SigmaGraph

Data Graphing and Analysis Software

http://www.hamady.org
sidi@hamady.org

Contents

SigmaGraph is a data plotting and analysis software designed to be lightweight, reliable and easy to use. SigmaGraph runs on Windows XP, Vista and Windows 7/8/10. SigmaGraph offers almost all the functionality needed by scientists and engineers: editable **datasheets** (column properties, create series, set column values by using any mathematical expression, show column statistics, import/export from/to ASCII file, mask and unmask cells, printing, etc.); **scientific graphing** with a complete control of the graph (line/symbol style, colors, fonts, legends, axis properties, grid, tick, labels, scale, auto scale, log/linear scale, zoom in/out, copy format to another graph, export as image, printing, graph templates, etc.); **curve fitting** (24 models including linear, polynomial, exponential, Gaussian (up to 5 peaks), Lorentzian (up to 5 peaks), Pearson VII, logistic, power, etc.); **error bars** (percentage, constant or any user defined data); **drawing tools** (line, rectangle, ellipse); **mathematical console**; powerful **scripting engine**; **customizable script editor** with syntax highlighting, code completion, etc.

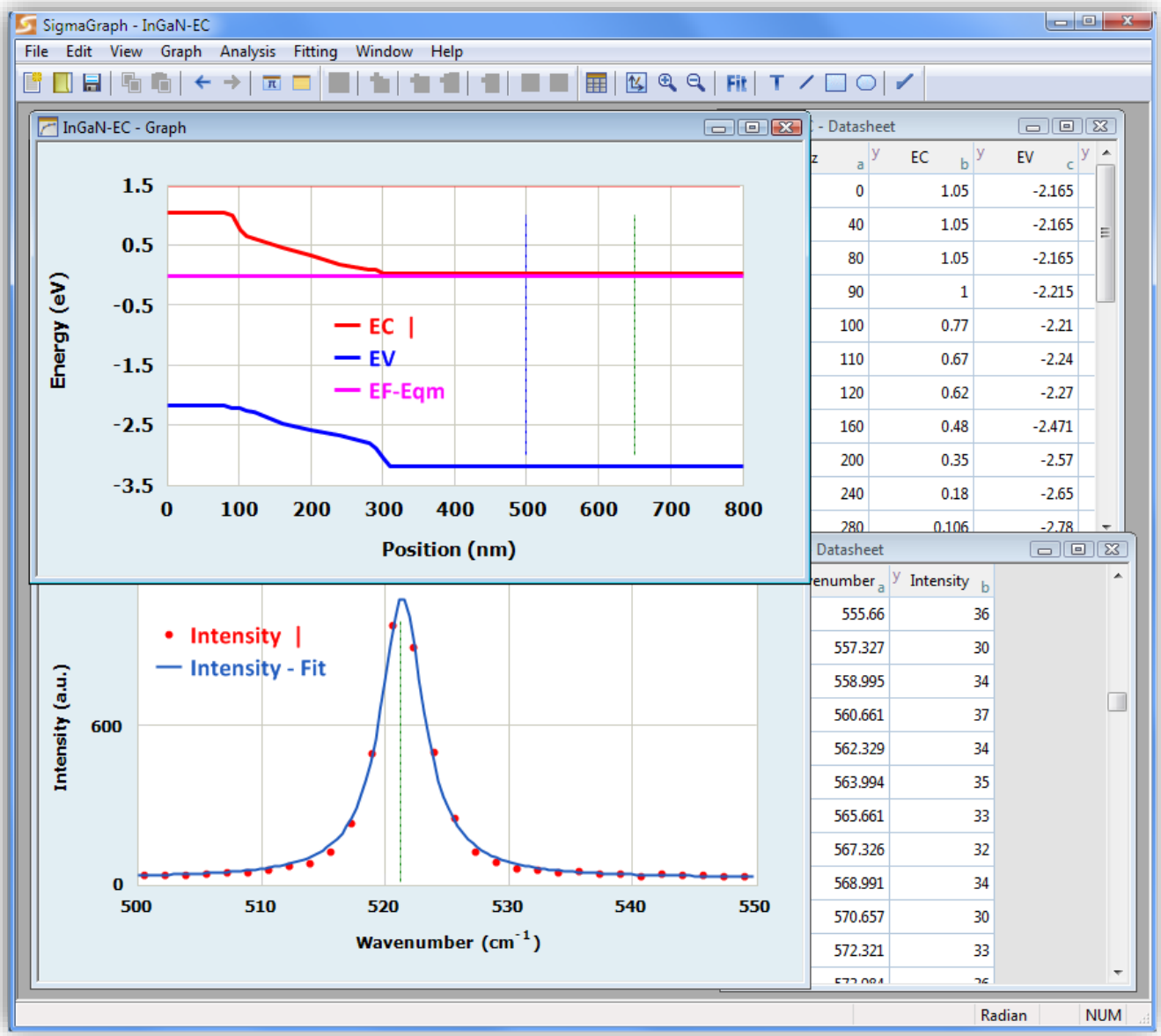
↗ New in SigmaGraph v2.6: • Enhanced Lua scripting code editor with syntax highlighting, line numbering, markers (bookmarks), code completion, etc. • New data importing options • Help system redesigned (integrated HTML Help and PDF) • New fitting models (multi-peak Lorentz and Gauss) • User interface enhancements • Performance optimization • etc.

1. Install

Download the **portable version** (**sigmagraph_portable_windows.zip**), unzip in any location (USB key for example) and run *SigmaGraph.exe* located in the *bin* directory.

The SigmaGraph package contains two executables in the *bin* directory:

- **SigmaGraph.exe**: main SigmaGraph component.
- **SigmaConsole.exe**: mathematical console.



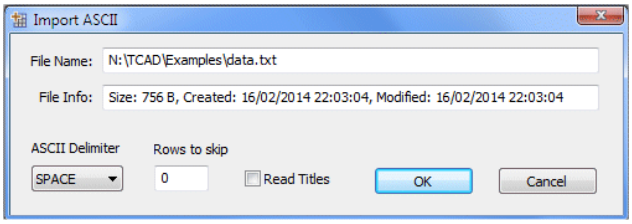
2. Quick Start Guide

2.1 Importing and Analyzing Data

2.1.1 Importing Data

Click the '*Open*' button (in the main toolbar) and select the file to be imported.

The Import ASCII dialog appears, giving you the possibility:



(i) To select the delimiter (TAB, SPACE or ;).

(ii) To set the number of rows to skip at the beginning of the file.

(iii) To set the number of rows to read from the file.

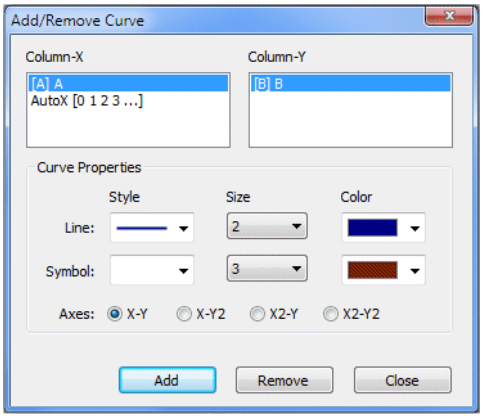
(iv) To select whether or not to read titles into the datasheet.

When ready, press '*OK*' to import the file content into a new datasheet.

2.1.2 CREATING A LINE/SCATTER GRAPH

To plot the imported data:

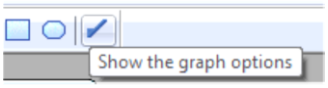
(i) Click the '*Add Curve*' button (or click the '*Graph/Add Curve*' menu) and select the X and Y columns to be plotted. You can change the column properties (and modify the column type: X or Y…): right click and choose '*Properties*'.



(ii) Set the line and symbol style and color.

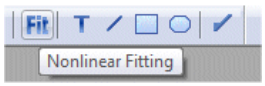
(iii) Click the '*Add*' button. And close the '*Add Curve*' dialog. You can choose the curve axes by clicking the X-Y, X-Y2, X2-Y or X2-Y2 button.

You can modify all the graph properties (curve style, colors, fonts, axis, scale, etc.) in the *'Graph Options'* dialog by clicking the '*Graph Options'* button in the toolbar.

