Utilities User Guide

Version T-2022.03, March 2022

SYNOPSYS®

Copyright and Proprietary Information Notice

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

About This Guide

This user guide describes various utilities used by the Synopsys® TCAD Sentaurus™ tools.

For additional information, see:

- The TCAD Sentaurus release notes, available on the Synopsys SolvNetPlus support site (see Accessing SolvNetPlus)
- Documentation available on the SolvNetPlus support site

Conventions

The following conventions are used in Synopsys documentation.

Convention	Description
`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 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

The datexcodes.txt File

This chapter describes the datexcodes.txt file.

File Structure

The `datexcodes.txt` file is the Synopsys configuration database for materials, doping species, and other quantities that are used in semiconductor process and device simulations. Various TCAD tools refer to the database for different purposes. The file does not contain the physical properties of materials or quantities, but rather 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.

Note:

The `datexcodes.txt` file provides different color schemes. The Classic color scheme has been traditionally used in TCAD tools. The Vivid color scheme is the default. You can easily switch between the two color schemes. The following sections refer to the Classic color scheme.

Header

The first four lines in the `datexcodes.txt` file constitute the header. The first two lines specify the version number and the file type. The last two lines are strings containing comments. For example:

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

Materials

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

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

Table 1 Fields used for material specification

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

You can declare multiple names (or aliases) for a material. For example:

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

Alternative colors can be listed after the primary color. For example:

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

If you specify only one value for `color`, then it is used for the Classic color scheme. If you specify a second value for `color`, then it is used for the Vivid color scheme. If you do not specify any color, then the default color (gray) is used.

Variables

A variable such as `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 Fields used for variable specification

Field	Description
`alter1`	Name used for translation to and from SUPREM-4a.
`alter2`	Name used for translation to and from SUPREM-4b.
`arsinh`	Scaling factor for `arsinh` interpolation mode. Default: 10^{14}
`doping`	Specification of doping species (see Doping Specification on page 10).
`factor`	Scaling factor. Default: `1`
`interpol`	Interpolation mode. Options are `asinh`, `linear`, or `log`. Default: `linear`
`label`	Name used for display purposes. Default: `undefined`
`material`	Specifies the validity (domain of definition) of this quantity. Options are `All`, `Conductor`, `Insulator`, or `Semiconductor`. Default: `All`
`parity`	Symmetry property of tensors, either `+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 `material` field can specify multiple materials or material groups. For example:

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

Doping Specification

Doping species are identified by the `doping` field. For example:

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

For details, see *Sentaurus™ Device User Guide*, Specifying Doping Species.

Search Strategy

You can use multiple `datexcodes.txt` files, in which case, 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.

2

TCAD Log File Browser

This chapter discusses the TCAD Log File Browser.

The TCAD Log File Browser allows you to access information in a TCAD log file efficiently. 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 have an overview.

Launch the TCAD Log File Browser

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

These 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
```

You can launch the TCAD Log File Browser by entering, for example:

```
> logbrowser n1_fps_log.xml
```

The XML tag information is preprocessed and optimized for efficient display in a browser. The TCAD Log File Browser calls the 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 might take a few seconds.

During preprocessing, the TCAD Log File Browser provides information about the progress of preprocessing and the action taken. This information is written to the *standard output* pipe.

If you start the TCAD Log File Browser from the command line of a terminal window, the output is shown in that window. If you start it from Sentaurus Workbench, the output is shown in the terminal window from which you started Sentaurus Workbench (see Integration in Sentaurus Workbench on page 16).

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, enter on the command line:

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

Note:

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

After preprocessing is completed, your default browser opens automatically. The preprocessed marked-up version of the log file is saved with the `*.html` extension. To reload a log file that has been preprocessed already, you can call the browser directly by entering, for example:

