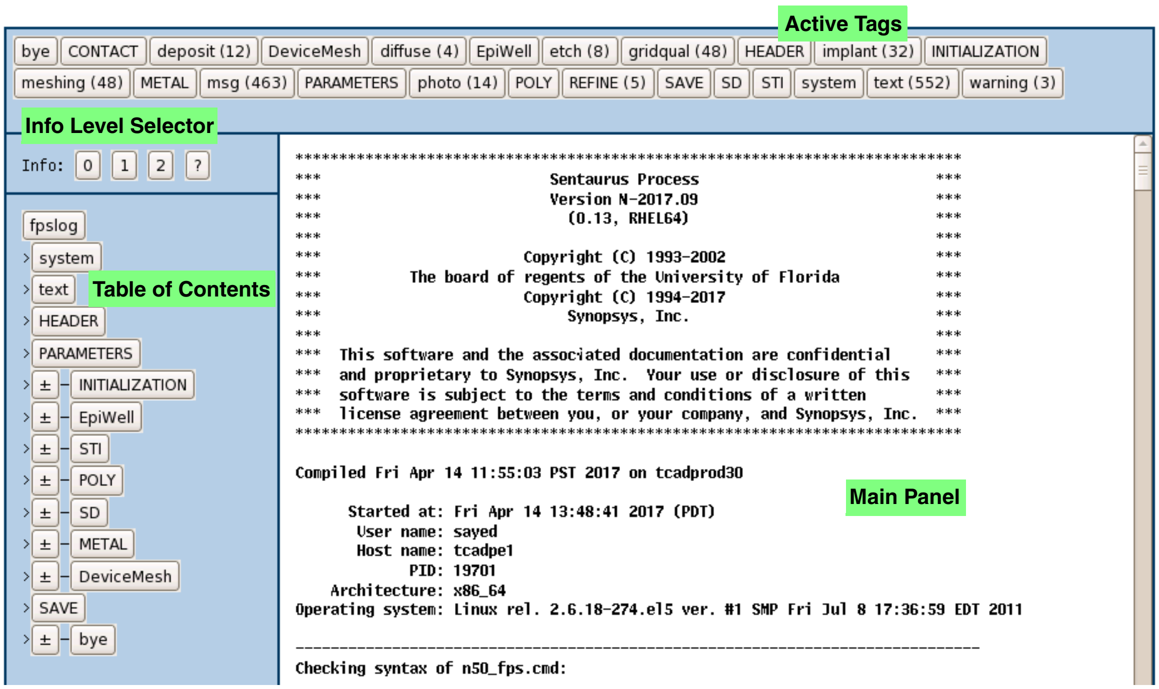


Figure 1 User interface of the TCAD Logfile Browser

Table of Contents

The Table of Contents shows the structure of the log file as a tree of section tags. Each section tag is a button.

If a particular section tag contains other section tags, the ± button is shown to the left of the section tag button and is used to expand or collapse the list of the contained section tags.

The number of angle brackets displayed to the left of each section tag button shows the level of containment. All section tags are contained in the root section tag, which for a Sentaurus Process log file is **fpslog**. For example, a **text** block that is part of the log file output of an **etch** command shows two angle brackets.

When you click a section tag button, the log file content corresponding to this section is displayed in the main panel.

When you click a word in the main panel, all section tag buttons leading to the sections that contain the clicked word are shown in bold. This allows you to quickly locate the respective section tags in the Table of Contents.

Active Tags Panel

The Active Tags panel shows an ordered list of all section tags contained in the section tag that is selected in the Table of Contents. Initially, the selected tag is the root section tag and, therefore, all section tags of the log file are displayed. If a particular section tag is found more than once in the selected section of the log file, the number of occurrences is shown to the right of the section tag name in parentheses.

Clicking a section tag button restricts the display in the main panel to only the log file content that belongs to that section tag. You can choose multiple section tags at a time. Selected buttons become gray. You unselect them by clicking the section tag buttons again.

You can use this feature, for example, to look at all the mesh quality messages and to find the message that showed a sudden increase in the mesh count. Then, you can click the text in that message to find its location in the Table of Contents, thereby finding which geometric operation triggered this change in mesh.

Info Level Selector Panel

Each section tag is associated with an information level. The buttons of the Info Level Selector panel let you filter out of the main panel all log file content that belongs to an information level higher than the one noted on the button. The selector button for a given information level is shown only if the log file contains information tagged for that level.

For example, in Figure 1 on page 7, the button for info level 3 is missing because, in the Sentaurus Process input file, the info level was set to 2 and, therefore, the log file does not contain any information tagged for info level 3.

Clicking the **1** button filters out all content belonging to higher information levels. Clicking the **0** button restores the display to the content for all information levels.

The help (**?**) button of this panel opens this content.

Main Panel

The main panel shows the selected sections of the log file. Initially, this panel shows the entire content of the log file. The content is shown using different foreground and background colors. The background color depends on the information level. For example, all content belonging to info level 0 is shown in black, while content belonging to info level 1 is shown in blue.

Specifically marked sections of the log file are displayed with a different background color. For example, for Sentaurus Process, content from the `implant` command is displayed with a purple background. The color scheme can be customized (see Custom Color Schemes on page 10).

Clicking text in the main panel highlights all section tag buttons that are associated with this text by making the text label bold. This allows you to find out more about the context of specific text you found in the main panel. For example, if you find log messages from a meshing operation, you may want to know which `etch` command triggered this remeshing step.

Integration in Sentaurus Workbench

To use the TCAD Logfile Browser in Sentaurus Workbench, you must first activate the `--xml` option for the respective TCAD tool. For example, open the Tool Properties dialog box for a Sentaurus Process tool instance. On the **Tool Properties** tab, add `--xml` in the **Command Line** field.

You can launch the TCAD Logfile Browser directly from the Node Explorer of Sentaurus Workbench by double-clicking the `*.xml` file.

To reopen already processed log files, double-click the `*.html` file.

Cleaning up the nodal output will remove both the `*.xml` file and the `*.html` file.

Custom Markups for Sentaurus Process

You can add custom section tags to Sentaurus Process log files to mark important processing units such as the gate stack definition or the contact formation.

To insert a main section tag, use the `Section` command:

```
Section tag=<c> [title=<c>]
```

For example:

```
Section tag= EpiWell title= Creation of the Epitaxial Well
...
Section tag= STI
```

When using the `Section` command, the section tag terminates automatically when the next section tag is encountered.

To add subsections, use the `SubSection.Start` command and the `SubSection.End` command, for example:

```
Section tag= EpiWell title= Creation of the Epitaxial Well
...
SubSection.Start tag= REFINE title= Global Refinement
...
SubSection.End tag= REFINE
...
Section tag= STI
```

Section tags for the main Sentaurus Process commands `deposit`, `diffuse`, `etch`, and `implant` are added automatically. In addition, important sections containing mesh and grid quality information as well as version and system information are tagged automatically. The Sentaurus Process ending message that contains the CPU run-time report and a summary of all warnings is contained in the **bye** tag. Warning messages are tagged as **warning**.

All messages sent at information levels other than 0 are contained in **msg** tags. Log file content that is not otherwise contained in a tag is wrapped in a **text** tag.

Custom Color Schemes

To customize the color scheme, copy the cascading style sheet `$STROOT/tcad/$STRELEASE/lib/logbrowser/logbrowser.css` to your local project directory, and edit it as required.

