User Manual for Abscissa 3.4

Abscissa is an easy to use 2D plot program for Mac OS X

by

Rüdiger Brühl

October 2015

Introduction

Main Features

- ASCII data import and export
- fast reading and plotting
- plotting in many styles
- linear, logarithmic, and date axes
- text objects in the diagram
- x,y error bars
- 4 interpolation methods
- user entered functions f(x)
- flexible functions (parameters, user data, compiled C-code)
- parameter fitting using a Least Squares Method
- fit of function with several independent coordinates (R^n->R), more
- external programs can use Abscissa as plot front-end, more
- perl scripts can be recorded, played, and saved
- data editing in a spread sheet, more
- data sorting
- data statistic and correlation

New Features

- Scripting Examples: FFT, ...
- new auto-updating but still configurable legend
- new interface
- universal binary
- faster drawing of long polygons
- transparent bitmap export in desired resolution (mainly for MS-Powerpoint)
- mosaic plots by means of a script
- reverse axis

Author

Rüdiger Brühl
Physikalisch Technische Bundesanstalt
Abbestr. 2-12
10587 Berlin
Germany

e-mail: ruediger.bruehl@ptb.de

Thanks to Dirk Schwarzhans for his formula object.

Abscissa is Freeware.

Bugs

If you find a reproducible bug feel free to send me a report.

A Quick Tutorial to Plotting and Fitting

Let‘s assume you want to plot the following data.

```
x_data  y_data
2.25    11.5
2.5     9.7
2.75    9.0
3       7.8
3.25    7.1
3.5     6.2
4       5.8
4.25    5.4
4.5     5.7
4.75    5.9
5       6.4
5.25    6.7
5.5     7.3
5.75    7.8
6.25    10.6
6.5     11.5
```

In order to do so, copy these lines to your paste board, switch to the plot window and use the menu command "Edit->Paste". The curve looks like a parabola with some noise on it. We will try to fit the parameters of a parabola.

- First you have to tell Abscissa which curve it should fit to. Click on the curve and the Browser Panel will show up. Choose the "mark as fit data" command from the pop-up-menu.
- Open the Formula-Panel. Tool-Menu->Formula
- Type the following formula in the text field
  `a*(x-x0)^2+y0`
  and press the "set" button.
- The parameters are listed and should get start values. Fill the field with numbers: `a=1; x0=1; y0=1`
- Click on the "fit"-button and you get the correct values
- Click on the "show"-button to plot the fitted parabola
- To make it look better click on the original curve and disable the line switch in the Inspector‘s Basic Tab. Instead, open the Inspector‘s Symbol Tab and display the original data with a symbol of your choice.

File Format

There are 4 kinds of files that Abscissa can read and write.

- ASCII data files (any extension)
- Binary data files (**.8x8** extension)
- Binary layout files (**.abs** extension)
- Project files (**.absd** extension)

Data files can be compressed (**.Z** or **.gz** extension).

ASCII File Format

Data should be arranged in a table, in which the first column will be interpreted as **x-data**. The other columns are **y-data**. If there is only one column in the file, then Abscissa creates x-data with values 1, 2, ... The number of columns of a table is determined by the first data line. Columns must be separated by at least one space or tabulator. Optionally, the first line of a table can be a line of names of the following columns. Before this title line you can place any number of **comment** line which start with a **#-sign**. Abscissa will associate this comment to the x-data. It can later be viewed and edited. A file can contain more than one table. Tables must be separated by a blank line or by a name line. Instead of a value there can be a "NAN" (not a number) in table. This means that this data point will be ignored for plotting

and calculating. If you want to display all columns as y-data then create a x-data column with "**create**".

To import data from Mathematica use:

```
T = Table[{x,x^2},{x,0,10}];
TableForm[T, TableSpacing -> {0,1}]
```

Project File Format

A "Project File" is a directory with data files in it. These data files can be binary or ascii formated. Additional there are a "Layout"-file and a "FileList"-file that describes the order of these files.

Binary File Format

Binary files have the same structure as ASCII files. They are saved as "typedstreams" and can be read by all NextStep / OpenStep / MacOS X-computers. The specification is available from the author.

Layout Files

Layout files hold all information of the panels (Axis values; Curve settings; ...) but not the table data. Only the information of one main-window is stored in a layout file. It is better to read in first the data and then the layout file.

Save Layout as Default

The command "Save Layout as Default" saves the layout of the current plot to the file "~/Library/Preferences/AbscissaStartUp.abs" which is imported every time you start Abscissa. This file contains the default plot style and window positions.