```
> firefox n1_fps_log.html
```

User Interface

The user interface of the TCAD Log File Browser consists of the following areas:

- Table of Contents
- Active Tags Panel
- Info Level Selector Panel
- Main Panel

Table of Contents

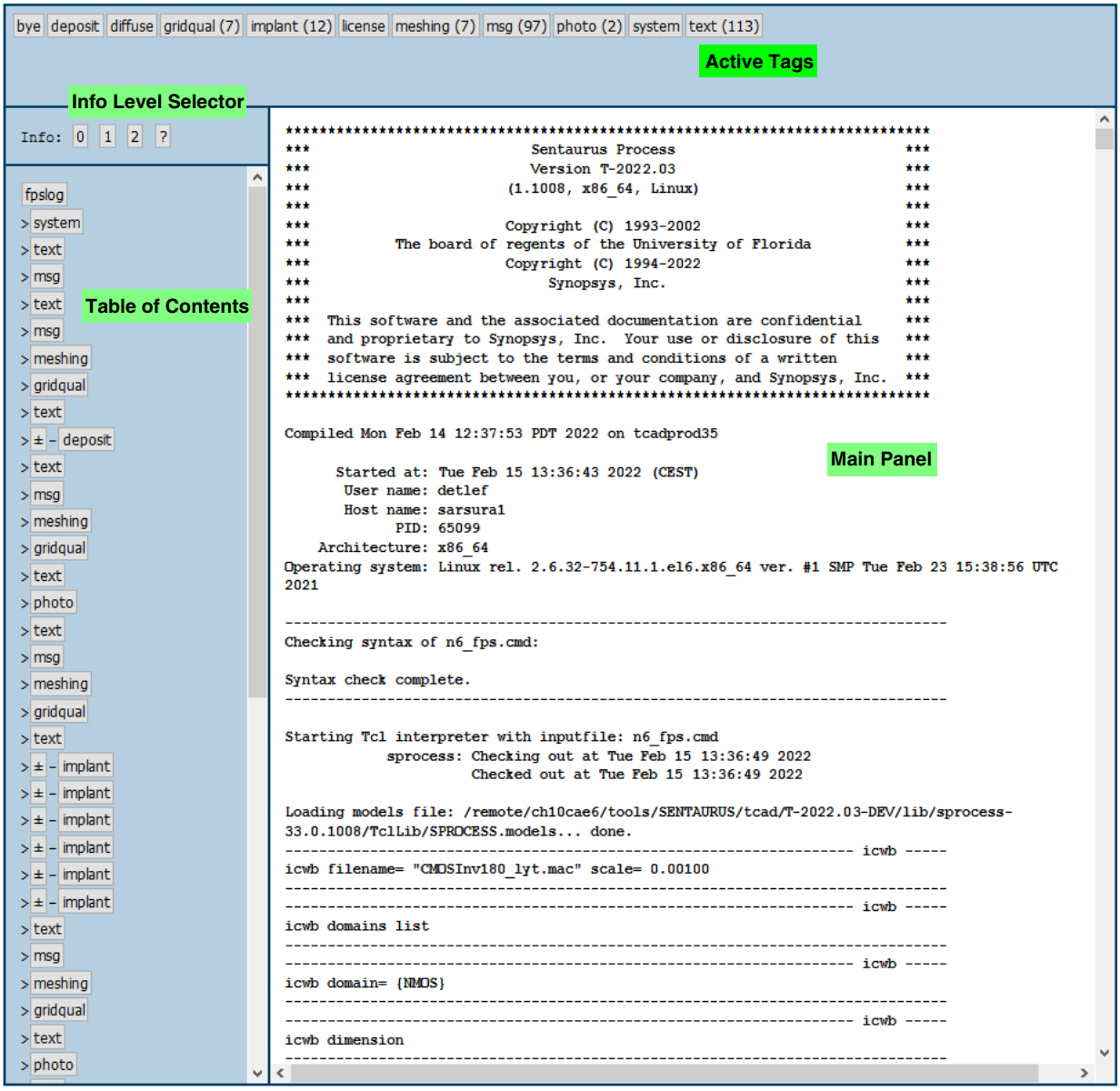
The Table of Contents shows the structure of the log file as a tree of section tags, where each section tag is a button. When you click a section tag button, the log file content corresponding to that section is displayed in the main panel.

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 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. For a Sentaurus Process log file, this 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 word in the main panel, all section tag buttons leading to sections that contain the clicked word are shown in bold. This allows you to quickly locate the relevant section tags in the Table of Contents.

Figure 1 User interface of the TCAD Log File Browser



Active Tags Panel

The Active Tags panel shows an ordered list of all section tags contained in the section tag 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 instances is shown to the right of the section tag name in parentheses (see Figure 1 on page 14).

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 the same time. Selected buttons become gray. You deselect them by clicking the section tag buttons again.

You can use this feature to view 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 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 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, 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 displays the selected sections of the log file. Initially, this panel shows the entire content of the log file, using different foreground and background colors. The background color depends on the information level. For example, all content belonging to info level 0 is black and content belonging to info level 1 is 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. You can customize the color scheme (see Custom Color Schemes on page 17).

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 in the main panel. For example, if you find log messages from a meshing operation, you might want to know which `etch` command triggered this remeshing step.

Integration in Sentaurus Workbench

To use the TCAD Log File 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, enter `--xml` in the **Command Line** text box.

You can launch the TCAD Log File 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 removes 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 runtime 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

The TCAD Log File Browser can visualize log files generated with only Version K-2015.06 (or later versions) of Sentaurus Process, Sentaurus Interconnect, or Sentaurus Device.

The TCAD Log File Browser was designed for Firefox version 3.6.18, the default browser that ships with Red Hat Enterprise Linux v5. It 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 Log File Browser requires that ActiveX is enabled.

3

Sentaurus spice2sdevice Utility

This chapter discusses the Sentaurus spice2sdevice utility that converts a subset of Synopsys PrimeSim™ HSPICE® netlist files into equivalent circuit files of Sentaurus Device.

PrimeSim HSPICE netlist files (extension `.cir`) are documented in the *PrimeSim™ 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 PrimeSim HSPICE netlist files directly in the `System` section. See the *Sentaurus™ Device User Guide*.

PrimeSim HSPICE Netlist Files

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

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

The title line is followed by a sequence of PrimeSim 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 (for PrimeSim HSPICE model files).

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

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

Comments

A line starting with either a dollar sign ($) or an asterisk (*) is a comment. For example:

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

You can use in-line comments after the dollar sign. For example:

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

Continuation Lines

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

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

.INCLUDE Statements

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

```
.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}

Table 3 Scale factors (Continued)

Scale factor	Description	Multiplying factor
`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}
`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 PrimeSim HSPICE tool, parameters are names that you associate with a value. Numeric and string parameters are supported. For example:

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

Table 4 Supported built-in mathematical functions

Function	Description
sin	Returns the sine of x (radians).
cos	Returns the cosine of x (radians).
tan	Returns the tangent of x (radians).

Table 4 Supported built-in mathematical functions (Continued)

Function	Description
asin	Returns the inverse sine of x (radians).
acos	Returns the inverse cosine of x (radians).
atan	Returns the inverse tangent of x (radians).
sinh	Returns the hyperbolic sine of x (radians).
cosh	Returns the hyperbolic cosine of x (radians).
tanh	Returns the hyperbolic tangent of x (radians).
abs	Returns the absolute value of x: |x|.
sqrt	Returns the square root of the absolute value of x: sqrt(-x)=-sqrt(|x|).
pow	Absolute power. Returns the value of x raised to the integer part of y: $x^{(\text{integer part of y})}$.
pwr	Signed power. Returns the absolute value of x, raised to the y power, with the sign of x: (sign of x)$\|x\|^y$.
log	Natural logarithm. Returns the natural logarithm of the absolute value of x, with the sign of x: (sign of x)log(|x|).
log10	Base 10 logarithm. Returns the base 10 logarithm of the absolute value of x, with the sign of x: (sign of x)$\log_{10}(\|x\|)$.
exp	Exponential. Returns e, raised to the power x: e^x.
db	Decibels. Returns the base 10 logarithm of the absolute value of x, multiplied by 20, with the sign of x: (sign of x)$20\log_{10}(\|x\|)$.
int	Returns the integer portion of x (which ignores the fractional portion of the number).
nint	Rounds x up or down, to the nearest integer.
sgn sign	Return sign, as follows: • Returns -1 if x is less than 0. • Returns 0 if x is equal to 0. • Returns 1 if x is greater than 0.
floor	Rounds down to the nearest integer (ignores the fractional part of the number).