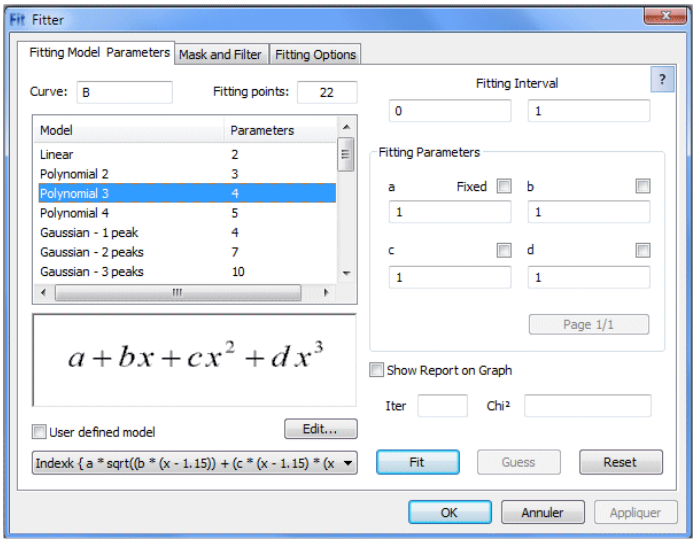


2.1.3 FITTING

To analyze the curve previously created, you can use the **SigmaGraph** fitting functionality (in this guide, we are using the Lorentz peak function):

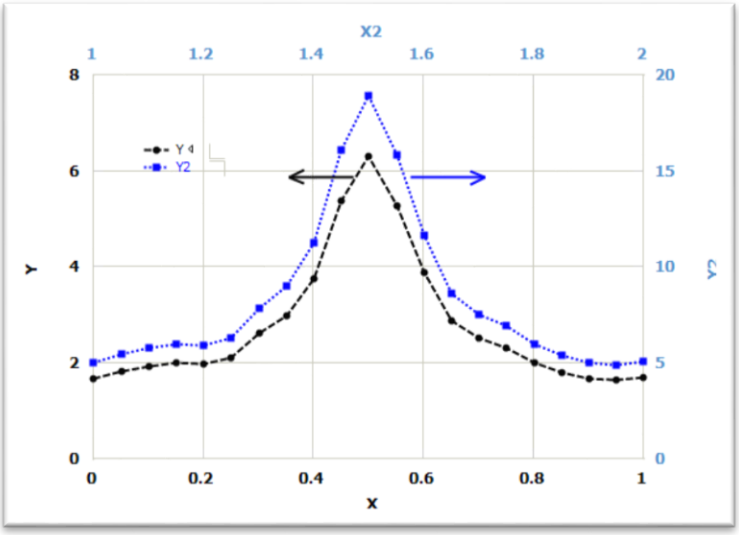


(i) Click the '*Fit*' button and choose the Lorentzian model (or click the '*Fitting/Lorentzian*' menu). Set the fitting number of points to 50. Click the '*Show Fitting Report on Graph*' button (in order to print out the fitting result on the graph window).



(ii) The model used in this guide (Lorentz peak function) has four parameters: a, b, c and d. In order to calculate quickly and precisely the best estimation of these parameters values, the fitting algorithm needs an initial guess. This can be done automatically for the Lorentzian model. Simply click the 'Guess' button. When done, click the 'Fit' button.

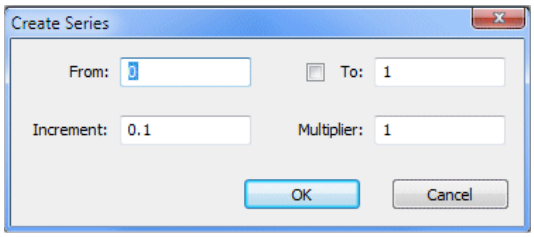
(iii) Press 'OK' to finish. The fitting curve will be plotted and the parameters values will be printed out on the graph window. You can view the fitting datasheet by clicking the 'View/Fit Datasheet' menu.



2.1.4 Adding Error Bars

You can add error bars to your data:

(i) Click the *'Graph/Errors Bars'* menu. In the *'Error Bar Settings'* dialog, click the *'Y Err'* and *'Fixed Value'* buttons. Set the fixed value to 0.4.

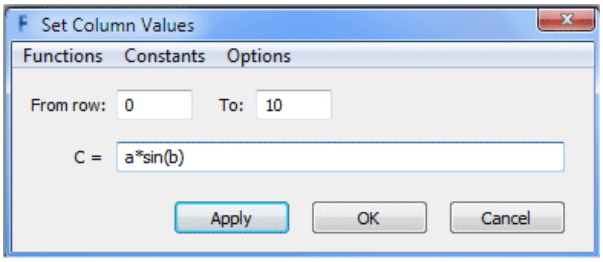


(ii) Choose the error bars line style, size and color.

(iii) Click *'Apply'* and close the dialog.

The error bars are added to your graph.

2.1.5 Exporting Graph

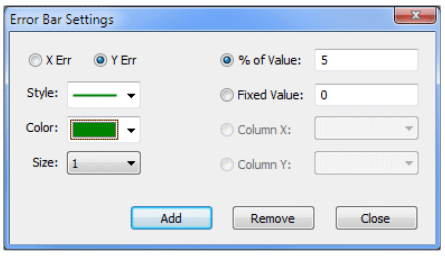


You can export graph in EMF (Enhanced Metafile) or SVG (Scalable Vector Graphics) format: click the *'File/Export Graph'* menu (or *'File/Save As…'*), choose the file type and name and press *'OK'*.

2.1.6 Auto Export Data and Graph

SigmaGraph include an option to automatically export data in text file and plot in SVG format when the document is saved. To enable or disable this option, click the *File/Auto Export* menu.

2.2 Plotting function



To plot a mathematical function:

(i) Create a new document by clicking the *'New'* button in the main toolbar. A new datasheet will be created, with two blank columns. Right click the first column (A) and choose *'Create Series'*. Set the range from 0 to 1 with 0.1 as increment.

(ii) Right click the second column (B) and choose *'Set Column Values'*. Write the formula as shown is the figure, and press *'Apply'* and then *'OK'*.

(iii) To plot the function, right click the second column (B) and choose *'Add/Remove Curve'*. Press *'Add'* and then *'OK'*.

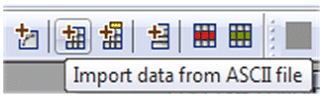
3. Help Topics

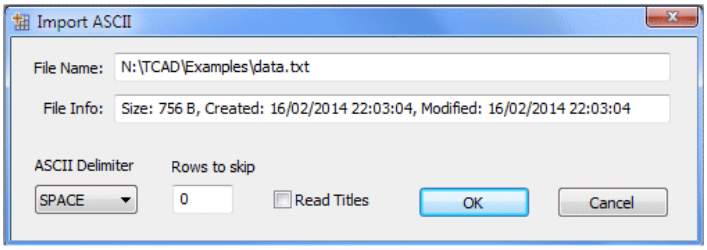
3.1 Working with Datasheet

3.1.1 Creating a datasheet

To create a datasheet you can either:

(i) **import a data from an ASCII file**. Click the *'Open'* button (in the main toolbar) and select the file to be imported. The Import ASCII dialog appears, giving you the possibility to select the delimiter (TAB, SPACE or ;), to set the number of rows to skip at the beginning of the file, to set the number of rows to read from the file. When ready, press *'OK'* to import the file content into a new datasheet. A **posted note** is associated to the datasheet. To view and modify it, select *'View/Note'*.

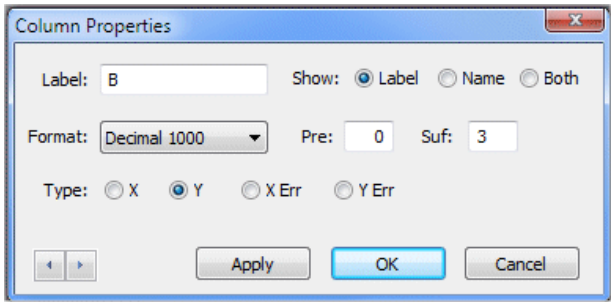




(ii) **create a blank document** by clicking the *'New'* button (or click *'File/New'* menu). A new datasheet, with two blank columns, will be created.

3.1.2 Column properties

To show and set the column properties, right click the column and choose *'Properties'*. The *'Column Properties'* appears giving you the possibility to:



(i) To change the column label. You can select to show column name (A, B, C, and so on), label or both.

(ii) To change numeric format of the column data. You can choose between decimal (1000.0) and scientific (1e3). You can also set the format prefix (*'Pre'* = number of digits before the decimal point) and suffix (*'Suf'* = number of digits after the decimal point).

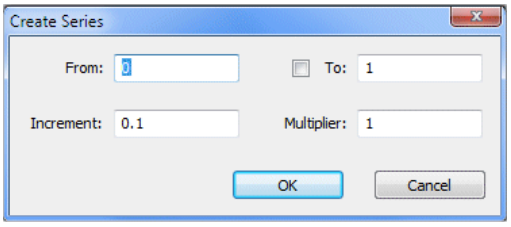
(iii) To change the column type: X, Y, X-Err (Error Bars on X) and Y-Err (Error Bars on Y).

When ready press *'Apply'* and then *'OK'*. You can change the active column by using the left-right arrows.

3.1.3 Inserting, Appending and Deleting columns

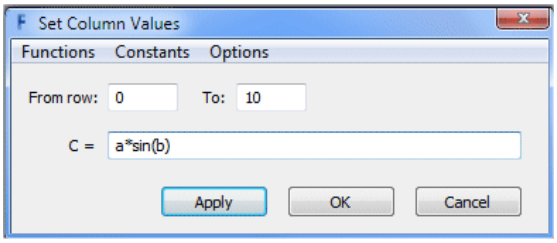
To **insert** a new column after an existing one, right click the latter and choose *'Insert Column'*. A new blank column will be created. You can also **append** columns (inserting column after the last one) by right clicking on any column and choose *'Append columns'*. To **delete** a column, right click it and choose *'Delete columns'*.