For example, to alter the background color of text tagged with **etch** to a light gray, change the setting for the background color to:

```
span.etch {
  background-color: #cccccc;
  display:     block; }
```

Limitations

Only log files generated with Version K-2015.06 (or later versions) of Sentaurus Process, Sentaurus Interconnect, or Sentaurus Device can be visualized with the TCAD Logfile Browser.

The TCAD Logfile Browser was designed for Firefox version 3.6.18, the default Web browser that ships with Red Hat Enterprise Linux v5.

The TCAD Logfile Browser was tested with later versions of Firefox (including versions 16.0 and 33.1.1), Chrome 33.0, and Internet Explorer 11.

NOTE For Internet Explorer 11, the TCAD Logfile Browser requires that ActiveX is enabled.

CHAPTER 3 Sentaurus spice2sdevice

This chapter discusses the Sentaurus spice2sdevice utility.

The Sentaurus `spice2sdevice` utility converts a subset of Synopsys HSPICE® netlist files into equivalent circuit files of Sentaurus Device. HSPICE netlist files (extension `.cir`) are documented in the *HSPICE® User Guide: Basic Simulation and Analysis*. The circuit files of Sentaurus Device (extension `.scf`) are discussed in the *Sentaurus™ Device User Guide*.

NOTE Sentaurus Device also can read HSPICE netlist files directly in the `System` section. This is discussed in the *Sentaurus™ Device User Guide*.

HSPICE Netlist Files

The first line of a netlist file is assumed to be a title line and is ignored:

```
.TITLE 'amplifier netlist'
```

The title line is followed by a sequence of HSPICE statements, and the netlist is terminated by an optional `.END` statement:

```
.END
```

Everything after the final `.END` statement is ignored.

The command-line option `-m` must be used if no title line is present (HSPICE model files).

The netlist parser is case insensitive, except for string literals or file names in `.INCLUDE` statements:

```
.PARAM s = str('This is a case sensitive string.')
.INCLUDE 'Case/Sensitive/Filename'
```

Comments

A line starting with either the dollar sign ($) character or the asterisk (*) character is a comment line, for example:

```
* This is a comment.
```

You can use in-line comments after the `$` character:

```
R1 1 2 R=100 $ drain resistor
```

Continuation Lines

Use the plus sign (+) character in the first column to indicate a continuation line:

```
R1 1 0
+ R=500
```

The INCLUDE Statement

Use the `.INCLUDE` statement to include another netlist in the current netlist:

```
.INCLUDE models.sp
```

Numeric Constants

You can enter numbers in one of the following formats:

- Integer (for example, `7`)
- Floating point (for example, `-4.5`)
- Floating point with an integer exponent (for example, `3e8` and `-1.2e9`)
- Integer with a scale factor listed in Table 3 (for example, `6k`)
- Floating point with a scale factor listed in Table 3 (for example, `-8.9meg`)

Table 3 Scale factors

Scale factor	Description	Multiplying factor
`t`	tera	10^{12}
`g`	giga	10^{9}
`meg` or `x`	mega	10^{6}
`k`	kilo	10^{3}
`m`	milli	10^{-3}
`mil`	one-thousandth of an inch	$25.4 \cdot 10^{-6}$
`u`	micro	10^{-6}
`n`	nano	10^{-9}

Table 3 Scale factors

Scale factor	Description	Multiplying factor
p	pico	10^{-12}
f	femto	10^{-15}
a	atto	10^{-18}

NOTE The scale factor `a` is not a scale factor in a character string that contains `amps`. For example, the expression `20amps` is interpreted as 20 amperes of current, not as `20e-18mps`.

Parameters and Expressions

In the HSPICE tool, parameters are names that you associate with a value. Numeric and string parameters are supported:

```
.PARAM a = 4
.PARAM b = '2*a + 7'
.PARAM s = str('This is a string')
.PARAM t = str(s)
```

The following built-in mathematical functions are supported:

sin, cos, tan, asin, acos, atan, sinh, cosh, tanh, abs, sqrt, pow, pwr, log, log10, exp, db, int, nint, sgn, sign, floor, ceil, min, max

Subcircuits

Reusable cells can be specified as subcircuits. The general definition is given by:

```
.SUBCKT name n1 n2 ... [param1=val] [param2=val] ...
.ENDS
```

or:

```
.MACRO name n1 n2 ... [param1=val] [param2=val] ...
.EOM
```

String parameters are supported as well:

```
.SUBCKT name n1 n2 ... [param=str('string')] ...
.ENDS
```

Examples:

```
.PARAM P5=5 P2=10

.SUBCKT SUB1 1 2 P4=4
R1 1 0 P4
R2 2 0 P5
X1 1 2 SUB2 P6=7
X2 1 2 SUB2
.ENDS

.MACRO SUB2 1 2 P6=11
R1 1 2 P6
R2 2 0 P2
.EOM

X1 1 2 SUB1 P4=6
X2 3 4 SUB2 P6=15
```

Model Statements

A `.MODEL` statement has the following general syntax:

```
.MODEL model_name type [level=num] [pname1=val1] [pname2=val2] ...
```

Table 4 lists the recognized model types.

Table 4 Model types

Type	Description	Type	Description
c	Capacitor model	npn	NPN BJT model
csw	Current-controlled switch	pjf	P-channel JFET model
d	Diode model	pmf	P-channel MESFET
l	Mutual inductor model	pmos	P-channel MOSFET model
njf	N-channel JFET model	pnp	PNP BJT model
nmf	N-channel MESFET	r	Resistor model
nmos	N-channel MOSFET model	sw	Voltage-controlled switch

Examples:

```
.MODEL mod1 NPN BF=50 IS=1e-13 VFB=50 PJ=3 N=1.05

.MODEL mod2 PMOS LEVEL=72
+ aigbinv = 0.0111
+ at = -0.00156
```

Table 5 lists the values for the parameter `level` in MOSFET models that are recognized. In the case of Level 1, 2, and 3, the corresponding device may be either an HSPICE MOSFET (`HMOS_L1`, `HMOS_L2`, or `HMOS_L3`) or a Berkeley SPICE MOSFET (`Mos1`, `Mos2`, or `Mos3`). By default, the Sentaurus `spice2sdevice` utility selects the HSPICE MOSFET, but the command-line option `-b` can be used to switch to a Berkeley SPICE MOSFET.

Table 5 SPICE MOSFET models

Level	Device	Description
1	`HMOS_L1 or Mos1`	Shichman–Hodges
2	`HMOS_L2 or Mos2`	Grove–Frohman
3	`HMOS_L3 or Mos3`	Empirical model
4	`BSIM1`	BSIM
5	`BSIM2`	BSIM2
6	`Mos6`	MOS6
8	`BSIM3`	BSIM3
9	`B3SOI`	Partially depleted SOI MOSFET model
14	`BSIM4`	BSIM4
28	`HMOS_L28`	Modified BSIM model
49	`HMOS_L49`	BSIM3v3 MOS model
53	`HMOS_L53`	BSIM3v3 MOS model
54	`HMOS_L54`	BSIM4 model
57	`HMOS_L57`	UC Berkeley BSIM3-SOI model
59	`HMOS_L59`	UC Berkeley BSIM3-SOI fully depleted (FD) model
61	`HMOS_L61`	RPI a-Si TFT model
62	`HMOS_L62`	RPI Poly-Si TFT model
64	`HMOS_L64`	STARC HiSIM model
68	`HMOS_L68`	STARC HiSIM2 model
69	`HMOS_L69`	PSP100 DFM support series model
72	`HMOS_L72`	BSIM-CMG multigate MOSFET model
73	`HMOS_L73`	STARC HiSIM-LDMOS/HiSIM-HV model
76	`HMOS_L76`	LETI-UTSOI MOSFET model

Elements

Element names must begin with a specific letter for each element type. Table 6 lists the HSPICE element types that are supported.