Table 4 Supported built-in mathematical functions (Continued)

Function	Description
ceil	Rounds up to the nearest integer (ignores the fractional part of the number).
min	Smaller of two arguments. Returns the numeric minimum of x and y.
max	Larger of two arguments. Returns the numeric maximum of x and y.

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 also supported:

```
.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

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

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

Table 5 Supported 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 6 lists the values for the parameter `level` in recognized MOSFET models. In the case of Levels 1, 2, and 3, the corresponding device can be either a PrimeSim 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 a PrimeSim HSPICE MOSFET, but you can use the command-line option `-b` to switch to a Berkeley SPICE MOSFET.

Table 6 Supported 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

Table 6 Supported SPICE MOSFET models (Continued)

Level	Device	Description
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 7 Supported PrimeSim 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 8 Supported Berkeley SPICE element types [1]

First letter	Element	Example
s	Voltage-controlled switch	`S1 1 2 3 4 SWITCH1 ON`
w	Current-controlled switch	`W1 1 2 VCLOCK SWITCHMOD1`

Table 8 Supported Berkeley SPICE element types [1] (Continued)

First letter	Element	Example
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. For example:

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

Use the .OPTION PARHIER statement to specify scoping rules. For example:

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

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

Command-Line Options

Table 9 lists the command-line options of the Sentaurus spice2sdevice utility.

Table 9 Command-line options

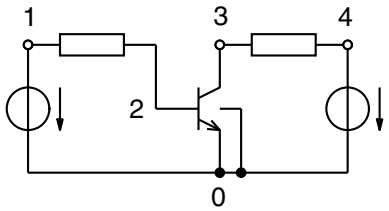
Option	Description
-b	Use Berkeley SPICE models instead of PrimeSim HSPICE models (applies only to MOSFETs Level 1, 2, and 3).
-c	Translates a SPICE circuit file (default).
-d	Prints additional debug information.
-h	Displays a help message.
-m	Translates a SPICE model file.
-o <*filename*>	Stores the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

This example considers a simple resistor transistor logic (RTL) inverter as shown in Figure 2.

Figure 2 Simple RTL inverter



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 following 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)
```

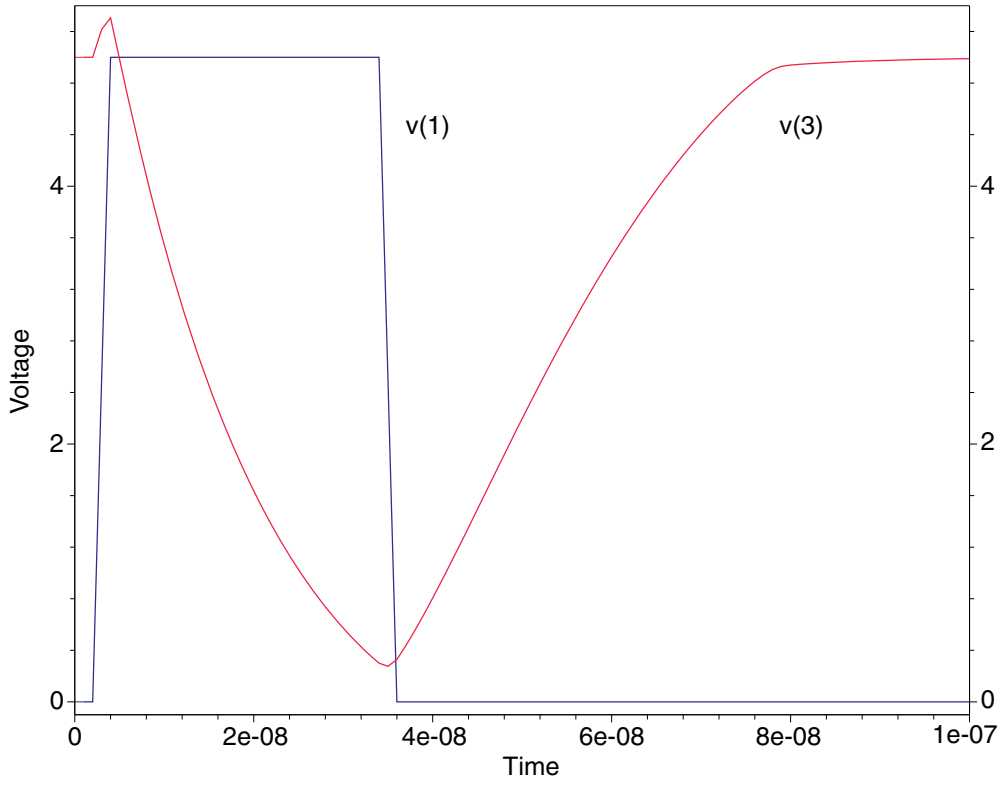
```
    Quasistationary (Goal {Parameter=vcc."dc"  Value=5})
       { Coupled { Circuit } }

    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.

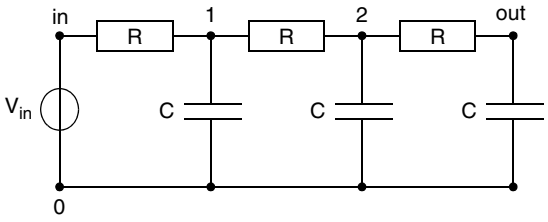
Figure 3 Transient simulation of a simple RTL inverter



Subcircuit Example

The Sentaurus `spice2sdevice` utility supports basic SPICE subcircuits. This example considers a chain of low-pass filters as 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`). For example:

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

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

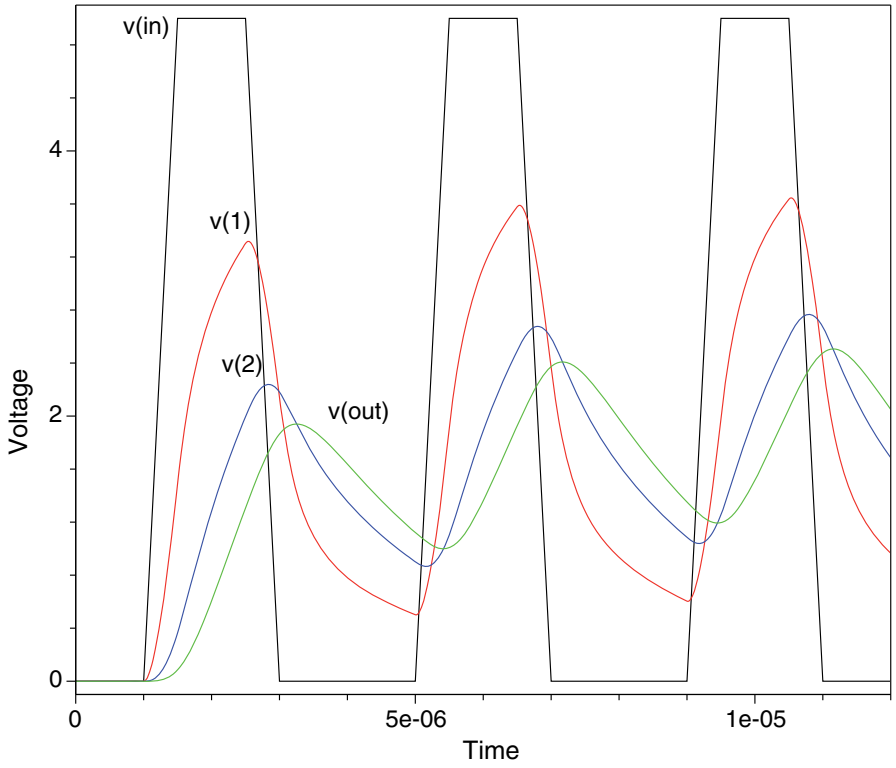
```
File {
   Output = "filter"
   SPICEPath = "."
}