3.1.4 CREATING SERIES



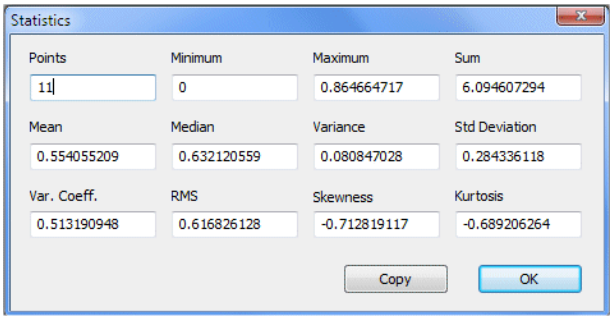
You can fill a column with a series of numbers, by setting the initial (*'From'*) value $\mathbf{y_0}$, Increment **δ** and, if needed, a multiplier **m**. At index **i** (varying from 0 to n-1 (n = number of rows)), the value $\mathbf{y_i}$ is given by $\mathbf{y_i = m * y_{i-1} + \delta}$. To create a series, right click the column and select *'Create Series'*. The *'Create Series'* dialog appears, giving you the possibility to set the initial value $\mathbf{y_0}$, the increment **δ** and the multiplier **m**. If you check the *'To'* button, the calculation will be stopped when $\mathbf{y_i}$ reach the final (*'To'*) value.

3.1.5 SETTING COLUMN VALUES



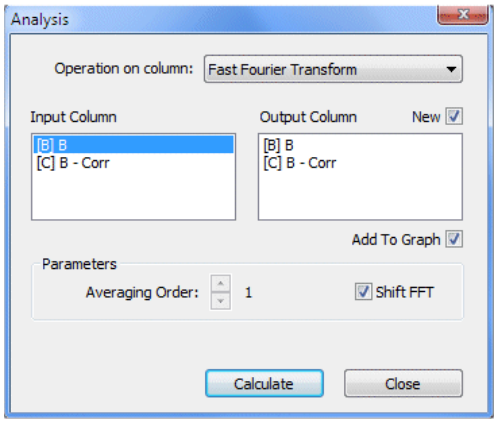
You can set the column values by using any mathematical expression. Right click the column and choose *'Set Column Values'* With the shown dialog you can set the column values. The columns are alphabetically named: **A**, **B**, **C**, … (the names are not case sensitive). To set the column values, write down the expression and click the *'Apply'* button. You can use the dialog menu to quickly include functions and constants in your expression. For more details cf. the paragraph on the mathematical console.

3.1.6 STATISTICS ON COLUMN



To view the column descriptive statistics, right click the column and select *'Statistics'*. The statistics dialog appears with the standard parameters: Minimum, Maximum, Sum, Mean, Median, Variance, Standard Deviation, Variance Coefficient, Root Mean Square (RMS), Skewness and Kurtosis. You can copy statistics to the clipboard by clicking the *'Copy'* button.

3.1.7 ANALYSIS ON COLUMN



You can perform some useful calculations on column (e.g. integrate, differentiate, average, calculate the FFT (Fast Fourier Transform) or the autocorrelation function of a transient signal), by selecting the *'Analysis'* menu. You can then select, in the *'Output Column'* list, an existing column to update with the calculation results, or create a new column by checking the *'New'* button. To add the integral, derivative, averaged or FFT curve to the graph, check the *'Add to Graph'* button. Note that you can program you own algorithms by using the **SigmaGraph scripting capabilities**.

SigmaGraph use a centered difference formula to approximate the **derivative**:

$$f'(x_i) \sim \frac{f(x_{i+1}) - f(x_{i-1})}{x_{i+1} - x_{i-1}}$$

For **integration**, the trapezoidal rule is used.

The **averaging** formula used is the following (where *m* is the averaging order):

$$\bar{f}(x_i) = \frac{1}{2m}\left(\sum_{j=i-m}^{i-1} f(x_j) + \sum_{j=i+1}^{i+m} f(x_j) \right)$$

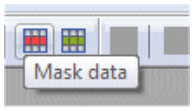
For **FFT**, the module ($\sqrt{(real\ part)^2 + (imaginary\ part)^2}$) is calculated and normalized by the number of points (this number of points should be a power of two). The frequency is calculated based on the sampling period (X-step in second).

The **Autocorrelation** function is calculated by using the following formula:

$$R(k) = \left(\sum_{i=1}^{N-k} (y_i - \bar{y}) * (y_{i+k} - \bar{y}) \right) / \left(\sum_{i=1}^{N} (y_i - \bar{y})^2 \right)$$

For equi-spaced measurements $y_1, y_2, \ldots, y_N$.

3.1.8 MASKING DATA



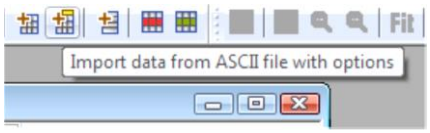
You can mask selected cells in a column (or group of columns) by right clicking and selecting *'Mask'*. The masked cell will be printed out in red and the corresponding data will not be plotted or used in fitting. You can invert mask or unmask cells: right click the selected cells and choose *'Invert Mask'* or *'Unmask'*. You can also use the datasheet toolbar buttons to perform these operations.

3.1.9 SORTING COLUMN

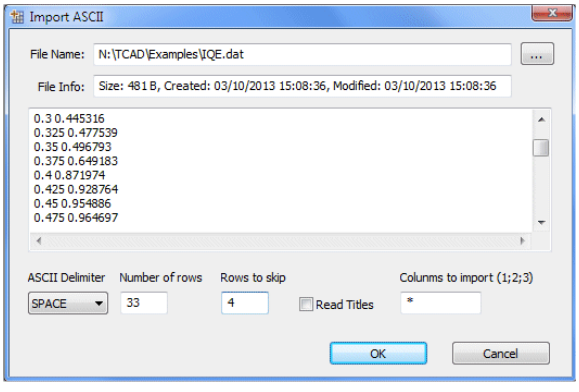
To sort a column, select it, right click and choose *'Sort Column'* (ascending or descending).

3.1.10 IMPORTING ASCII DATA

SigmaGraph gives you two ways to import ASCII data into a datasheet:

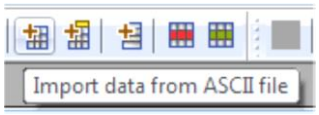


(i) Importing with options. Select the datasheet to import in, and click *'Import with options'* toolbar button. The importing dialog will appears. Click the browse button ('…') and select the ASCII file. The file content and info will be displayed. You can select the **delimiter** (TAB, SPACE or ;), set the number of **rows to skip** at the beginning of the file, set the number of rows to read from the file, and select whether or not to **read titles** into the datasheet. You can select the **columns to import** by entering their indexes (example: **1;3** means importing the first and the third



columns) or leaving the default behavior (* means import all columns). When ready, press *'OK'* to import the selected columns into datasheet.

(ii) Importing directly from file. Select the datasheet to import in, click *'Import'* toolbar button, and choose the ASCII file to import from. The default importing options are used (ASCII separator, rows to skip…).



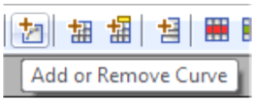
3.1.11 Copy/Paste data

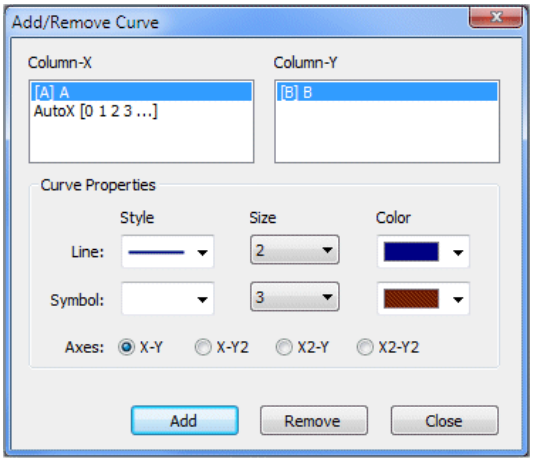
You can copy/paste cells, rows, column… by selecting the data and clicking the *'Copy'* toolbar button (or *Ctrl+C* or right-click and select *'Copy'*). To paste data, select destination cells, rows or column and click the *'Paste'* toolbar button (or *Ctrl+V* or right-click and select *'Paste'*).

3.2 Working with Graph

3.2.1 Create graph/add curve

When you create a **SigmaGraph** document, a blank graph is automatically associated with the datasheet. When the datasheet is active, you can show or hide the graph by clicking the *'View Graph'* toolbar button or select *'View/Graph'* menu. To add curve to the graph, select the column to be plotted and click the *'Add Curve'* toolbar button (or right click the column and select *'Add/Remove Curve'*, or select the *'Graph/Add Curve'* menu). With the 'Add/Remove Curve' dialog you can select the X and Y columns, and set the curve style (line/symbol), size and color. You can also remove or modify the existing curves by selecting the plotted column and press the *'Modify'* or *'Remove'* button.