Table 6 HSPICE element types

First letter	Element	Example
c	Capacitor	`Cbypass 1 0 10pf`
d	Diode	`D7 3 9 D1`
e	Voltage-controlled voltage source	`Ea 1 2 3 4 K`
f	Current-controlled current source	`Fsub n1 n2 vin 2.0`
g	Voltage-controlled current source	`G12 4 0 3 0 10`
h	Current-controlled voltage source	`H3 4 5 Vout 2.0`
i	Current source	`IA 2 6 1e-6`
j	JFET or MESFET	`J1 7 2 3 model_jfet w=10u l=10u`
k	Linear mutual inductor	`K1 L1 L2 0.98`
l	Linear inductor	`Lx a b 1e-9`
m	MOS transistor	`M834 1 2 3 4 N1`
q	Bipolar transistor	`Q5 3 6 7 8 pnp1`
r	Resistor	`R10 21 10 1000`
v	Voltage source	`V1 8 0 5`
x	Subcircuit call	`X1 2 4 17 31 MULTI WN=100 LN=5`

Table 7 lists the Berkeley SPICE models that also are recognized [1].

Table 7 Berkeley SPICE element types

First letter	Element	Example
s	Voltage-controlled switch	`S1 1 2 3 4 SWITCH1 ON`
w	Current-controlled switch	`W1 1 2 VCLOCK SWITCHMOD1`
z	GaAs MESFET	`Z1 7 2 3 ZM1 AREA=2`

Netlist Commands

A limited set of netlist commands is recognized.

To make node names global across all subcircuits, use a `.GLOBAL` statement:

```
.GLOBAL node1 node2 node3 ...
```

Use the `.OPTION PARHIER` statement to specify scoping rules:

```
.OPTION PARHIER=GLOBAL|LOCAL
```

Other HSPICE netlist commands that have not been already mentioned explicitly are ignored.

Command-Line Options

Table 8 lists the command-line options that the Sentaurus `spice2sdevice` utility recognizes.

Table 8 Command-line options

Option	Description
`-b`	Use Berkeley SPICE models instead of HSPICE (applies only to MOSFETs Level 1, 2, and 3).
`-c`	Translates a SPICE circuit file (default).
`-d`	Prints additional debug information.
`-h`	Displays a help message.
`-m`	Translates a SPICE model file.
`-o filename`	Stores the Sentaurus Device circuit file in `filename`.
`-v`	Shows version information.

A SPICE model file is assumed to have no title line. Otherwise, it is identical to a SPICE circuit file.

Inverter Example

A simple resistor transistor logic (RTL) inverter as shown in Figure 2 is considered.

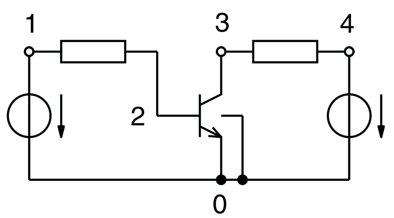


Figure 2 Simple RTL converter

This circuit can be described by the following SPICE circuit file (`rtl.cir`):

```
SIMPLE RTL INVERTER
VCC 4 0 5
VIN 1 0 PULSE 0 5 2NS 2NS 2NS 30NS 100NS
RB 1 2 10K
Q1 3 2 0 Q1
RC 3 4 1K
.PLOT DC V(3)
.PLOT TRAN V(3) (0,5)
.PRINT TRAN V(3)
.MODEL Q1 NPN BF 20 RB 100 TF .1NS CJC 2PF
.DC VIN 0 5 0.1
.TRAN 1NS 100NS
.END
```

The command:

```
spice2sdevice -o rtl.scf rtl.cir
```

produces the following output file (`rtl.scf`):

```
PSET q1
  DEVICE BJT
  PARAMETERS
    bf = 20
    cjc = 2e-12
    npn = 1
    pnp = 0
    rb = 100
    tf = 1e-10
END PSET
```

```
INSTANCE q1
  PSET q1
  ELECTRODES
    3  2  0  0
  PARAMETERS
END INSTANCE

INSTANCE rb
  PSET Resistor_pset
  ELECTRODES
    1  2
  PARAMETERS
    resistance = 10000
END INSTANCE

INSTANCE rc
  PSET Resistor_pset
  ELECTRODES
    3  4
  PARAMETERS
    resistance = 1000
END INSTANCE

INSTANCE vcc
  PSET Vsource_pset
  ELECTRODES
    4  0
  PARAMETERS
    dc = 5
END INSTANCE

INSTANCE vin
  PSET Vsource_pset
  ELECTRODES
    1  0
  PARAMETERS
    pulse = [0  5  2e-09  2e-09  2e-09  3e-08  1e-07]
END INSTANCE
```

The following command file of Sentaurus Device can then be used to perform a transient simulation:

```
File {
  SpicePath = "."
}

System {
  Plot "rtl.plt" (time()  v(1)  v(3))
}

Solve {
  Set (vcc."dc" = 0)

  Quasistationary (Goal {Parameter=vcc."dc"  Value=5})
    { Coupled { Circuit } }

  NewCurrentPrefix = "new_"

  Transient (InitialTime=0  FinalTime=100e-9
             InitialStep=1e-9  MaxStep=1e-9)
    { Coupled { Circuit } }
}
```

Figure 3 shows the voltages v_1 and v_3 as a function of time.

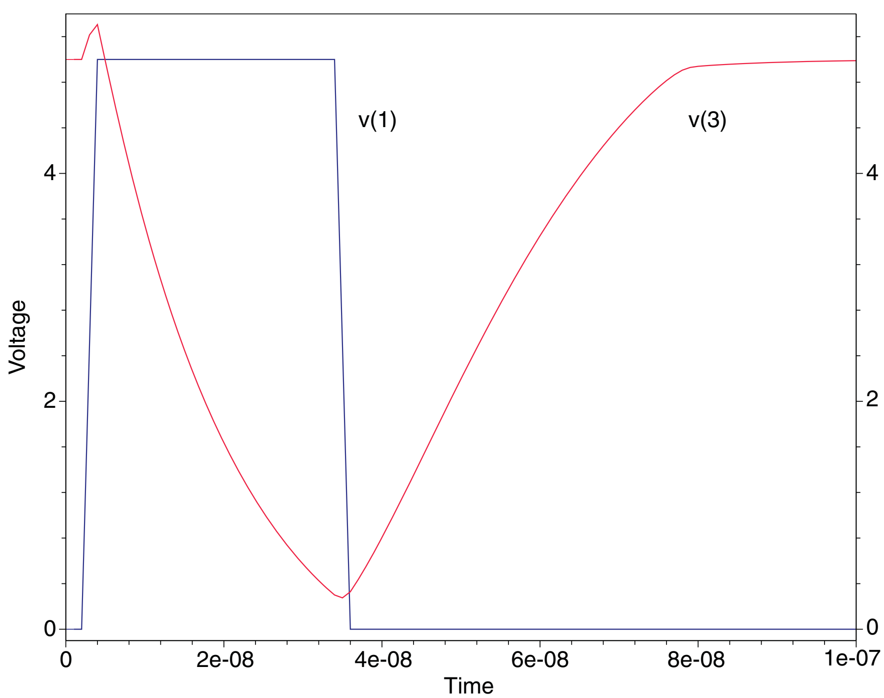


Figure 3 Transient simulation of simple RTL inverter

Subcircuit Example

The Sentaurus `spice2sdevice` utility supports basic SPICE subcircuits. As an example, consider the chain of low pass filters shown in Figure 4.