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.

4

Box Method Utility

This chapter describes the box method utility.

Using the Box Method Utility

The box method utility analyzes the quality of a mesh. It 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).

There are two versions of this utility:

- *Double* precision (boxmethod)
- *Long double* precision (boxmethodl)

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.

Syntax

The syntax of the box method utility is one of the following:

```
boxmethod [options] GridFile
```

```
boxmethodl [options] GridFile
```

Here, *GridFile* is a TDR file. The available options are:

-a *Algorithm* *Algorithm* can be one of the following:

- AverageBoxMethod
- CVPL_AverageBoxMethod (default)
- MaterialBoundaryTruncatedVoronoiBox
- RegionBoundaryTruncatedVoronoiBox
- TruncatedVoronoiBox

`-h`	Show this help message and exit
`-NoGas`	Without computation in Gas
`-numThreads` *n*	Parallel computation where *n* is the number of threads (default: `1`)
`-StackSize` *n*	Parallel computation where *n* is the size of the stack (default: `0`)
`-v`	Print header with version number

For detailed descriptions, see *Sentaurus™ Device User Guide*, Chapter 38.

Definitions

The following sections provide definitions of basic elements.

Obtuse Element

An element is *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 *obtuse* if the center of the circumsphere of the element and the element itself lie in different half-spaces `Sf1` and `Sf2`.

Note:

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

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, two, or three obtuse faces.

Non-Delaunay Element

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

Flat Element

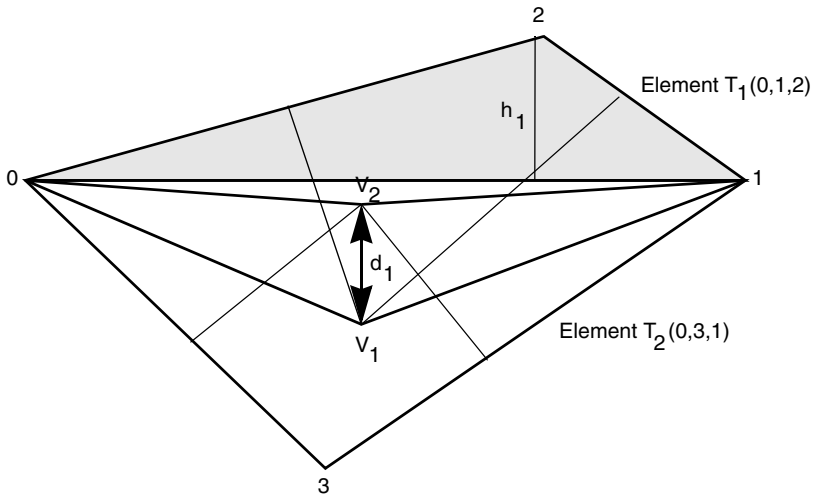
Let α_i be an angle between faces for edge i (in the 2D case, between edges for vertex i). An element is *flat* if for all angles (a tetrahedron has six angles) $\alpha_i < 10^{-8}$ (the angle is close to zero) or $\alpha_i > \pi - 10^{-8}$ (the angle is close to π).

Non-Delaunay Measure

Figure 6 shows the following elements of a non-Delaunay measure:

- $V_{1,2}$ 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.

Figure 6 Schematic of a non-Delaunay measure



There are three definitions of a non-Delaunay measure:

1. $\delta_1 = d_1 = \text{Length}(V_1, V_2)$, [µm]. 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` = $\frac{R}{H_{min}}$

Here, 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 Files

This section describes the content of log files.

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`).