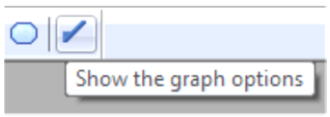




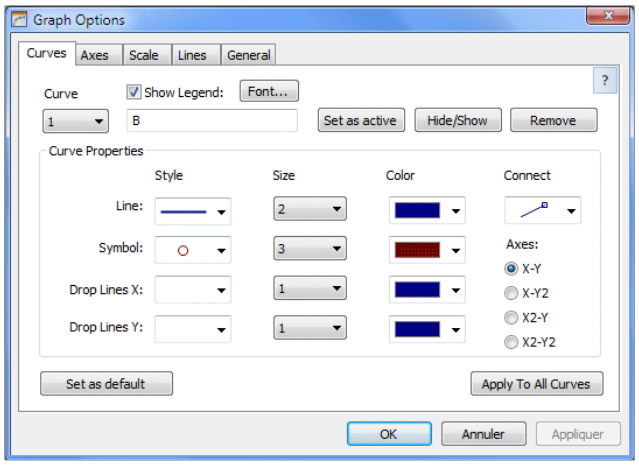
You can choose the curve axes by clicking the X-Y, X-Y2, X2-Y or X2-Y2 button.

3.2.2 Graph options

You can modify the graph options, including curve style, axis, scale… To show the *'Graph Options'* dialog, click the toolbar button or select *'Graph/Options'* menu.



3.2.2.1 CURVES

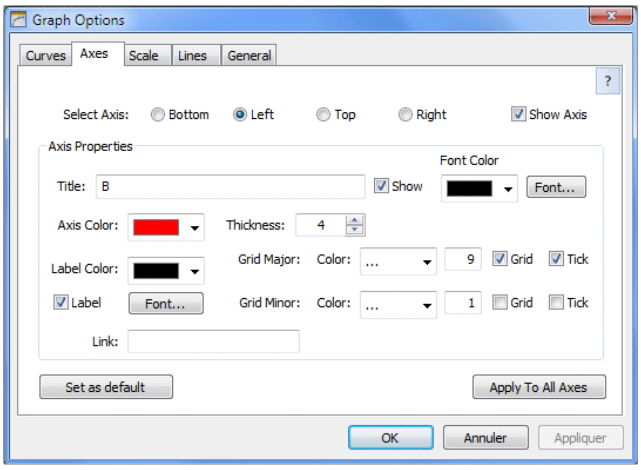


To change the selected curve legend font, click the *'Font'* button. You can modify also the line, symbol style, size and color of each curve. You can add X and Y drop lines and change their style, size and color.

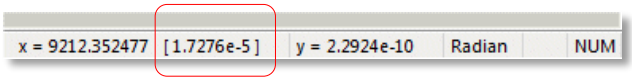
To activate the selected curve, click the *'Set as active'* button (the active curve is used for the fitting for example). To hide or show curve, press *'Hide/Show'*. To remove the curve from graph, press *'Remove'*.

You can apply the current curve style to all curves by clicking the *'Apply to All Curves'* button. To set the current style as the default one, click the *'Set as default'* button.

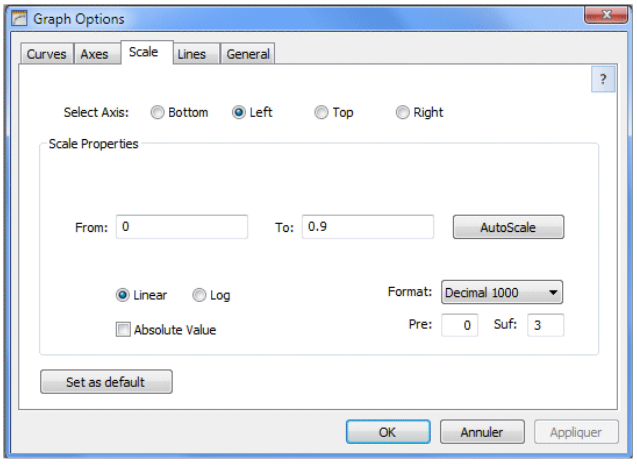
3.2.2.2 AXES



To change axis properties, select the axis (bottom (X), left (Y), top (X2) or right (Y2)). To show or hide the axis, check or uncheck the *'Show Axis'* button. You can modify the axis title, show or hide this title, choose its font and color. The axis style (color, thickness, and label) can be be changed easily. The grid and tick options can also be modified by selecting color, grid count, … You can apply the current axis style to all axes by clicking the *'Apply to All Axes'* button. To set the current style as the default one, click the *'Set as default'* button. In the *'Link'* input, you can optionally enter the axis link formula (<u>example</u>: ***1/(2*pi*x)*** for Bottom axis, or ***20*log(y)*** for the Left axis), and the corresponding value will be displayed in the status bar when the mouse moves over the graph.



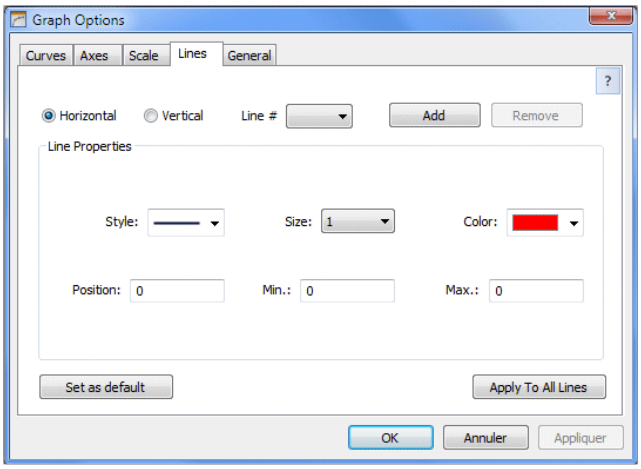
3.2.2.3 SCALE



The X and Y scale can be manually entered, or automatically calculated. For each axis (bottom (X), left (Y), top (X2) or right (Y2)), select the range manually (*'From'* and *'To'* values) or click the *'AutoScale'* button. You can use the logarithmic or Linear scale (*'Log'* and *'Linear'* buttons). The numerical format of the axis

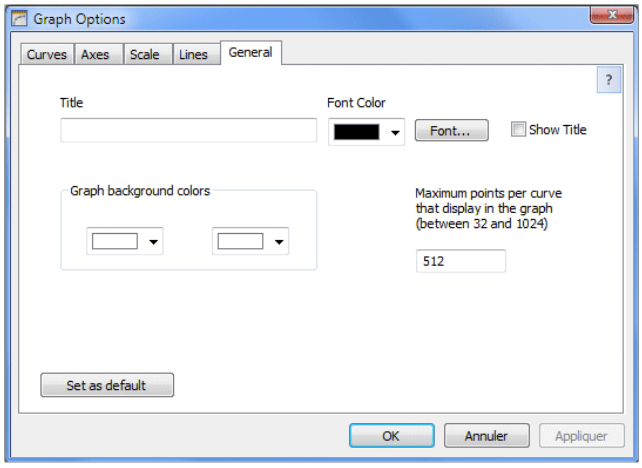
label can be selected (decimal or scientific), and the format prefix (*'Pre'* = number of digits before the decimal point) and suffix (*'Suf'* = number of digits after the decimal point) can be entered. To set the current style as the default one, click the *'Set as default'* button.

3.2.2.4 LINES



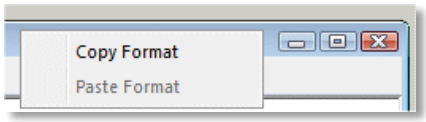
You can add horizontal (y = constant) or vertical (x = constant) lines to the graph. To do so, select style, size, color, position (the x-position for vertical line, and y-position for horizontal line), line limits (min. and max.) and click *'Add'*. You can change the line properties and click *'Apply'*, or remove the line by clicking the *'Remove'* button.

3.2.2.5 GENERAL OPTIONS



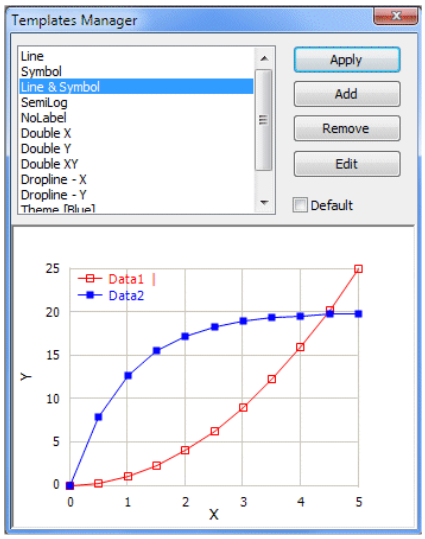
The general options include the graph title text, font, color and the maximum points displayed in the graph. To show or hide the graph title, check or unchek the *'Show Title'* button. To change the graph or window background color, click inside the rectangle (in the center for the graph, and close to the border for the window) and select the color. To set the current style as the default one, click the *'Set as default'* button.

3.2.3 COPY/PASTE GRAPH FORMAT



With **SigmaGraph** you can apply one graph format (colors, fonts, axis and curves style…) to another graph. Simply right click the graph title bar and select *'Copy Format'*. Then select the graph to modify, right click its title bar and select *'Paste Format'*. The copied format will be applied to the selected graph.

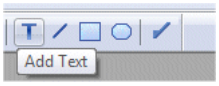
3.2.4 TEMPLATES MANAGER



With the **SigmaGraph** Templates Manager (*'Graph/Templates'* menu), you can customize and save all of the properties of your graph and reuse them very quickly. With this useful feature, you will save time by using a predetermined style (i.e. template) to create a new graph. With the

Template Manager, you can add, remove or edit up to sixteen templates, covering your needs. When clicking the *'Apply'* button, the selected template will be applied to the active graph.