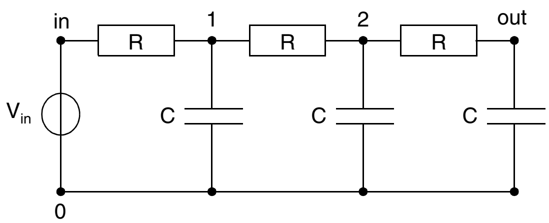


Figure 4   Sample chain of low pass filters

The following SPICE command file analyzes the transient response of this network to a pulse signal (file `filter.sp`):

```
low pass filter

.subckt filter 1 2 g
r1 1 2 100
c1 2 g 5n
.ends

vin in 0 pulse (0 5 1u 0.5u 0.5u 1u 4u)
x1 in 1 0 filter
x2 1 2 0 filter
x3 2 out 0 filter

.tran 10n 12u
.print tran v(in) v(1) v(2) v(out)

.end
```

To run the same simulation in Sentaurus Device, an equivalent `.scf` circuit file must be generated (file `filter.scf`):

```
spice2sdevice -o filter.scf filter.sp
```

The following Sentaurus Device command file then performs the same transient simulation as the SPICE command file above:

```
File {
   Output = "filter"
   SPICEPath = "."
}
```

```
System {
   Plot "filter_des.plt" (time() v(in) v(1) v(2) v(out))
}

Solve {
   Transient (InitialTime = 0 FinalTime = 12u
               InitialStep = 10n MaxStep = 10n MinStep = 1n) {
      Coupled { Circuit }
   }
}
```

Figure 5 shows the resulting voltages v_{in}, v_1, v_2, and v_{out} as a function of time.

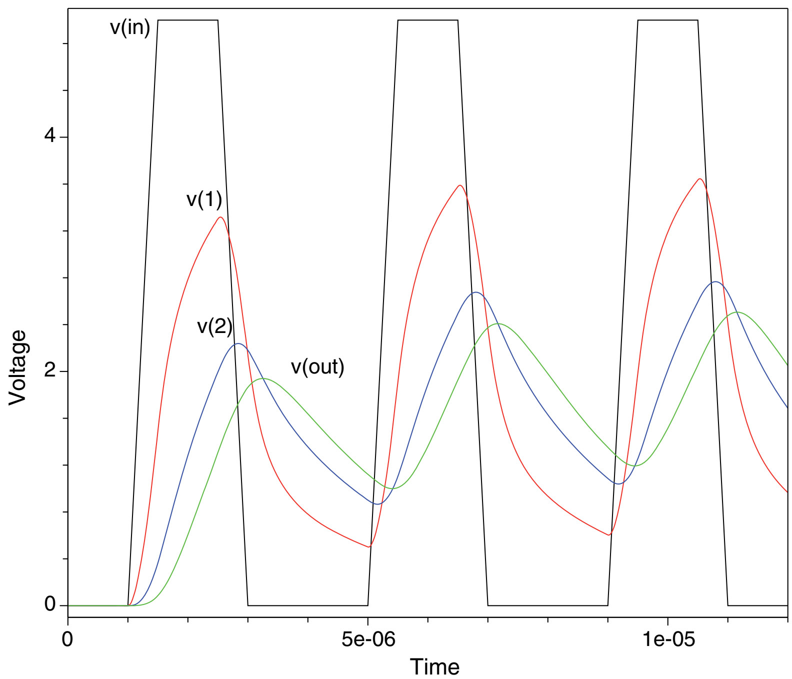


Figure 5   Transient simulation of a low pass filter

References

[1] T. Quarles *et al.*, *SPICE 3 Version 3F5 User's Manual*, Department of Electrical Engineering and Computer Sciences, University of California, Berkeley, CA, USA, 1994.

CHAPTER 4 Box Method Utility

This chapter describes the box method utility.

Basic Use

The box method utility is used to analyze the quality of a given mesh. There are two versions of this utility:

- *double* precision (`boxmethod`)
- *long double* precision (`boxmethodl`)

The box method utility reads a TDR mesh and reports various measures for the mesh quality. The result is a TDR data file that contains mesh information (for example, *`GridFile`*`_bxm.tdr`).

The `boxmethodl` version computes the box method parameters with long double precision, after which the data is converted to double precision, and the output TDR file contains only double precision information.

The syntax for the box method utility is either:

```
boxmethod [options] GridFile
```

or:

```
boxmethodl [options] GridFile
```

where *`GridFile`* is a TDR file.

The options are:

`-a Algorithm`	`Algorithm` is:
	`AverageBoxMethod`
	`CVPL_AverageBoxMethod` (default)
	`TruncatedVoronoiBox`
	`RegionBoundaryTruncatedVoronoiBox`
	`MaterialBoundaryTruncatedVoronoiBox`

`-h`	Show this help message and exit

`-NoGas`	Without computation in Gas
`-numThreads n`	Parallel computation where `n` is the number of threads (default value is `1`)
`-StackSize n`	Parallel computation where `n` is the size of the stack (default value is `0`)
`-v`	Print header with version number

For detailed descriptions, see Sentaurus™ Device User Guide, Chapter 37 on page 1011.

Basic Definitions

Obtuse Element

An element is called *obtuse* if the center of the circumsphere (circumcircle) is outside this element.

Obtuse Face

Let `Pf` be the plane that contains the face `f` of an element. Each plane splits 3D space into two half-spaces `Sf1` and `Sf2`. A face `f` is called *obtuse* if the center of the circumsphere of the element and the element itself lie in different half-spaces `Sf1`, `Sf2`.

NOTE In the 2D case, an obtuse triangle has only one obtuse edge.

NOTE In the 3D case:

- An obtuse prism has only one obtuse face.
- An obtuse tetrahedron has one or two obtuse faces.
- An obtuse pyramid has one or two or three obtuse faces.

Non-Delaunay Element

An obtuse element is called *non-Delaunay* if the interior of the circumsphere (circumcircle) around this element contains another mesh vertex.

Flat Element

Let α_i be an angle between faces for edge i (in the 2D case, between edges for vertex i). An element is called *flat* if for all angles (a tetrahedron has six angles) $\alpha_i < 10^{-8}$ (the angle is close to zero) or $\alpha_i > \pi - 10^{-8}$ (the angle is close to π).

Non-Delaunay Measure

A non-Delaunay measure is shown in Figure 6.

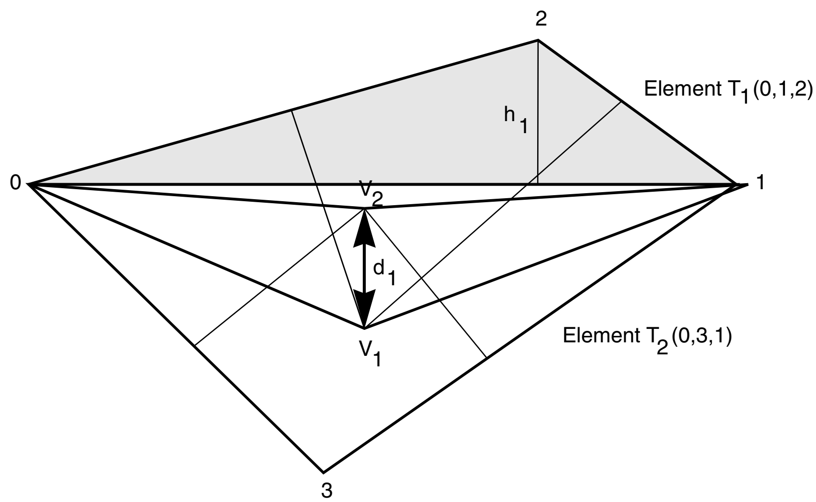


Figure 6 Schematic of non-Delaunay measure

A description of the elements are:

- $V_{1,2}$ is the Voronoï center (center of circumcircle) of elements $T_{1,2}$.
- h_1 is the height of T_1.
- T_1 is the non-Delaunay element.
- T_2 is the Delaunay element.