For example:

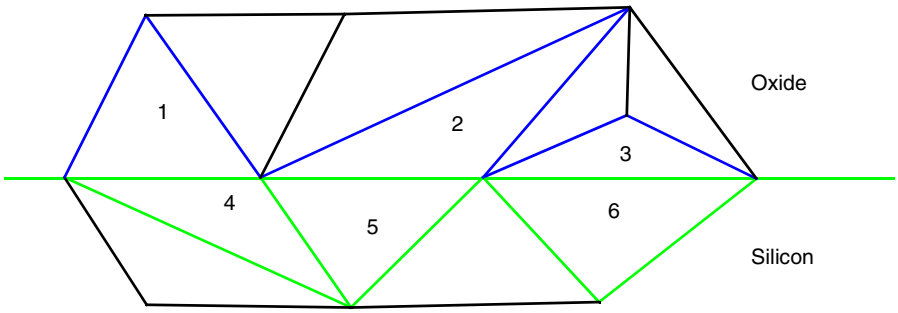
```
/-------- Region non-Delaunay elements ----------------------------------------
 Region  Volume        BoxMethodVolume DeltaVolume Elements non-Delaunay non-
DelaunayVolume
 name   [um2]         [um2]           [%]                   Elements    [um2] [%]
 ----------------------------------------------------------------------------
 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 an 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).

In Figure 7, 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).

Figure 7 *Blue (1, 2, 3) and green (4, 5, 6) elements are oxide and silicon interface elements, respectively*



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 %)
\------------------------------------------------------------------------
```

Datasets

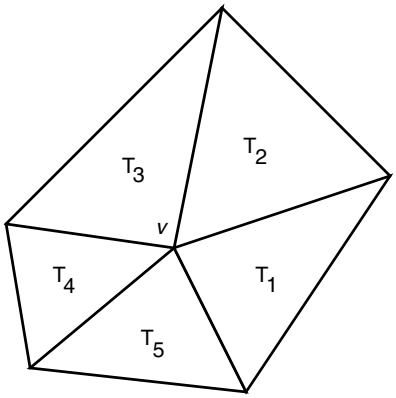
This section describes datasets that are vertex based and defined on all regions. The examples consider the mesh information for a given vertex `v`.

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

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

The interface edges are additional and 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 differ.

Figure 9 Example of EdgesPerVertex

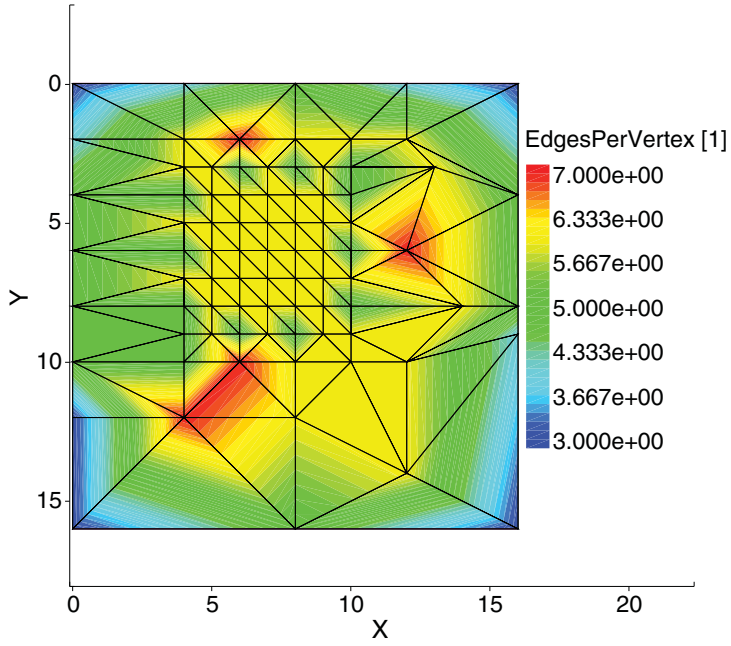
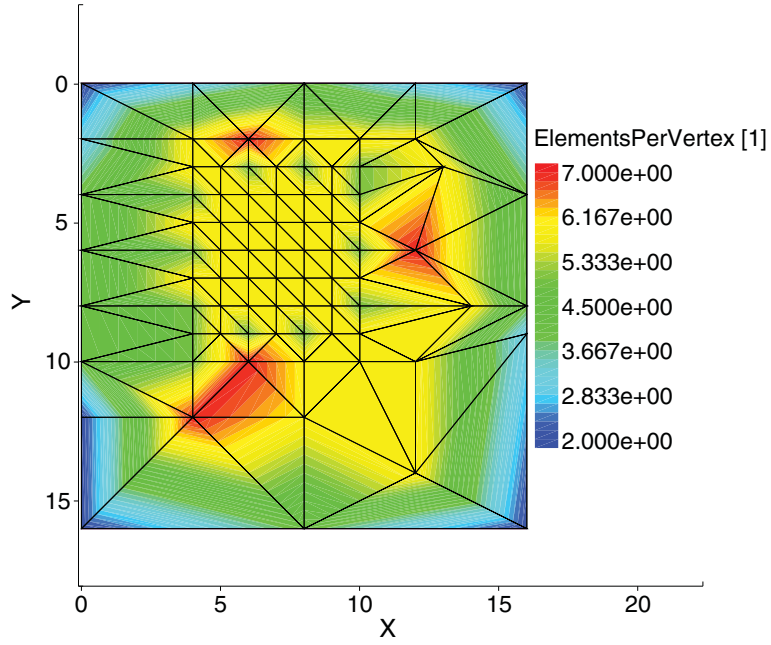