3.2.5 Adding text

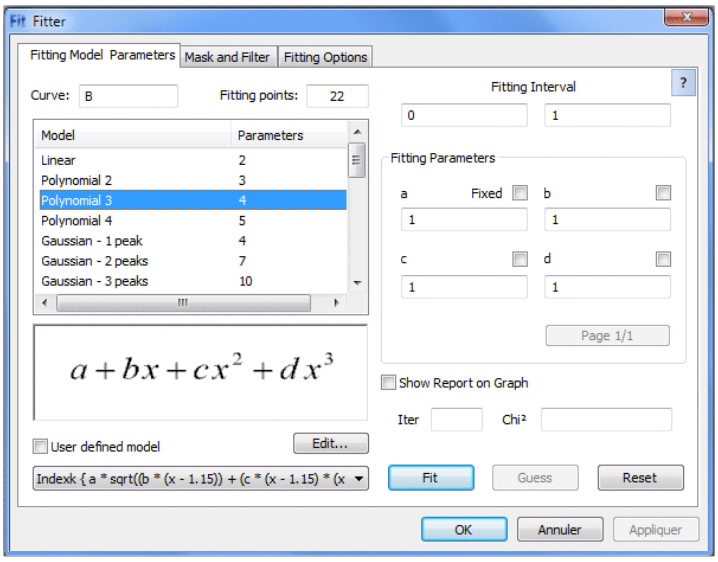


To add text to the graph, click the *'Add Text'* toolbar button and then click somewhere in the graph window and enter the text. To modify an existing text, double click it. You can change the text font, color… To move a text, simply select it and move it with the mouse. You can create superscript or subscript text by using **^** or **_** character. Example: **x^2** will be displayed as x^2 and **x_2** will be displayed as x_2. You can use parenthesis if you need more complex expression. Example: **T^(3/2)** will be displayed as $T^{3/2}$.

3.2.6 Adding Line, Rectangle or Ellipse

To add line, rectangle or ellipse to the graph, click the *'Add Line'*, *'Add Rectangle'* or *'Add Ellipse'* toolbar button and then click somewhere in the graph window, hold down the mouse button and draw the line, rectangle or ellipse. To modify an existing item, double click it. You can change the size, style, color, fill, arrow… To move item, simply select it and move it with the mouse.

3.3 Curve Fitting



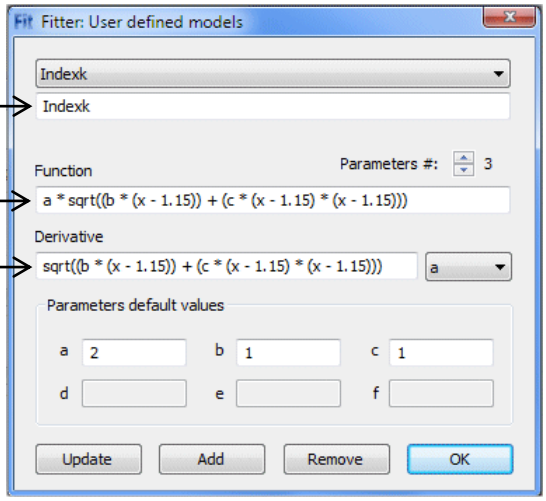
SigmaGraph gives you the fitting functionality you need in order to analyze data (e.g. data obtained from experiment). 24 models are available, including linear, polynomial, exponential, Gaussian (up to 5 peaks), Lorentzian (up to 5 peaks), Pearson VII, logistic, power…

To perform a curve fitting, select the graph and activate the curve (*'Graph Options / Activate'* or right click the curve legend and select *'Activate'*). Then click the *'Nonlinear Fitting'* toolbar button, or select *'Fitting/ Nonlinear Fitting'* menu. The fitting dialog appears letting you to select the model, mask or smooth data or set fitting options.

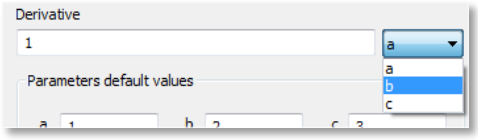
- The Fitting **Model** Parameters: you can select the model to use by the fitting algorithm. The model function is shown, with the parameters: *a*, *b*, *c*, …. In order to calculate quickly and precisely the best estimation of these parameters values, the fitting algorithm needs an initial guess. This can be done automatically for some models (Gauss, Lorentz, Pearson VII, Exponential,…). Simply click the *'Guess'*

button. You can select the X fitting interval (by default it's the same than the data range), and change the fitting number of points (usually few times the data number of points). When done, click the *'Fit'* button. The fitting curve will be plotted and the parameters values will be printed out on the graph window (if the *'Show Fitting Report on Graph'* button is checked).

You can also define and use your own fitting models. Click the *'Edit…'* button to show the user-defined models window:



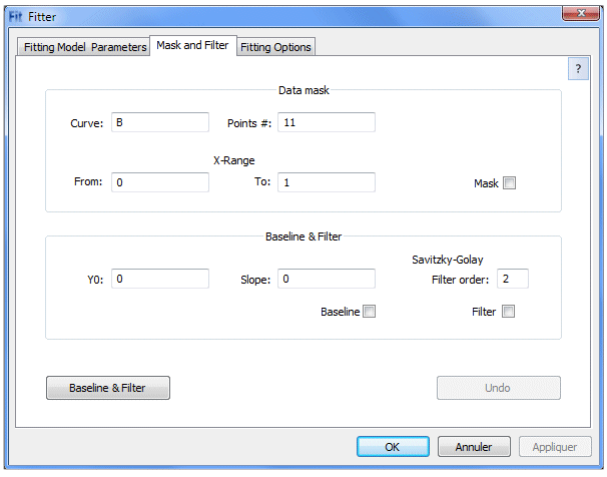
- ✓ Enter the name of your fitting model;
- ✓ Define the number of the fitting parameters (between 2 and 6);
- ✓ Enter the fitting function f(x,a,b,…). The fitting parameters are named ***a***, ***b***, ***c***, ***d***, and so on;
- ✓ Enter the derivative (δf/δa) expression;
- ✓ Give the parameters default/starting values;
- ✓ Click the *'Add'* button;
- ✓ Select parameter ***b*** and enter the derivative (δf/δb) expression, and then ***c***, ***d***, and so on.



- ✓ Click the *'Update'* button to save the fitting model;
- ✓ Click *'OK'* to close the user-defined models window and return to the fitter main window.

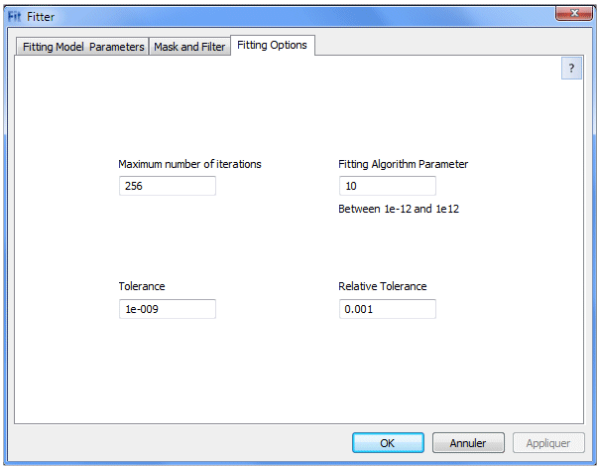
Now you can select your fitting model from the list and use it.

- **Mask and Smooth**: before performing curve fitting, you can mask, smooth and/or subtracting a baseline from the data. To select the fitting range in the data (and then mask the data outside this range), give the start (*'From'*) and the end (*'To'*) values for X-Range and check the *'Mask'* button. To smooth the data, enter the smooth order (high number means more smooth) and check the *'Smooth'* button. To subtract a baseline



(y = y0 + slope*x) from the curve, enter the line parameters (y0 and slope) and check the *'Baseline'* button.

When ready, click the *'Baseline & Smooth'* button to perform calculations.

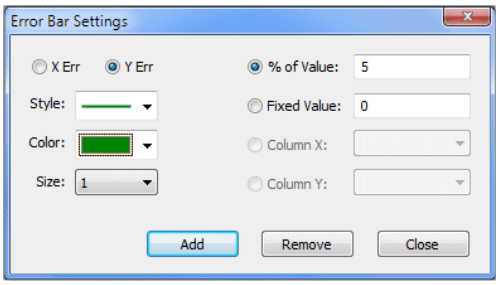


- You can set some **fitting algorithm options** like the maximum number of iterations, the relative and absolute tolerance…

In the '*Fitting Model Parameters*' tab, press '*OK*' to finish. The fitting curve will be plotted and the parameters values will be printed out on the graph window. You can view the fitting datasheet by clicking the '*View/Fit Datasheet'* menu or the fitting report by clicking '*View/Fit Report*'.

3.4 Error Bars

To add error bars to your data, click the '*Graph/Errors Bars'* menu. In the *'Error Bar Settings'* dialog, click the '*X Err'* or '*Y Err'* to add error bars to the X data or the Y data. You can choose to fix the error value by clicking the '*Fixed Value'* button, or give the error percentage by clicking the '*% of Value'* button. Then set the fixed or percentage value. You can also select one existing error column by clicking on *'Column X'* or *'Column Y'* and selecting the column you want. You can choose the error bars line style, size and color. When ready click '*Apply'* and close the dialog.