There are three definitions of a non-Delaunay measure:

1. $\delta_1 = d_1 = \text{Length}(V_1, V_2), [\mu\text{m}]$. For Delaunay element T_2, the value is $d_2 = 0$.
2. $\delta_2 = \dfrac{\text{Area}(0, V_1, 1, V_2, 0)}{\text{Area}(T_1)} = \dfrac{d_1}{h_1}$. In the 3D case, Area = Volume.
3. For 2D only: $\delta_3 = \dfrac{d_1}{\text{Length}(0, 1)}$, delta of coefficients for obtuse edge (0,1).

The $\mathrm{Area}(0, V_1, 1, V_2, 0)$ is called the non-Delaunay volume for element T_1. A non-Delaunay measure is defined for each element. In this example for Delaunay element T_2, the value is $d_2 = 0$, which means all non-Delaunay measures δ_k are equal to zero.

Tetrahedron Quality

The box method utility has the following tetrahedron (triangle) quality criteria:

- `TetQualityEdge` $= \frac{R}{L_{min}}$
- `TetQualityHeight` $= \frac{R}{H_{min}}$

where R is the radius of a circumscribed sphere (a circle in two dimensions) around an element, L_{min} is the length of the shortest edge of an element, and H_{min} is the shortest height of an element.

The box method utility saves the maximum values of `TetQualityEdge` and `TetQualityHeight` in the log file.

Log File

Region Non-Delaunay Elements

A log file contains common data about the mesh and information about non-Delaunay elements per region (for Delaunay mesh `DeltaVolume=0` and `non-DelaunayVolume=0`):

```
/-------- Region non-Delaunay elements -------------------------------------------------------------
 Region        Volume    BoxMethodVolume  DeltaVolume  Elements  non-Delaunay    non-DelaunayVolume
  name          [um2]        [um2]            [%]                  Elements        [um2]      [%]
 ----------------------------------------------------------------------------------------------------
 Nitride   1.9500000e-04  2.2635574e-04    16.080         53    12 (22.64 %)   1.8215e-04 (1.1e-05)
   . . .
 Oxide     6.0618645e-03  8.0705629e-03    33.137       2500   818 (32.72 %)   2.3715e-04 (2.0e-04)
 Silicon   3.5548100e-02  4.9531996e-02    39.338      12656  5057 (39.96 %)   1.0715e-04 (1.0e-05)
 Total     4.6402113e-02  6.4934852e-02    39.939      16550  6383 (38.57 %)   2.9218e-04 (2.1e-05)
\----------------------------------------------------------------------------------------------------
```

Interface Non-Delaunay Elements

An *interface element* is an element that has a face (or edge in two dimensions) lying on the interface. A non-Delaunay element is an *interface non-Delaunay element* only if its obtuse face lies on the surface of the interface (see Figure 7).

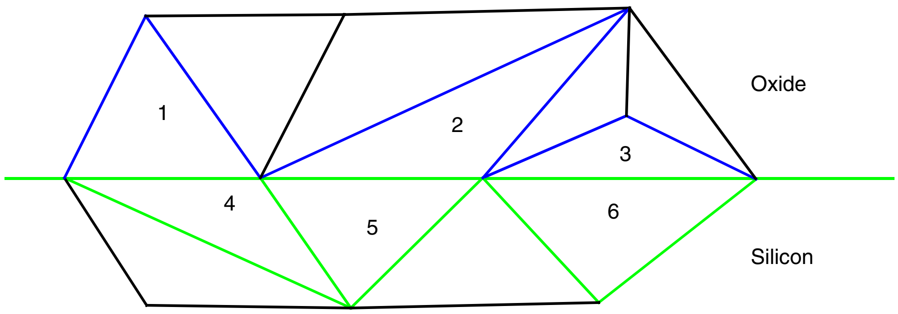


Figure 7 Blue (1, 2, 3) and green (4, 5, 6) elements are oxide and silicon interface elements, respectively. The elements 2, 3, and 4 are non-Delaunay elements, but only element 3 is an interface non-Delaunay element (that is, only this element has an obtuse edge that lies on the surface of the interface).

The following example is a log file for interface non-Delaunay elements:

```
/-------- Interface non-Delaunay elements -----------------------------------
 Region1       Elements  non-Delaunay     Volume            non-Delaunay
 Region2                   Elements        [um2]          DeltaVolume [um2]
 ----------------------------------------------------------------------------
.............................................................................
 silicon         3       0 ( 0.00 %)   1.5775139e-03   0.0000000e+00 ( 0.00 %)
 oxide           3       1 ( 33.0 %)   1.6776069e-03   0.1100000e-03 ( 0.10 %)
.............................................................................
Total            6       1 ( 16.0 %)   3.6951838e-02   0.1100000e+00 ( 0.05 %)
\----------------------------------------------------------------------------
```

Descriptions of Datasets

This section presents examples of datasets that are vertex based and defined on all regions. In the examples, the mesh information for a given vertex `v` is considered.

EdgesPerVertex and ElementsPerVertex

For the example in Figure 8, the dataset value is:

```
EdgesPerVertex(v) = ElementsPerVertex(v) = 5
```

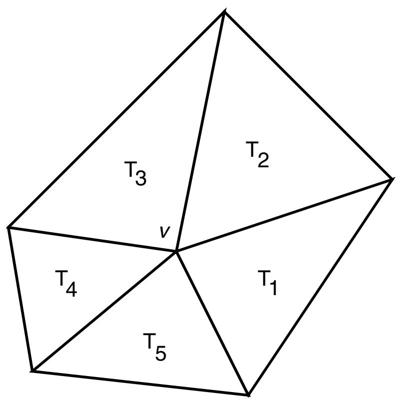


Figure 8 Triangular elements around vertex v

If the vertex v lies on the boundary of a device, then (in the 2D case only, see Figure 9 and Figure 10 on page 31):

```
EdgesPerVertex(v) = ElementsPerVertex(v) + 1
```

The interface edges are additional, and they are defined in `Regions` with the parameter:

```
"material = Interface"
```

If there are interface edges, these edges are added to the list of `EdgesPerVertex`. For example, if there are interface edges between elements T_1, T_2 and T_3, T_4, then:

```
EdgesPerVertex(v) = 7, ElementsPerVertex(v) = 5
```

The examples in Figure 9 and Figure 10 show distribution edges and elements per vertex, respectively. Only the boundary values are different.