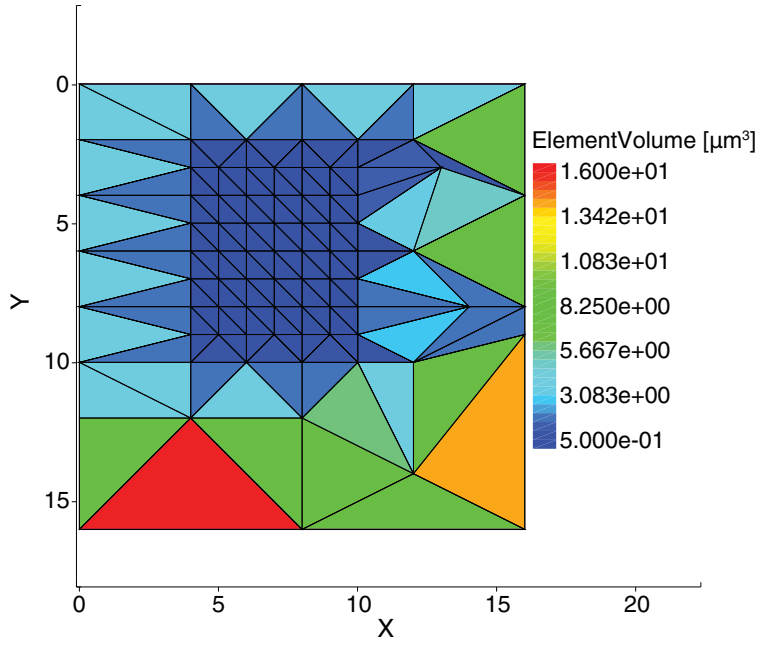
Figure 10 Example of ElementsPerVertex



ElementVolume

This dataset has `location=element`.

Figure 11 Example of ElementVolume



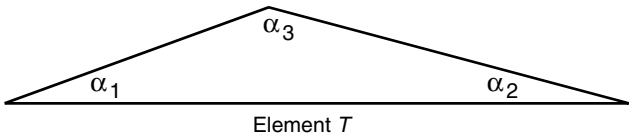
AngleElements

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:

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

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

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



For the dataset `AngleElements`:

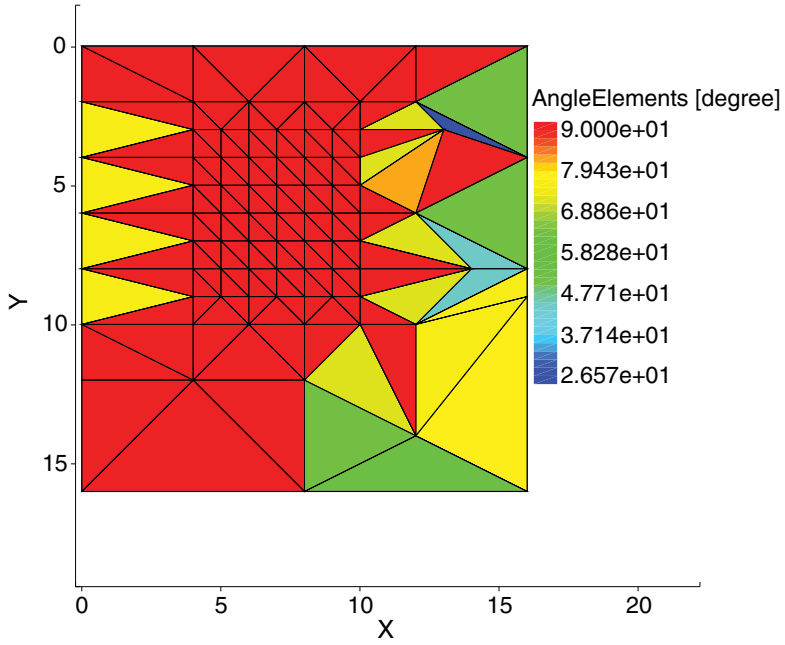
- If the element has a right angle, then Angle(T) = 90°.
- In the 2D case, if Angle(T) < 60°, then this triangle is obtuse:

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

- If Angle(T) < 10^{-8}, then this element is flat (see Definitions on page 33).

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

Figure 12 Example of AngleElements



AngleVertex

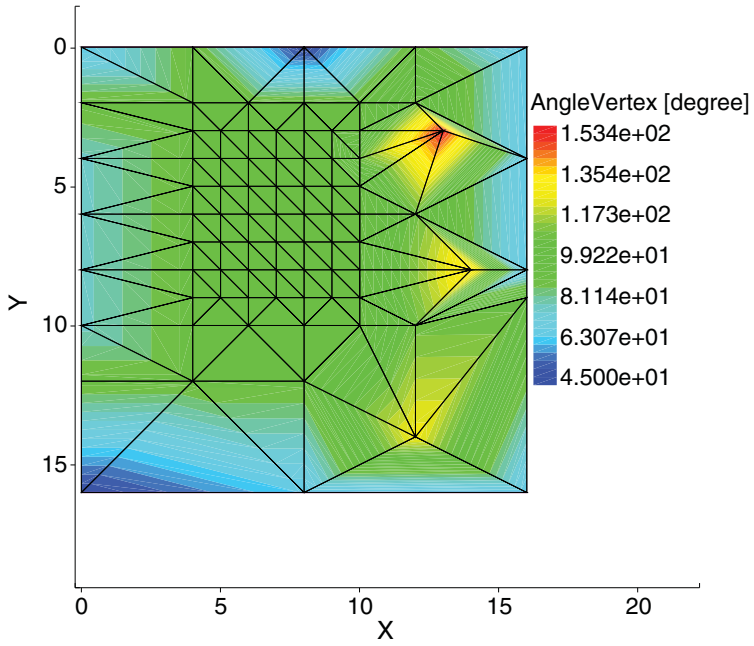
The dataset `AngleVertex` has the following definition (see Figure 13):