Limitations

- 15 decimal digits value precision
- maximal 10240 characters of ascii input files line length
- only one Derivate() and Integrate() -function in a formula

The Axis Panel

After opening the axis panel first check wether the right axis selector on the top is highlighted. To turn an invisible axis on, press "**top x-Axis**" and unmark "**no line**" and "**no num**". The axis on the right side and on the top can either have individual "**min**" and "**max**"-values or can be connected to the main axis on the opposite side if min and max are set to 0. The switches "**right y-axis**" and "**top x-axis**" in the "**Inspector->curves**"-panel define to which axis a given curve belongs.

If the x-axis is selected the **"local range"** command of the "various menu" will reset the y-axis, in a way that all data point in the x-range become visible. The "aspect ratio" command will calculate a new axis-max-value to adapt the tick distance of both axes.

Additional axis related parameter like axis length, title spacing, or tick length can be found in the inspector->document tab.

Date Axis

Abscissa has the ability to plot the axis with date strings instead of numbers. Mark the "date" switch of the axis panel to use this feature. As in MS-Excel one data unit represents 1 day. The date representation of the data value zero is defined in the "AxisPanel->various->date format->origin". Below the date format is defined which depends only on the data bandwidth of the axis called "range". The distance to the next date label is defined by "increment". "ticks" ticks are used between labels.

To let the first label begin at a round date/value use the "AxisPanel->disloc" offset.

Abscissa has no ability to read a date string from a text file. However, MS-Excel can be used to convert a date string to a real value by means of cell format.

hints:

- to avoid the automatic setting of "inc", "prec."and "#ticks" disable inspector->document->auto range
- especially, when using extra long or short axes, the number spacing can be improved by means of the label density parameter. Larger values lead to more numbers per axis.

The Inspector

The **Inspector** panel is the main interface to change properties of curves, text objects, and the document layout.

Curve Properties in the Curve Tab

The browser offers all data available for plotting. The left column shows the origin or filename of the data. If present, the table index within a file is shown in the second browser column. The last column displays the names of all curves as they appear in the table. In front of the name the internal name is shown (e.g. `"Y(21,x)"`) which you will only use in formulas. All settings in the lower part of the **Inspector** panel are related to all selected curves.

However, you can make curve selections that can not be displayed by the browser. Let us assume you want to select the third curve of many tables. The browser can not visualize this selection. To overcome this problem first select the third curve in a single table. Then hold down the CONTROL key while selecting more tables. Now only the third curve of all selected tables will be selected. The only indication for doing it right thing is the panel title of the browser which tells you how many curves are selected.

On the selected curves you can perform the following commands which are accessible in the pull-down menu **"perform on selection above".**

delete

All selected curves will be deleted.

mark as fit data

Before fitting works use this command to choose the fit data curve.

sort by x-data

The lines of the table will be sorted so that the first column (x-values) increases.

sort by selection up/down

The lines of the table will be sorted so that the values of a selected column increase/decrease. The order of two lines that do not differ in the selected

column is preserved. So you can sort successively from least to most important criteria.

move forward / backward

This can change the order of selected curves in the Browser. The order seen in the browser is used for drawing the curves. Changing the order is especially important for filled curves.

swap

This can be used to change the x-column.

duplicate as plotted

Here the selected curves will be copied and placed into new tables. Instead of copying the source data the plotted data are used. This makes it possible to access smoothed or interpolated data. Open the **DataViewer** panel to use these data. If the axis is logarithmic the logarithmic values will be used.

create

With this command you can create data. Distinguish two cases:

- If a table is selected then a new table will be created that has one x-column and one y-column. Therefore you should provide the number of lines of the table and values for start value and increasing value.
- If a curve is selected then only one column will be added.

convolution

Convolutes the selected curve S(x) by the convolution function ("tmp") that is created by "show"-command in the "Function"-Panel. Example: A smoothing can be done by convoluting your data with the convolution function $f(x) = 2*sqrt(log(2)/pi) / w *G(x,0,1,w)$ (an area normalized Gaussian function). To do this, type this function in the "Function"-Panel, set the x-range to an interval where the convolution function is important [-3*w, 3*w], and press "show". Outside the x-range the convolution function is set to 0.0 . The numerical integration works with the linear interpolation of tmp(x) and S(x). So increase "**interpol**" in the Inspector panel if tmp(x) changes rapidly. The computation time of convoluting scales with numberOfPoints(S())*numberOfPoints(tmp()).

```
S_conv(x)=Integral(z=-Inf to +Inf)(tmp(z-x) * S(z) *dz)
```

correlation

The correlation of all selected columns to each other is printed in a matrix form. The values are always in the range -1...+1. Zero means no correlation and 1 or -1 means total correlation.

$$\text{corr}(x,y) = (\text{avg}(x_i*y_i) - \text{avg}(x_i)*\text{avg}(y_i))/$$
$$\text{sqrt}((\text{avg}(x_i^2)-\text{avg}(x_i)^2)*(\text{avg}(y_i^2)-\text{avg}(y_i)^2))$$

The curve properties are distributed over many tabs from **"basic"** to **"text"**. Most of the options are self explaining.