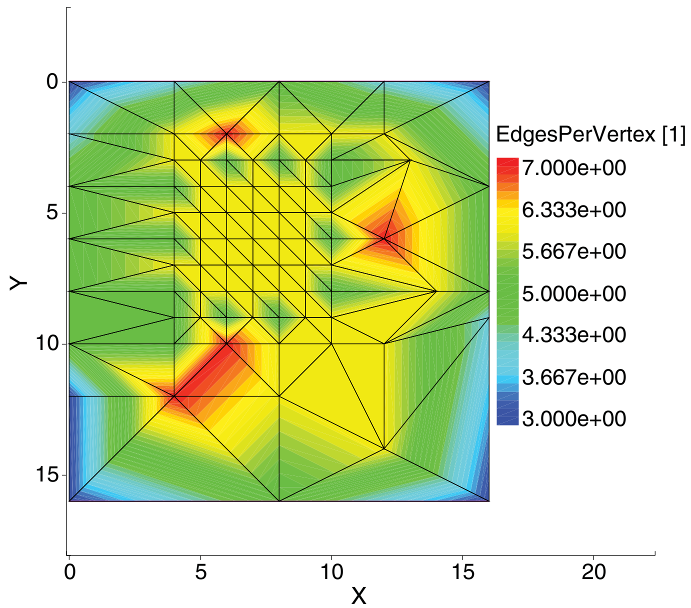


Figure 9 Example of EdgesPerVertex

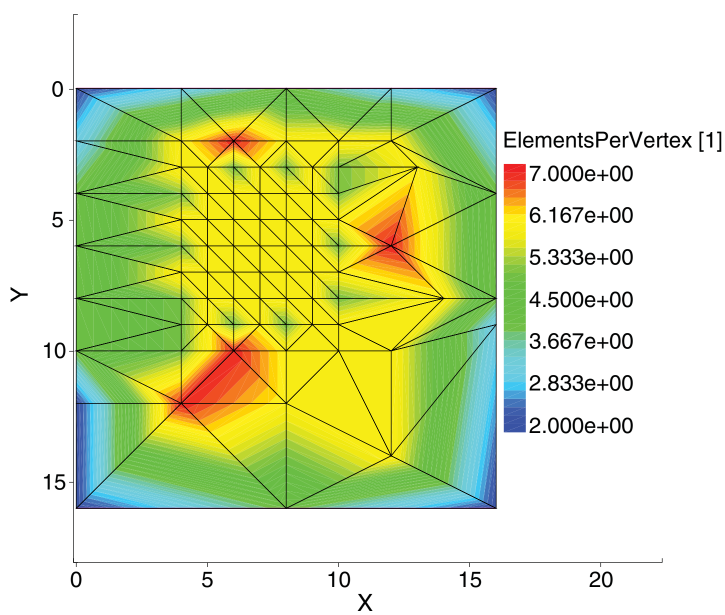


Figure 10 Example of ElementsPerVertex

ElementVolume

This dataset has `location=element`.

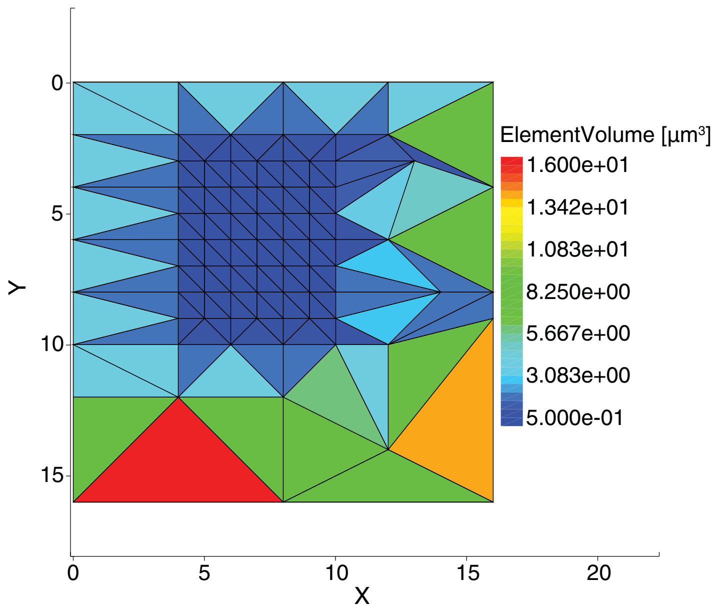


Figure 11 Example of ElementVolume

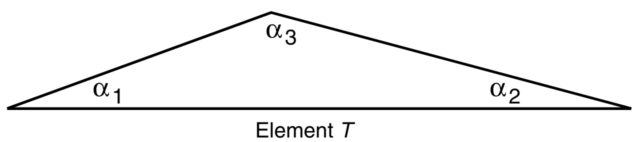
AngleElements

Let α_i be an angle between faces for edge i (in the 2D case, between edges for vertex i). The angle of element T has the following definition:

$$\text{Angle}(T) = \frac{180}{\pi} \cdot \text{asin}(\max(\sin(\alpha_i))) \tag{1}$$

For example, the angle of triangle T has the value:

$$\text{Angle}(T) = \frac{180}{\pi} \cdot \text{asin}(\max(\sin(\alpha_1), \sin(\alpha_2), \sin(\alpha_3))) \tag{2}$$



Notes

For the dataset `AngleElements`:

- If the element has a right angle, then $\text{Angle}(T) = 90°$.
- In the 2D case, if $\text{Angle}(T) < 60°$, then this triangle is obtuse:

$$\alpha_1, \alpha_2 < \text{Angle}(T) \qquad \alpha_3 > 180 - \text{Angle}(T) \tag{3}$$

- If $\text{Angle}(T) < 10^{-8}$, then this element is flat (see Basic Definitions on page 26).

The dataset `AngleElements` has `location=element`.

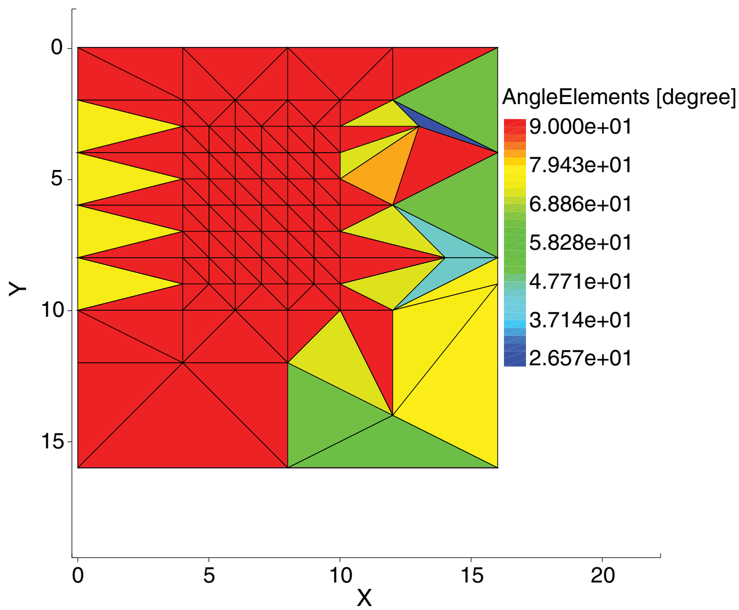


Figure 12 Example of AngleElements

AngleVertex

The dataset `AngleVertex` has the following definition (see Figure 13):

```
AngleVertex(v) = max(Alpha(Tk,v)), k=1,...,nbElements(v)
```

where `nbElements(v)` is equal to the number of elements per vertex `v`, and `Alpha(Tk,v)` corresponds to the element–vertex angle.

In Figure 13, the `AngleVertex` dataset shows a ‘poor’ vertex of the mesh as a red vertex.