3.5 Exporting Data and Graph

To export datasheet content in ASCII file, select *'File/Export Data'* menu, choose the file and *OK*.

You can export graph in EMF (Enhanced Metafile) or SVG (Scalable Vector Graphics) format by clicking the '*File/Export Graph'* menu. **SigmaGraph** include an option to **automatically export data in text file and plot in SVG format** when the document is saved. To enable or disable this option, click the *File/Auto Export* menu.

3.6 Printing

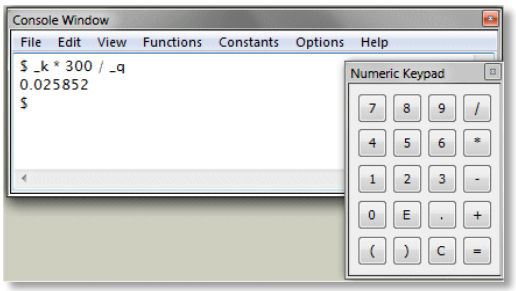
To print a datasheet or graph, select *'File/Print'* menu or click the *'Print'* toolbar button.

3.7 Saving/Opening document

To save the active document (datasheet, graph, note, and so on), select *'File/Save'* menu or click the *'Save'* toolbar button (or *CTRL+S*). The **SigmaGraph** file extension is **sid**.

To open a **SigmaGraph** document, select *'File/Open'* menu or click the *'Open'* toolbar button (or *CTRL+O*).

3.8 Working with the Mathematical Console



SigmaConsole is an advanced mathematical console. It supports the most common and useful functions. It's easy to use: to evaluate an expression, simply write it, using operators (+ - * / ^), parenthesis and mathematical functions and press ENTER. You can also use the numeric keypad to enter numbers and operators. You can set variables (with any non-reserved name), using fundamental constants, etc. The **SigmaConsole** menu gives you an easy way to use the software functionality.

The following mathematical **functions** are supported:

exp(x)	// exponential
ln(x)	// natural logarithm
log(x)	// decimal logarithm
log2(x)	// base-2 logarithm
pow(x,n)	// x^n
sin(x)	// sine
cos(x)	// cosine
tan(x)	// tangent
asin(x)	// arc sine
acos(x)	// arc cosine
atan(x)	// arc tangent
sinh(x)	// hyperbolic sine
cosh(x)	// hyperbolic cosine
tanh(x)	// hyperbolic tangent
abs(x)	// absolute value
sqrt(x)	// square root
ceil(x)	// ceiling, the smallest integer not less than x
floor(x)	// floor, the largest integer not greater than x
int(x)	// integer part of x
fmod(x,y)	// x modulo y
erf(x)	// error function
j0(x)	// Bessel function of x of the first kind of order 0
j1(x)	// Bessel function of x of the first kind of order 1
jn(n,x)	// Bessel function of x of the first kind of order n
y0(x)	// Bessel function of x of the second kind of order 0

y1(x)	// Bessel function of x of the second kind of order 1
yn(n,x)	// Bessel function of x of the second kind of order n
bern(x)	// Bernoulli function: x / (exp(x) – 1)
hypot(x,y)	// hypotenuse, sqrt($x^2 + y^2$)
min(x,y)	// smallest value of x and y
max(x,y)	// largest value of x and y
rand(x)	// random number between 0 and 1 (if x ≠ 0 then initialize the generator)
time()	// elapsed time in seconds since January 1, 1970
sign(x)	// sign of x (-1 if x < 0, +1 if x > 0 and 0 if x = 0)
exp2(x)	// 2^x
log2(x)	// logarithm base 2
cbrt(x)	// cubic root
hypot(x,y)	// sqrt(x^2+y^2)
erf(x)	// error function
erfc(x)	// complementary error function
lgamma(x)	// ln(gamma(x))
tgamma(x)	// gamma(x)
trunc(x)	// nearest integer
round(x)	// nearest integer, rounding
rint(x)	// rounds the floating-point to an integer

Constants:

Pi	// π
_q	// electron charge
_m	// electron mass
_k	// Boltzmann constant
_h	// Planck constant
_c	// speed of light in vacuum
_e	// vacuum permittivity
_n	// Avogadro constant

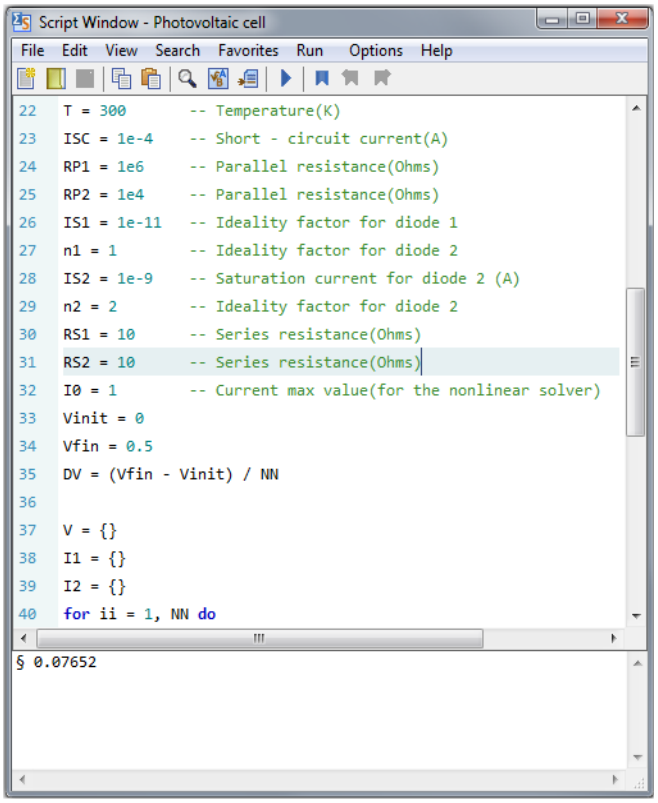
Commands:

format short	// set the numerical format to short real
format long	// set the numerical format to long real
format int	// set the numerical format to integer
help	// show help
exit	// exit the application

3.9 SIGMAGRAPH SCRIPTING

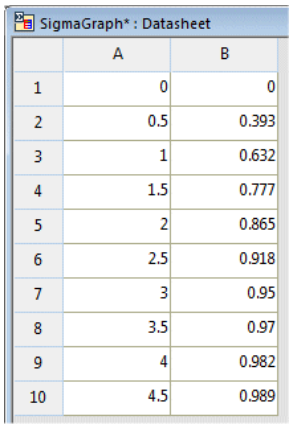
3.9.1 PRESENTATION

SigmaGraph integrates the Lua scripting language, which gives you the possibility to programmatically control the application and program you own algorithms. **Lua** was chosen to be embedded in SigmaGraph mainly because of its high speed and its relatively small size. To learn Lua, you can read the official documentation at Lua official website (http://www.lua.org/manual/) or tutorials.



To write and run script, open the Script Window by selecting *'View/Script Window'* menu (or press F11 key). The script output will be redirected to the Output Window. The SigmaGraph editor supports **syntax highlighting**, **line numbering**, **markers** (bookmarks), **code completion**, etc. You can customize all the editor options (font, colors, line spacing, ...); Menu/Options.

With SigmaGraph scripting capabilities, you can either control the application throw three objects ('classes'): **Doc**, **Data** and **Plot** or write and run you own general-purpose Lua programs. With the **Doc** class, you can access document specific methods, like creating, saving, opening or closing document. With the **Data** class, you can manipulate datasheet: adding, deleting columns or rows, setting cells values, masking/unmasking cells … With the **Plot** class, you can control graph: adding or removing curves, changing axis style …



	A	B
1	0	0
2	0.5	0.393
3	1	0.632
4	1.5	0.777
5	2	0.865
6	2.5	0.918
7	3	0.95
8	3.5	0.97
9	4	0.982
10	4.5	0.989

Note that all the **Doc** methods are accessible to **Data** and **Plot** classes (throw the inheritance mechanism).

Note also that all the SigmaConsole mathematical functions are accessible in the script with the same syntax (cf. §3.8).

In order to use SigmaGraph classes, the first step is to create a class instance:

```
tD = Data:create() -- create a Data class instance
```

When class instance created, you can use all the class methods.

Example: The following script will create a new datasheet named “SigmaGraph” and fill the first column A with values from 0 to 4.5 with 0.5 step and the second column B by a formula. The B versus A curve is then added to graph:

```
cls()                                   -- clear the Output Window
tD = Data:create()                      -- create an instance of the Data class
tD:new("SigmaGraph")                    -- create a new datasheet named SigmaGraph
x = {}
y = {}
for ii=1,10 do                          -- filling columns A and B
    x[ii] = (ii - 1) * 0.5
    y[ii] = 1 - exp(-x[ii])
end
tD:set("A", x)
tD:set("B", y)
tP = Plot:create()                      -- create an instance of the Plot class
tP:add("A", "B", 1)                     -- add B versus A curve to the graph, Line style
```

3.9.2 Reference

3.9.2.1 Mathematical functions

With SigmaScript, the math functions are mapped to global functions (e.g. : user can use **cos** or **math.cos**).