Basic Tab

line

This switch enables the drawing of a line between data points.

stepped

This draws a horizontal line for each data point. Useful for histograms.

y-error bar

If this switch is enabled the selected column will be used as half error bar length for the data points of the previous column. This means, there must be at least three columns in the table to plot error bars (x,y,error). Moreover, it will be used as date error for fitting. For fitting the Inspector->Document panel offers more data error settings.

Many properties of the error bar can be set by means of curve properties below i.e. color, "line thickness", "symbol size " for the cap size, and "every # point" to skip some error bars.

x-error bar

Same as y-error bar for the x-coordinate. x-errors are **not** used for fitting.

filled

The area below the curve will be filled with the color that is defined in the "fill color" field.

upper x-axis

Use this to connect the selected curve to the second x-axis which can be enabled in the Axis panel.

right y-axis

Same as "upper x-axis".

marker

See chapter Marker.

More Tab

dash on

The line which is activated by the "line" switch will be dashed with segment lengths that are defined by the "on/off" values.

break line

With "break line at NAN" you can tell Abscissa to treat a NAN data point as a separator of two curves. Otherwise, NAN data points are just ignored. Unexpected results can happen in combination of alternate NANs and real values with stepped plotting.

"break line on step back" let start a new curve in case a x value is smaller than it predecessor.

recover formula

Curves that have been created or modified by means of a formula remember this formula and its parameters. To copy this formula back to formula panel click this button.

Smooth Tab

average

"**average**" is a parameter for smoothing. It is the number of data points to be used to calculate the average point of x- and y-values. To visualize the width of averaging the "**stepped**" option is useful. "**average**"**=0** has a special meaning. Here "**average**" will be internally set to a value that will lead to about 800 plot points within the x-range.

skip

"**skip**" offers the ability to reduce the number of points to be plotted. When "skip" is active, plot points will be suppressed that lie on the line between their neighbor points. For instance a rectangular-function that is given by more than 4 points per cycle can be reduced to its 4 characteristic points. The benefit of this is not a reduction of plotting time but to get a smaller data set for usage in other programs. (see "duplicate as plotted" and "copy"). The "**dist**"-parameter defines the distance of the line to the point that should be excluded.

Interpolation

The choice of an interpolation method defines the form of the curve between two given data points. The parameter "**interpol**" in the "Inspector"-panel is the number of points to be plotted in the visible x-range. It is preset to 400 which could be too few for curves that change rapidly. If you choose "**none**" interpolation is inactive. "**linear**" means linear interpolation which is better done by drawing a line with the "**line**" switch. It is only useful for calculating data (see "modify"). "**spline**" is an interpolation with a power series of 3-rd order. It will sometimes swing over. "**akima**" is a method of the so-named author. "**FFT**" is a interpolation by Fourier-calculation. High frequencies will be damped. x-values are not used in this calculation. The lowest frequency that will be damped is defined by the "interpol." parameter. This wavelength is noted in numbers of data points.

stat tab

The tab shows some statistical values of the selected curve. If the x-values of the curve are sorted then the statistical values are based only on the part of data that is in the visible x-range or within the marked fit-range if present. The "**area**" entry is calculated by a set of trapezoids under the curve.

symbol tab

There are two kinds of symbols. In the first 5 by 3 symbols are made by lines and filled shapes. All other symbols are characters of the ZapfDingbats-Font. It is possible to plot several symbols on top of each others. The line- and fill-colors in the "basic"-tab are used for these symbols too.

symbol size

The size of symbols of the previous (above in the browser) curve is multiplied by a factor defined by the data in this curve. You can also change the symbol by adding 1000*N to the size factor to draw the N'th symbol.

Text Properties

The x and y coordinate define the position of the text object within the plot. In case of a selected floating check box a coordinate space is same as the data space. For an unchecked floating check box the coordinate range from 0 to 1 corresponds to the plot range. The anchor point of the text object is its center.

The shrink frame option ignores blank leading and trailing lines when it comes to draw the text frame. This is especially useful for automatically filled text objects (like the the legend) when some lines have undefined variables.