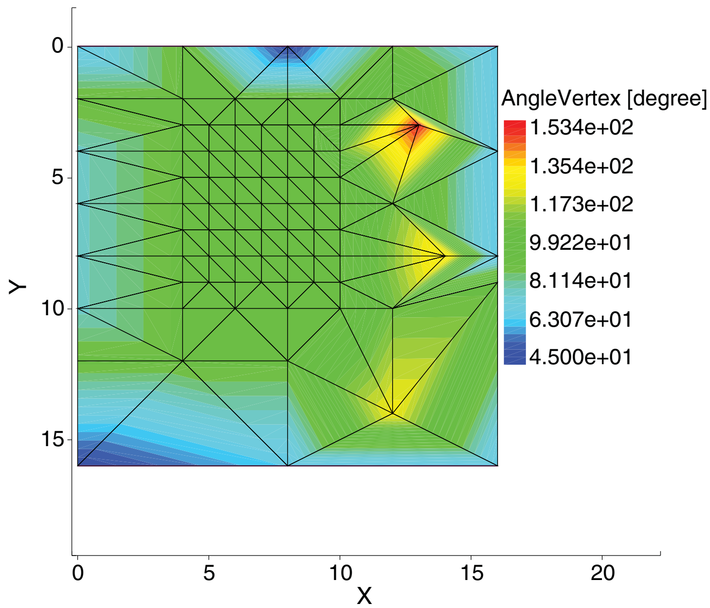


Figure 13 Example of AngleVertex

ShortestEdge

The dataset `ShortestEdge` has the following definition:

```
ShortestEdge(v) = min(Length(Edge_k)), k=1,...,nbEdges(v)
```

where `nbEdges(v)` is equal to the number of edges per vertex `v`. The unit of `ShortestEdge` is µm .

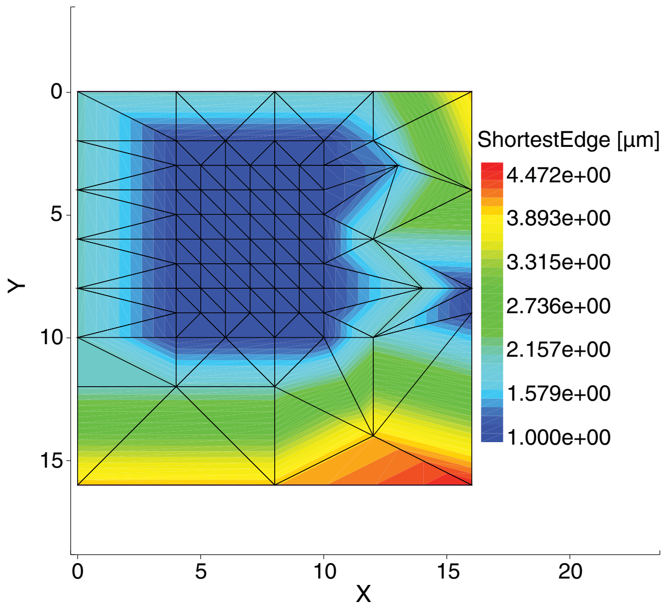


Figure 14 Example of ShortestEdge

IntersectionNonDelaunayElements

This dataset has `location=element`. It is equal to $\delta_1(T)$ (see Basic Definitions on page 26).

Figure 15 shows a mesh that contains seven non-Delaunay elements.

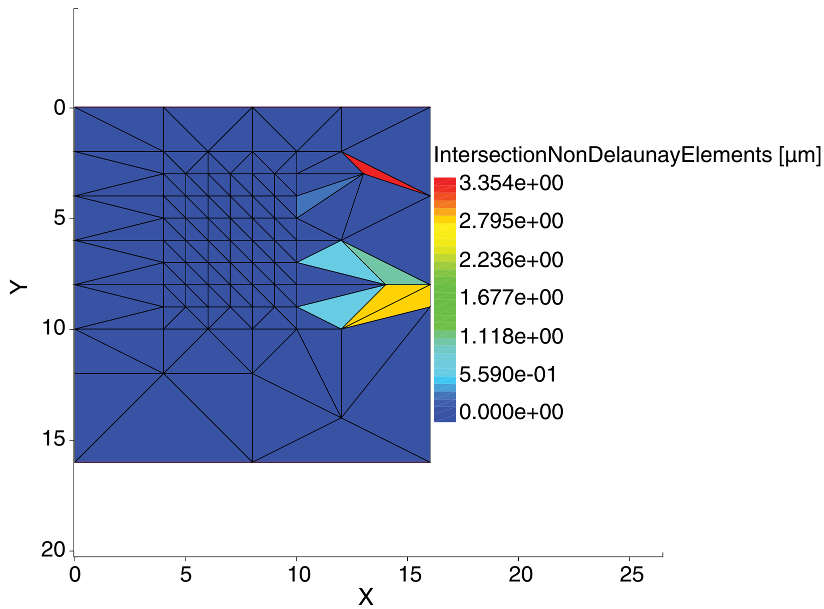


Figure 15 Example of IntersectionNonDelaunayElements

VolumeIntersectionNonDelaunayElements and CoeffIntersectionNonDelaunayElements (Two Dimensions)

The difference between these datasets and the `IntersectionNonDelaunayElements` dataset is the value of the non-Delaunay measures $\delta_2(T)$ and $\delta_3(T)$, instead of $\delta_1(T)$ (see Basic Definitions on page 26).

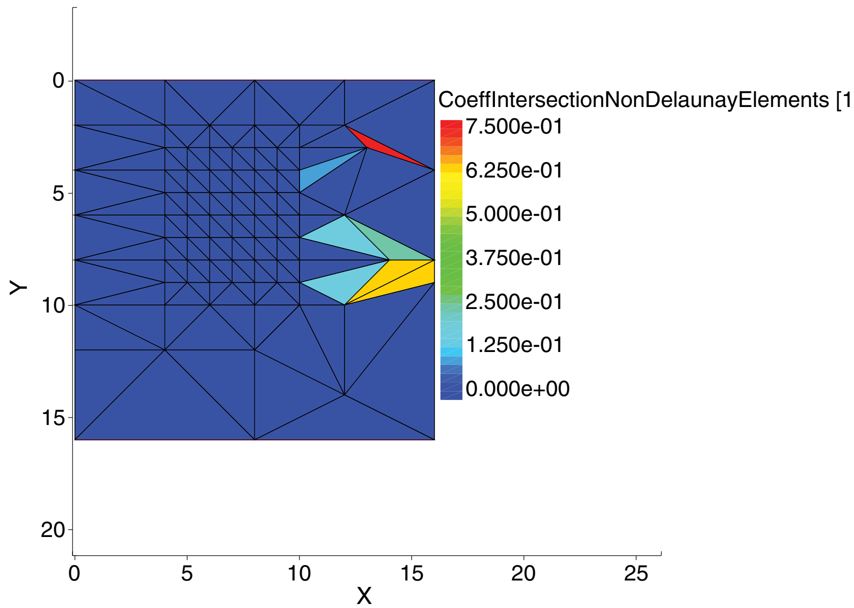


Figure 16   Example of CoeffIntersectionNonDelaunayElements

ElementsWithCommonObtuseFace

This dataset is similar to the `VolumeIntersectionNonDelaunayElements` dataset. The value of this dataset is positive only for a pair of neighbor elements with a common obtuse face.

In the 2D case, `Face` is `Edge`. Figure 17 shows two elements with a common obtuse edge.