Below is a summary of the Lua math functions:

math.abs	math.acos	math.asin
math.atan	math.atan2	math.ceil
math.cos	math.cosh	math.deg
math.exp	math.floor	math.fmod
math.frexp	math.huge	math.ldexp
math.log	math.log10	math.max
math.min	math.modf	math.pi
math.pow	math.rad	math.random
math.randomseed	math.sin	math.sinh
math.sqrt	math.tanh	math.tan

And a summary of the SigmaScript global math functions:

abs	acos	asin
atan	atan2	ceil
cos	cosh	deg
exp	floor	fmod
frexp	huge	ldexp
log	log10	max
min	modf	pi
pow	rad	random
randomseed	sin	sinh
sqrt	tanh	tan

NB: Lua gives the neperian logarithm the name log and the decimal logarithm is named log10, as in C language. Mathematical Console uses ln for neperian logarithm and log for decimal logarithm.

Special math functions are included (**lmath** namespace), extending the Lua math functions.

Summary of lmath functions and constants:

Functions:

lmath.exp2(x)	2^x
lmath.log2(x)	logarithm base 2
lmath.cbrt(x)	cubic root
lmath.hypot(x,y)	sqrt(x²+y²)
lmath.erf(x)	error function
lmath.erfc(x)	complementary error function
lmath.lgamma(x)	ln(gamma(x))
lmath.tgamma(x)	gamma(x)
lmath.trunc(x)	nearest integer
lmath.round(x)	nearest integer, rounding
lmath.isinf(x)	number is infinite ?
lmath.isnan(x)	not a number ?
lmath.isnormal(x)	number is normal ?
lmath.asinh(x)	
lmath.acosh(x)	
lmath.atanh(x)	
lmath.gauss(x,b,c)	G(x) = exp(-(x - b)² / 2c²)
lmath.lorentz(x,b,c)	L(x) = ((1/pi)*c / ((x - b)² + c²))

Constants:

Universal constants in international units (SI)

lmath.q	Electron charge (in C)
lmath.me	Electron mass (kg)
lmath.kb	Boltzmann constant (J/K)
lmath.h	Planck constant (Js)
lmath.c	Speed of Light in vacuum (m/s)
lmath.na	Avogadro constant (1/mole)

3.9.2.2 STATISTICS FUNCTIONS

The **data** namespace includes functions to calculate the descriptive parameters of a list of values, perform data fitting and sorting:

Minimum (`data.min(t)`), Maximum (`data.max(t)`), Sum (`data.sum(t)`), Mean (`data.mean(t)`), Median (`data.median(t)`), Variance (`data.var(t)`), Standard Deviation (`data.dev(t)`), Coefficient of Variation (`data.coeff(t)`), Root Mean Square (`data.rms(t)`), Skewness (`data.skew(t)`) and Kurtosis excess (`data.kurt(t)`).

The formulas used are as below:

Mean $\mu = \frac{1}{N}\sum_{i=0}^{N-1} x_i$	Variance $\sigma^2 = \frac{1}{N-1}\sum_{i=0}^{N-1}(x_i-\mu)^2$
Skewness $Skew = \left(\frac{1}{N}\sum_{i=0}^{N-1}(x_i-\mu)^3\right) / \left(\frac{1}{N}\sum_{i=0}^{N-1}(x_i-\mu)^2\right)^{3/2}$	Kurtosis $Kurt = \left[\left(\frac{1}{N}\sum_{i=0}^{N-1}(x_i-\mu)^4\right) / \left(\frac{1}{N}\sum_{i=0}^{N-1}(x_i-\mu)^2\right)^2\right] - 3$

All these `data` functions take a Lua table as argument, with 16384 maximum number of elements.

Example:

```
-- data
cls()
t = {1,1,2,3,4,4,5}
m = data.mean(t)
print(m)
```

With the **data** module, you can also perform **data fitting** by using the function `fit`:

```
pars,chi,iters,msg = data.fit(model, x, y, fpar, ipar, tol, iters)
```

where:

`model` is one of the builtin fitting models: **"Linear", "Poly2", "Poly3", "Poly4", "Gauss1", "Gauss2", "Gauss3", "Gauss4", "Gauss5", "Lorentz1", "Lorentz2", "Lorentz3", "Lorentz4", "Lorentz5", "Pearson", "Exp", "Exp2", "ExpDec", "ExpDec2", "Hyperbola", "Log", "Power", "Logistic", "Wave".**

`x` and `y` are the data to fit.

`fpar` is the fitting parameters table with the initial values.

`ipar` (optional) is a table with, for each parameter, value 1 if the parameter is varying or 0 if it is fixed.

`tol` (optional) is the desired relative tolerance (10^{-6} by default).

`iters` (optional) is the maximum iterations (100 by default).

The function returns the obtained parameters **pars**, the **chi** value, the number of iterations **iters** and a message **msg** from the fitting engine.

Example:

```
-- Linear Fitting y = a + b * x
cls()
x = {1,1,2,3,4}
y = {2.1,1.8,2.9,3.8,5.2}
fpar = {0.1,0.1}
pars,chi,iters,msg = data.fit("Linear", x, y, fpar)
io.write("a = ", pars[1], " b = ", pars[2], "\n")
```

You can also perform **sort data** by using the function **sort**:

```
ts = data.sort(t, asc)
```

where:

t is the data to sort.

asc = 1 if sort in ascending order or 0 in descending order

The function returns the sorted table **ts**.

Example:

```
-- Sorting data
cls()
t = {1,8,2,5,1,3,9,7}
asc = 1
ts = data.sort(t, asc)
```

3.9.2.3 SIGMAGRAPH DATA ANALYSIS AND PLOTTING FUNCTIONS

The following table summarizes the **Doc**, **Data** and **Plot** methods:

Class	Method	Purpose	Arguments
Doc	create()	Create an instance of the Doc class	–
	new(name)	Create a new document	In: the new document *name* Out: return the document name
	find(name)	Find an existing document	In: the document *name* Out: return the document name
	print()	Print the current document	–
	save()	Save the current document	–
	open()	Open a document	–
	update()	Update the current document	–
	close()	Close the current document	–

Data			
	create()	Create an instance of the Data class (inherited from the Doc class)	–
	new(name)	Create a new datasheet(inherited from the Doc class)	In: the new datasheet *name* Out: return the document name
	dim()	Get the number of columns and rows	Out: number of column and number of rows
	appendcol(dim, ctype)	Append a column to the datasheet	In: the column *dim*ension and the column type (*ctype* = 1 for X-column and 2 for Y-column) Out: return the column name
	insertcol(after, dim, ctype)	Insert a column into the datasheet	In: *after*, the name of the column to insert after; *dim*, the new column dimension; *ctype*, column type (1 for X-column and 2 for Y-column) Out: return the column name
	format(name, fmt)	Set the column numeric format	In: *name*, the name of the column; *fmt*, the numeric format (C-style: "%.3f", "%.6e", "%e", …)
	appendrow(count)	Append rows to the datasheet	In: *count*, the number of rows to append
	insertrow(iafter)	Insert a row into the datasheet	In: *iafter*, the index of the row to insert after
	deletecol(name)	Delete column	In: *name*, the name of the column to delete
	deleterow(ii)	Delete row	In: *ii*, the index of the row to delete
	set(name, ii, val)	Set cell value	In: *name*, the name of the column; *ii*, the index of the cell; *val*, the new cell value
	set(name, t)	Set column values	In: *name*, the name of the column; *t*, table containing the column values
	get(name, ii)	Get cell value	In: *name*, the name of the column; *ii*, the index of the cell Out: the cell value
	get(name)	Get the column	In: *name*, the name of the column Out: table containing the column values
	mask(name, ii)	Mask cell	In: *name*, the name of the column; *ii*, the index of the cell to mask
	unmask(name, ii)	Unmask cell	In: *name*, the name of the column; *ii*, the index of the cell to unmask
	sort(name, order)	Sort column	In: *name*, the name of the column to sort; *order* = 1 if ascending order and 0 otherwise
Plot			
	title(stitle)	Set the graph title	In: Graph title
	frame(backr, backg, backb, plotr, plotg, plotb)	Set the graph colors	In: RGB window and graph colors components
	add(namex, namey, istyle, iaxis)	Add curve to the graph	In: *namex*, *namey*, name of the X and Y columns; *istyle* = 1 for line, 2 for scatter and 3 for both; *iaxis* = 1 for XY axis, 2 for XY2, 3 for X2Y and 4 for X2Y2
	remove(name)	Remove curve from the graph	In: *name*, name of the X or Y column
	axis(iaxis, iscale, iautoscale, fmin, fmax, ititle, stitle)	Set the axis properties	In: *iaxis* = 1 for X, 2 for Y, 3 for X2 and 4 for Y2; *iscale* = 1 for linear and 2 for log; *iautoscale* = 1 if AutoScale and 0 otherwise; *fmin*, *fmax* = axis min and max; *ititle* = 1 if title visible and 0 otherwise; *stitle* = axis title
	label(iaxis, ilabel)	Activate the axis label	In: *iaxis* = 1 for X, 2 for Y, 3 for X2 and 4 for Y2; ilabel = 1 to activate axis label and 0 otherwise.