Document Properties

"**fiterr = sqrt(y)**", "**fiterr = y/10**" and "**fiterr = 1.0**" are usable for fitting if the fit-data curve has no y-error-bar. If "fiterr = y/10" is active the error of each point of the fit-data curve is set to 1/10 of the y-value of the point. So smaller the y-values the smaller its errors. This is most useful if the y-axis is logarithmic. "fiterr = sqrt(y)" is useful if the y-values come from a counting process and have a statistical error which is sqrt(N). "**fiterr = 1.0**" set the errors to 1.0. If none of these three options is enabled the error is set to **1/100** of the y-range (ymax-ymin of the "Axis"-panel).

"**multi Dim Fit**" (see "Multi Dimension Fit")

"**append read**" can be turned off to automatically delete all tables before paste in or read in data.

"**auto range**" can be turned off to avoid automatic x-y-range setting. (See also Axis panel)

The values in the fields "**space left**", "**space bottom**", ... define the distance of the 4 axes to the window frame. The values are noted in font size of axis

numbers. To add a title to the plot choose a "space top"-value of 3.0 and create a title with "Text->new". If the numbers of the second axis are not completely visible, increase the "space right"-value.

The Data Viewer

The DataViewer displays the values of the curves that are selected in the **Inspector** panel. If the x-data are sorted then only the part is displayed that is in the visible x-range. The values in the cells can be edited by double-clicking the cells.

Commands

- "**Edit->**" "**copy**", "**paste**", "**delete**" are useable on selected columns or rows
- "**Edit->find**" select all rows which values of the selected column are in the range "**min**" to "**max**". "**copy**" or "**delete**" can now be used on this selection.
- The value `NAN` (not a number) in a cell means that this data point will be ignored in the plot.
- A closed DataViewer panel will speed up selecting in the browser.

Text Objects

The "**Tools->Text->new**" menu command creates a text object that can be formatted in the usual way with "**Format->Font**" and "**Format->Text**". To move the text object click on the lower left corner and drag it to a new position. The **text** of the **abscissa** and **ordinate** are objects of the same kind but they are not resizable. To justify this axis text use the alignment functions in "Format->Text".

Since version 3.1 text objects can contain variables. After leaving the edit mode the text object is searched for variables that are replaced with their content. The following variables are currently implemented:

variable	example output	description
`${Xmax}`	`870`	
`${Xmin}`	`820`	
`${Ymax}`	`450`	
`${Ymin}`	`0`	
`${FitMax}`	`856.5942`	
`${FitMin}`	`831.956521`	
`${fitCurve}`	`1`	returns the curve index of the curve marked as fit data
`${targetY}`	`1`	returns the curve index of the selected curve
`${targetX}`	`0`	returns the curve index of the x-data of the selected curve
`${targetLength}`	`384`	returns the number of data points of the selected curve
`${targetTable}`	`0`	returns the table index of the selected curve
curve data access:		
`${Y, c=1, x=84.3}`	`1.2`	c defines the curve index. The output value is linear interpolated between the two neighbour points of x.
`${Yi, c=1, i=0}`	`1.2`	c defines the curve index and i the data point index.
All formula parameters like:		
`${a}`	`2.92758`	returns the current parameter value of "a" as shown in the formula panel

`${a, m=v+-err}`	`2.927 ± 0.024`	the value and its error are returned with a precision according to the error
`${a, m=err}`	`0.024`	error of "a"
`${a, m=corr}`	`2.2`	correlation of "a"
`${a, m=min}`	`-1e+30`	lower constrain of "a"
`${a, m=max}`	`1e+30`	upper constrain of "a"
`${a, m=min.max}`	`[-1e+30 .. 1e+30]`	

All numeric values can be printed with a supplied format:

`${a, f="%.1e"}`	`2.9e+00`	the format string is passed directly to the c-function "printf"

`${loginName}`	`bruehl`	
`${userName}`	`Rüdiger Brühl`	
`${projectPath}`	`/Users/bruehl/tmp/xx.absd`	
`${projectName}`	`xx`	
`${fitCurveName}`	`column title`	
`${fitCurveSource}`	`data file name`	
`${date f="%Y-%m-%d %H:%M:%S"}`	`2003-01-11 18:14:31`	returns the current date and time. The format is optional. It is passed to the foundation framework methode `descriptionWithCalendarFormat:`
`${projectDate f=...}`	`2002-10-29 20:21:01`	see `${date}`
`${formula}`	`a*x^b`	
`${symbol c=1 v=0 h=10 w=35 m=n}`		returns a little image of the symbol of a certain curve. The curve can either be specify by c=<curve index> or by v=<visibility index>. Latter is especially useful for a legend. For instance, v=2 will draw the symbol of third curve that has any graphical attribute. As anywhere in Abscissa indices start at zero. The size of the symbol's bounding box can be change with h=<height> and w=<width>. The mode m=n means that the symbol should be drawn without a line.