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

Here, `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



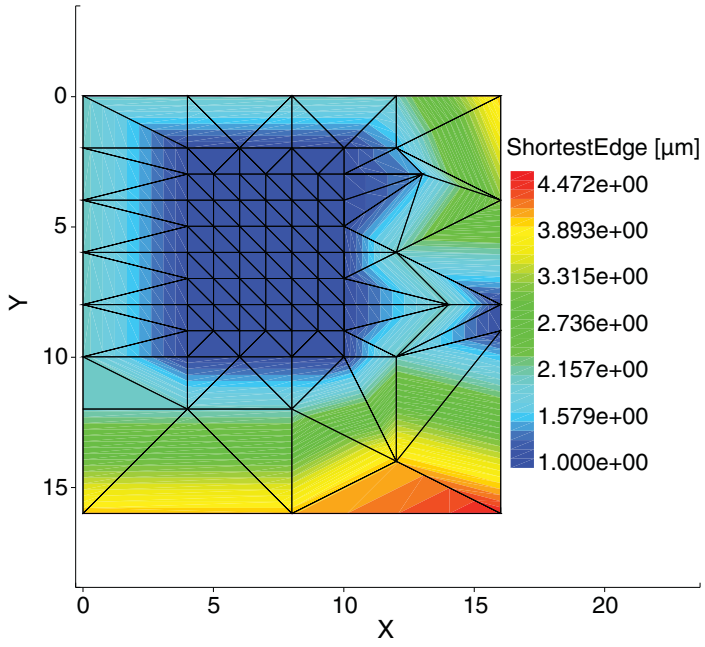
ShortestEdge

The dataset `ShortestEdge` has the following definition:

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

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

Figure 14 Example of ShortestEdge

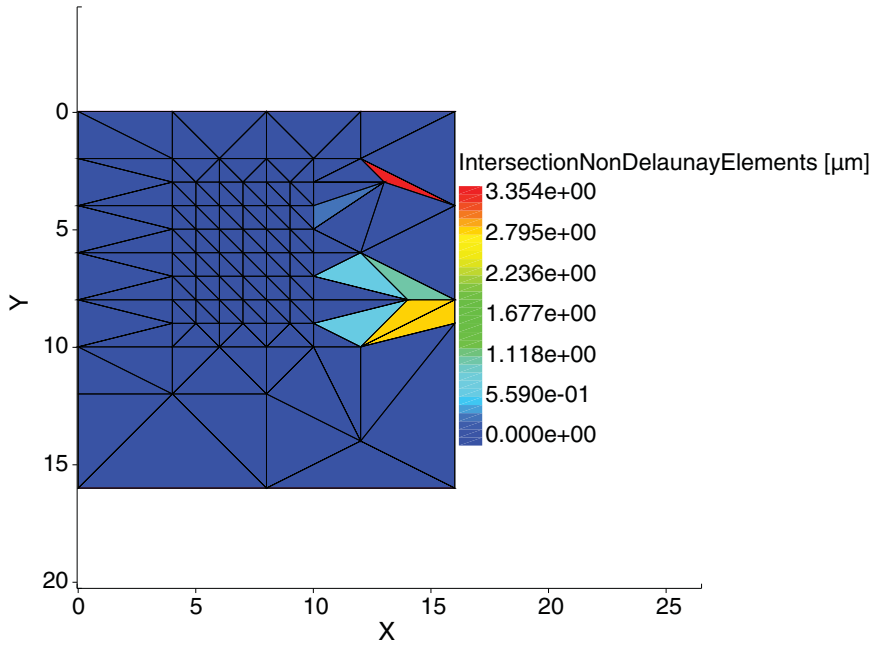


IntersectionNonDelaunayElements

This dataset has `location=element`. It is equal to $\delta_1(T)$ (see Definitions on page 33).

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

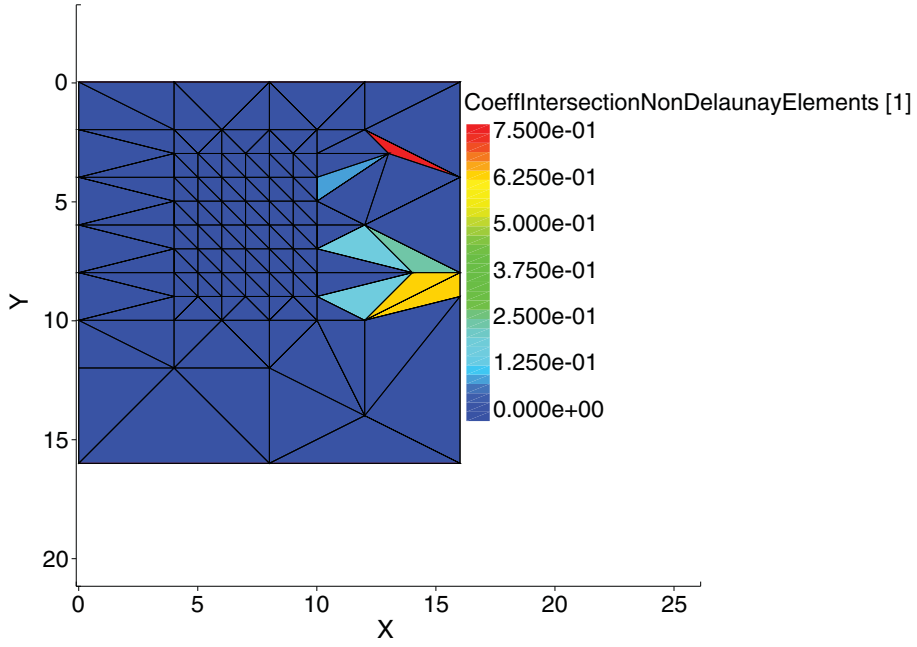
Figure 15 Example of IntersectionNonDelaunayElements



VolumeIntersectionNonDelaunayElements and CoeffIntersectionNonDelaunayElements (Two Dimensions)

The difference between these datasets and the `IntersectionNonDelaunayElements` dataset (see IntersectionNonDelaunayElements on page 43) is the value of the non-Delaunay measures $\delta_2(T)$ and $\delta_3(T)$, instead of $\delta_1(T)$ (see Definitions on page 33).