In addition to the SigmaGraph classes (**Doc**, **Data** and **Plot**), you can use the **Physics** table containing the fundamental constants and some useful functions:

```
-- Constants
      Physics.k                -- Boltzmann constant
      Physics.q                -- electron charge
      Physics.h                -- Planck constant
      Physics.c                -- speed of light in vacuum
      Physics.e                -- vacuum permittivity
      Physics.n                -- Avogadro constant
      Physics.m                -- electron mass
```

-- Functions

```
Physics.Current(V, T, ISC, RP, IS1, n1, IS2, n2, RS, I1, I2)
```

Calculate the current versus voltage of a photovoltaic cell by solving the two-diode model nonlinear equation:

$$I = I_{ph} - I_{S1}\left(exp\left(\frac{V + R_S I}{n_1 V_T}\right) - 1\right) - I_{S1}\left(exp\left(\frac{V + R_S I}{n_2 V_T}\right) - 1\right) - \frac{V + R_S I}{R_P}$$

V: Applied voltage.

T: Temperature.

ISC: Short-circuit current.

RP: Parallel resistance.

RS: Series resistance.

IS1: Saturation current for diode 1.

n1: Ideality factor for diode 1.

IS2: Saturation current for diode 2.

n2: Ideality factor for diode 2.

```
Physics.Capacitance(Freq, r, L, GLF, CHF, C1, Tau1, C2, Tau2)
Physics.Conductance(Freq, r, L, GLF, CHF, C1, Tau1, C2, Tau2)
```

Calculate the admittance (capacitance and conductance/ω) of a PN or Schottky junction, with two trap levels:

$$\frac{G_m}{\omega} = G\frac{1 + rG + \frac{\omega^2 rC^2}{G}}{\omega(1 + rG - \omega^2 LC)^2 + \omega\left(\omega(rC + LG)\right)^2} \qquad C_m = C\frac{1 - \left(\frac{LG^2}{C} + \omega^2 LC\right)}{(1 + rG - \omega^2 LC)^2 + \left(\omega(rC + LG)\right)^2}$$

$$G(\omega) = G_{LF} + \sum_{k=1}^{N} C_k\left(\frac{\omega^2\tau_k}{1 + \omega^2\tau_k^2}\right) \qquad C(\omega) = C_{HF} + \sum_{k=1}^{N} C_k\left(\frac{1}{1 + \omega^2\tau_k^2}\right)$$

L: Equivalent series inductance.

r: Series resistance.

GLF is the low-frequency parallel conductance, CHF the high-frequency parallel capacitance, Ck the difference between the “low” (relatively to the transition frequency of the level) and “high” frequency capacitances for the level k, ant Tauk its time response.

NB: the function `Physics.Conductance` return the conductance divided by the angular frequency ω (= 2πf).

3.9.3 SCRIPT EXAMPLES

Script examples are available in the *Help/Examples* menu.

Example 1: calculate and plot a function, using the math library:

```
-- SigmaGraph script sample

NN = 100    -- Number of points

tD = Data:create()                                  -- Create a Data object
tD:new("SigmaGraph")                                -- Create a new datasheet
nc,nr = tD:dim()                                    -- Get the number of column and rows

if (nc < 3) then                                    -- Append one column, if necessary
    tD:appendcol()
end

if (nr < NN) then                                   -- Append rows, if necessary
    tD:appendrow(NN - nr)
end

tD:format("A", "%.3f")                              -- Set numeric format...
tD:format("B", "%.3f")                              -- ... for columns A, B and C
tD:format("C", "%.3f")
Xinit = 0                                           -- Initial value
Xstep = 0.05                                        -- Step value

x = {}
y1 = {}
y2 = {}
for ii=1,NN do
    x[ii] = Xstep * (ii - 1)
    y1 = 1 - exp(-x[ii])
    y2 = 1 - exp(-2 * x[ii])
end
tD:set("A", x)
tD:set("B", y1)
tD:set("C", y2)

plotT = Plot:create()                               -- Create a Plot object
plotT:add("A", "B", 1, 1)                           -- Add Y1(X) curve
plotT:add("A", "C", 1, 1)                           -- Add Y2(X) curve

-- Set X-axis properties
plotT:axis(1, 1, 1, 0, 0, 1, "Time (a.u.)")

-- Set Y-axis properties
plotT:axis(2, 1, 1, 0, 0, 1, "Charge (a.u.)")
plotT:color(1,255,0,0)                              -- Curve color (red)
plotT:color(2,0,0,255)                              -- Curve color (blue)
plotT:update()                                      -- Update the graph
```

Example 2: Curve fitting:

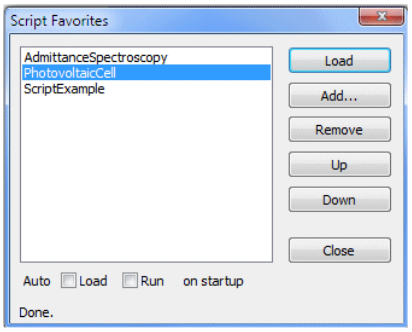
```
-- SigmaGraph script sample

tD = Data:create()                      -- Create a Data object
tD:new("Fitting")                       -- Create a new datasheet
nc, nr = tD:dim()                       -- Get the number of column and rows

if (nc < 3) then                        -- Append one column, if necessary
   tD:appendcol()
end

x = {}
y = {}
z = {}
a = 1
b = 2
for ii = 1, 10 do
   x[ii] = (ii - 1) * 0.1
   y[ii] = a + b * x[ii] + 0.3 * math.random()
end

pars, chi, iers, msg = data.fit("Linear", x, y, {0.1,0.1})

for ii = 1, 10 do
    z[ii] = pars[1] + pars[2] * x[ii]
end

tD:set("A", x)
tD:set("B", y)
tD:set("C", z)

plotT = Plot:create()                   -- Create a Plot object
plotT:add("A", "B", 2, 1)               -- Add data curve
plotT:add("A", "C", 1, 1)               -- Add fit curve
```

Example 3: calculate and plot the current-voltage characteristic curve of a photovoltaic cell:

```
-- SigmaGraph script sample
tic()                           -- Start timer.
cls()                           -- Clear the output window
NN = 100                        -- Number of points
tD = Data:create()
tD:new("Current")
nc,nr = tD:dim()
if (nc < 3) then
    tD:appendcol()
end
if (nr < NN) then
    tD:appendrow(NN - nr)
end
tD:format("A", "%.3f")
tD:format("B", "%.4e")
tD:format("C", "%.4e")
V = 0                       -- Voltage (V)
T = 300                     -- Temperature (K)
ISC = 1e-4                  -- Short-circuit current (A)
RP1 = 1e6                   -- Parallel resistance (Ohms)
RP2 = 1e4                   -- Parallel resistance (Ohms)
IS1 = 1e-11                 -- Ideality factor for diode 1
n1 = 1                      -- Ideality factor for diode 2
IS2 = 1e-9                  -- Saturation current for diode 2 (A)
n2 = 2                      -- Ideality factor for diode 2
RS1 = 10                    -- Series resistance (Ohms)
RS2 = 10                    -- Series resistance (Ohms)
I0 = 1                      -- Current max value (for the nonlinear solver)
Vinit = 0
Vfin = 0.5
DV = (Vfin - Vinit) / NN
V = {}
I1 = {}
I2 = {}
for ii=1,NN do V[ii] = DV * (ii - 1)
    I1[ii] = Physics.current(V[ii], T, ISC, RP1, IS1, n1, IS2, n2, RS1, I0)
    I2[ii] = Physics.current(V[ii], T, ISC, RP2, IS1, n1, IS2, n2, RS1, I0)
    if (I1[ii] >= 0) and (I2[ii] >= 0) then
           break
    end
end
tD:set("A", V)
tD:set("B", I1)
tD:set("C", I2)
tP = Plot:create()
tP:add("A", "B", 1)
tP:add("A", "C", 1)
tP:axis(1, 1, 1, 0, 0, 1, "Voltage (V)")
tP:axis(2, 1, 1, 0, 0, 1, "Current (A)")
tP:color(1,255,0,0)
tP:color(2,0,0,255)
tP:update()
```

```
print(toc())       -- Print the elapsed time (sec.) since tic was used
```

3.9.4 FAVORITES MANAGER



SigmaGraph allows you to organize very easily your scripts with the integrated favorites manager (select *'Favorites'* menu or click the corresponding toolbar button). You can (i) group all your most used scripts and load them very quickly; (ii) automatically run multiple scripts and open one script at SigmaGraph startup (very useful feature for specialized library or frequently used/modified scripts). With the favorites manager, you can add, remove, move up or down, load, auto-run or auto-load script by clicking the corresponding button.

4. Specifications

SYSTEM REQUIREMENTS

SigmaGraph runs on PC with Windows™ XP, Vista or Windows 7/8/10 installed.

The basic hardware requirements are:

- Pentium or better microprocessor.
- 256 MB RAM.
- 7 MB of hard disk space.
- VGA monitor with 800x600 or higher resolution.

CONTACT

http://www.hamady.org

sidi@hamady.org

COPYRIGHT

http://www.hamady.org

sidi@hamady.org

LICENSE