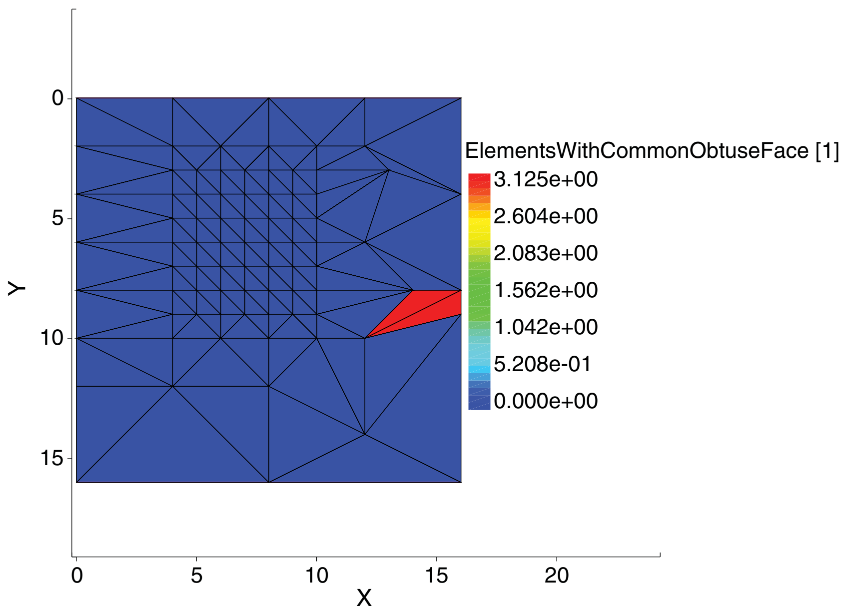


Figure 17 Example of ElementsWithCommonObtuseFace

ElementsWithObtuseFaceOnBoundaryDevice

This dataset has nonzero values only for elements that have an obtuse face on the boundary of the device. For these elements, the non-Delaunay measure $\delta(T)$ is defined as (see Figure 18 on page 39):

$$\delta(T) = \frac{\text{Area}(0, V, 1, 0)}{\text{Area}(T)} = \frac{d}{h}, \text{ in the 3D case, Area = Volume} \quad (4)$$

The value of the dataset is equal to:

`ElementsWithObtuseFaceOnBoundaryDevice(v) = max(`$\delta(T_k)$`), k=1,...,nbElements(v)`

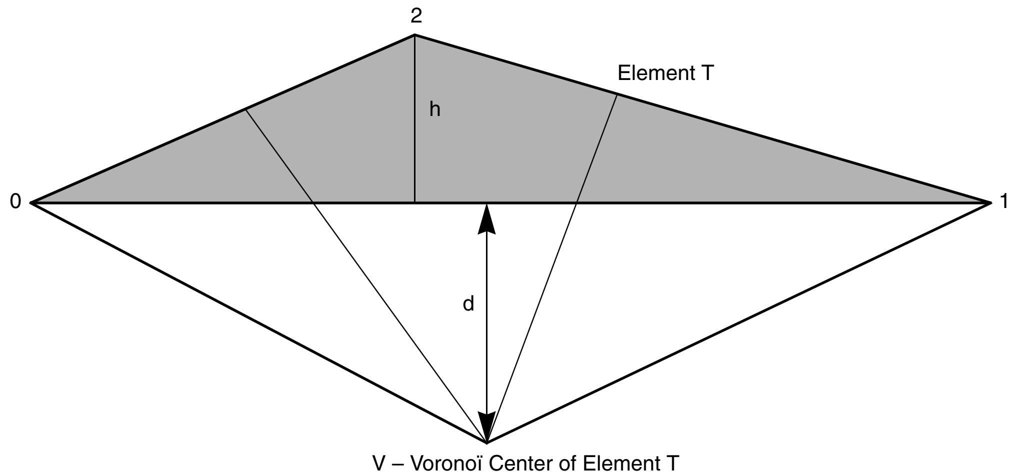


Figure 18 Non-Delaunay measure for element with obtuse face on boundary device

Figure 19 shows one element with an obtuse edge on the boundary device.

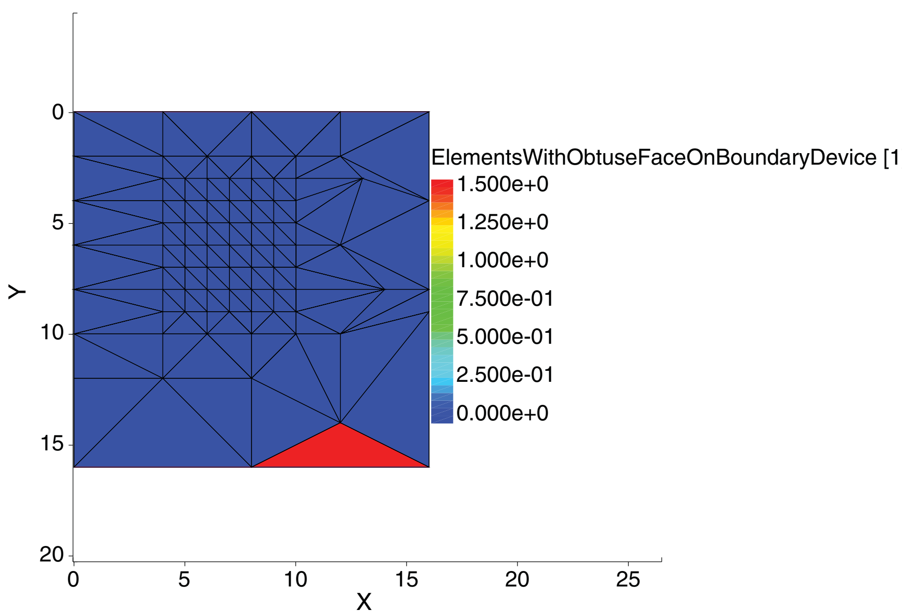


Figure 19 Example of ElementsWithObtuseFaceOnBoundaryDevice

TetQualityEdge and TetQualityHeight

This dataset has `location=element` (see Basic Definitions on page 26).

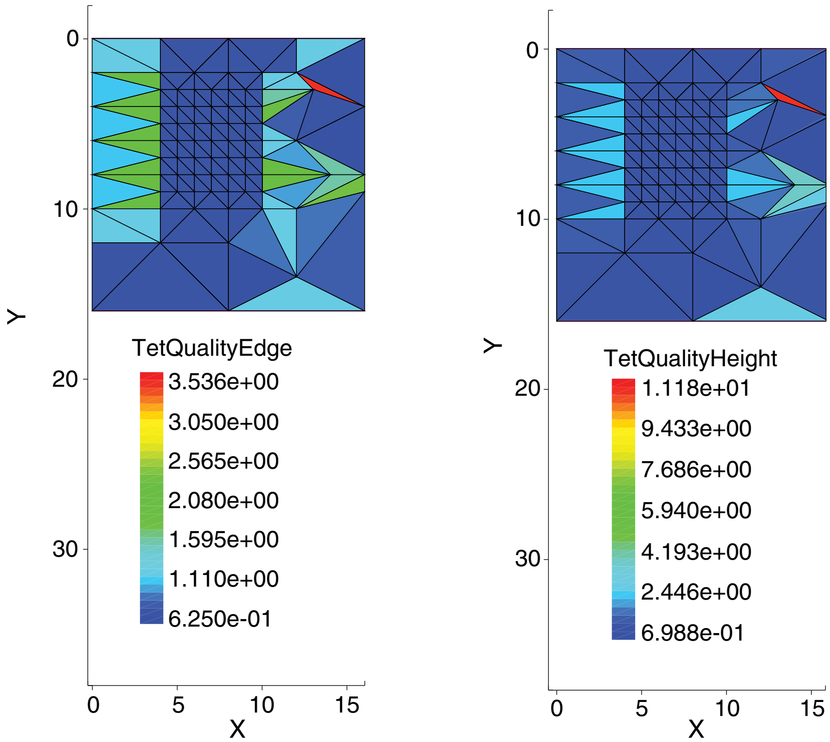


Figure 20 Examples of (*left*) TetQualityEdge and (*right*) TetQualityHeight