Figure 16 Example of CoeffIntersectionNonDelaunayElements

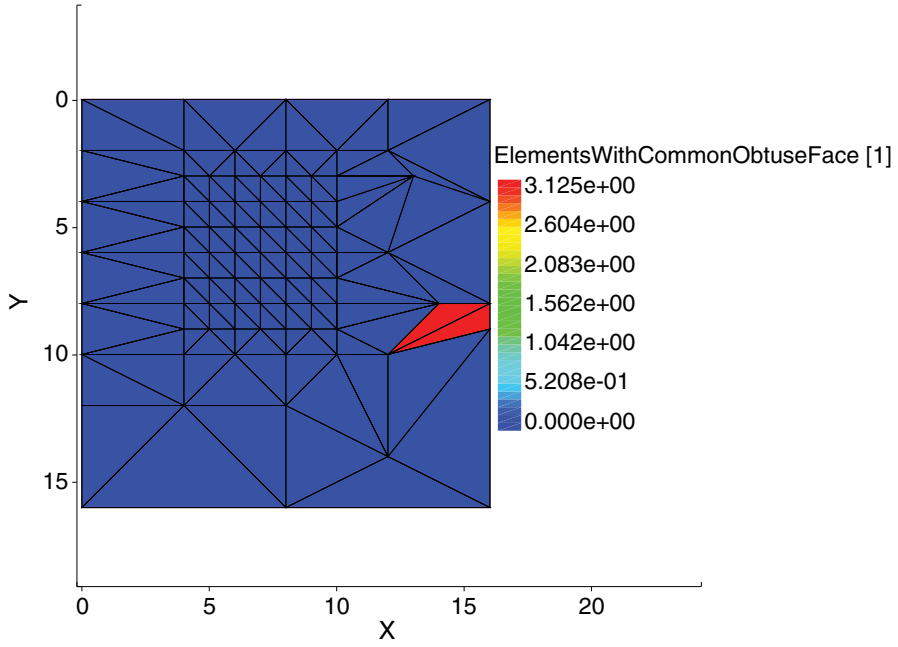


ElementsWithCommonObtuseFace

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 46):

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

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

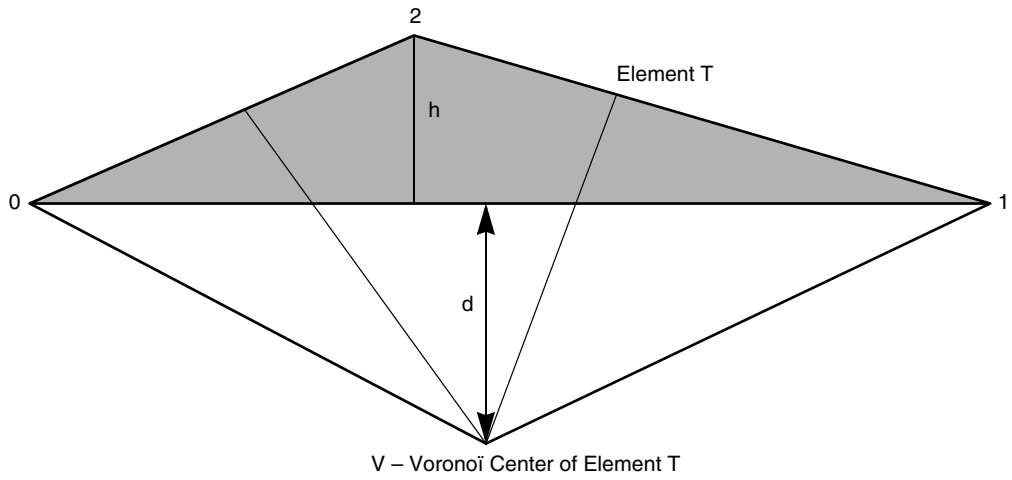
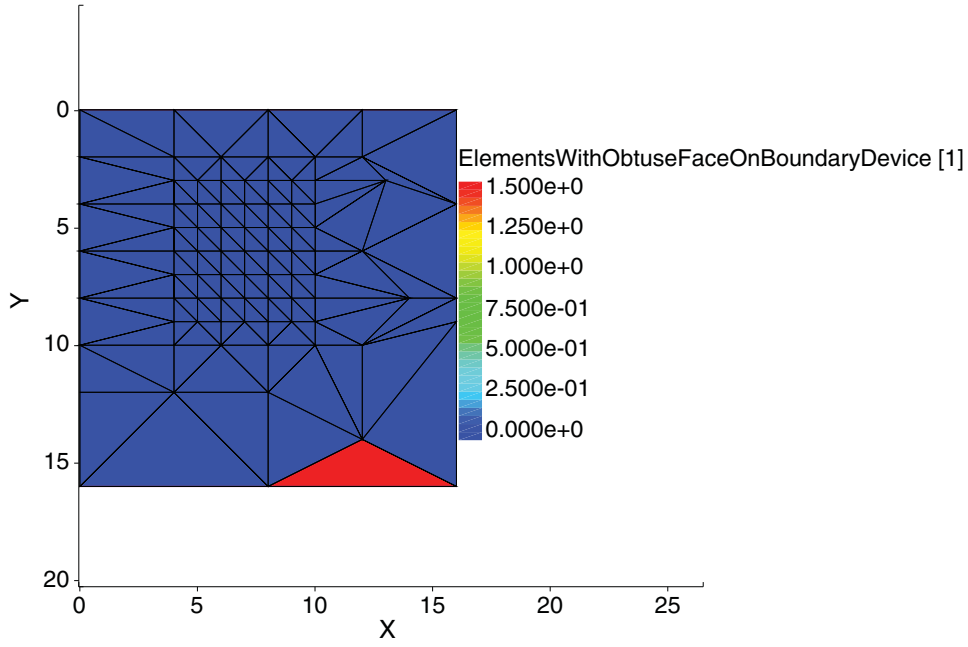


Figure 19 shows one element with an obtuse edge on the boundary device.

Figure 19 Example of ElementsWithObtuseFaceOnBoundaryDevice



TetQualityEdge and TetQualityHeight

These datasets have `location=element` (see Definitions on page 33).

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