`${cName c=1 v=0}`	`column title`	returns the column name of a certain curve. The curve is specified as above.
`${sName c=1 v=0}`	`source name`	returns the file name which is shown in the left most browser column.
`${cxName v=0}`	`x-column title`	returns the name of the corresponding x-data.

All string values can be printed with an optional format:

`${cName v=0 f=1:4}`	`olum`	the format (f=<start>:<length>) specifies the character range to be used. Negative <start> count from the end.

Legend

Since version 3.2 the legend is made by means of text variables (`${symbol}` and `${cName}`) which keep the legend up to date automatically. However, the user can change the text but has to take care of being up to date. By means of the command "fix variables" the current outcome of the selected text object can be made permanent.

To add a legend to your plot use the Tools->Text->LegendText menu. With File+Date menu you will create a text object containing the file name and the current date which might be useful on a printout.

These two menu items are actually rtf-files which reside in the application bundle. The user can add rtf-files at two places:

- ~/Library/Application Support/Abscissa/TextFolder/
- /Library/Application Support/Abscissa/TextFolder/

The position and width of the created object is taken from the file name as in: LegendText x=0.8 y=0.85 w=180.rtf

The Formula Panel

In addition to the ability of Abscissa to plot given data it is possible to plot functions which the user defines by a formula. Therefore you need to open the "Formula" panel. Here you find a text-field for the formula and a list of parameters of the formula. Assuming you want to plot a sine-function then type `"g+amp*sin(freq*x + p)"` into the text field and click the **"set"** button. If the formula is not correct you get an error message and the character next to the error is selected. If the formula is correct it will be interpreted and its terms will match the following syntax:

- built-in functions like `sin, sqrt, …`
- built-in variables `x, i, targetY, targetX, …`
- data access function `Y(…), Yi(…)`
- constants `pi, e, …`
- operators `*, +, -, /, ^, …`
- sub-expression definition
- comments enclosed in `{…}`
- `addin0(…), …, addin9(…)` are functions which are linked at run-time. You can write your own functions in C! more
- other names are parameters which will be listed in the panel

In our example there are three parameters which you should give a value (i.e. g=0.5; amp=0.3; freq=20; p=0). The variable `x` will be varied in the plot range "xmin"..."xmax" in "**#show**" steps. When you press the "**show**" button the formula will be calculated and displayed. The calculated values (x, f(x)) are held in a new created table with the name "**tmp**". The calculated values are also shown in the "DataViewer" panel. This table "tmp" created by "show" has a temporary state because a new "show" (necessary after you change a parameter or "xmin" or "xmax") will override its data. To remove this temporary state press "**fix**" and the table becomes a regular permanent curve.

Optionally you can define variables by an expression. These variables can be used in further variable definitions as well as in the actual formula. Syntax :

```
name1:=expression1;
name2:=expression2;
…
abc := a*b*c;
x^abc
```

If you are interested in the function value at a certain x-value then you should use the "display/variables"-command in the "Formula Output"-Panel. After you enter a x-value Abscissa will display f(x) and all variables.

Hints:

- all letters of a formula will be treated as **case-insensitive** except for a function name with only one character
- to see the whole curve that you create with "show" use "**range over all**"
- you can specify a x-range where "show" will calculate the formula by setting a "fit-range"
- often you want to calculate the formula at x-values of an existing table. Therefore create a new column with inspectors "**create**" command and click the "**modify**" button.
- In the parameter list the parameters appear in the same order as they appear in the formula. To reorder the parameter list you can insert a comment at the beginning of the formula text like {first_para, second_para}.

Syntax and Functions

General Functions

`sin, cos, tan, cot, asin, acos, atan, acot, sinh, cosh, tanh, asinh, acosh, atanh, exp, ln, lg, sqrt, sq, abs, floor, rint, even, atan2(y,x), random` (`random` has no argument), `erf, erfc` (=1-erf)

Other Functions

`if``(condition, true_value, false_value)` returns the `true_value` if `condition` is true and the `false_value` if not.

- `if (1=2, 77, 22)` returns 22
- `if (1<2, 77, 22)` returns 77

`switch``(selector, case0, case1, case2, …)` returns `case0` if `selector`=0 and `case1` if `selector`=1 ...

- `switch (2, 11, 22, 33, 44)` returns 33

Derivate(f(x),x) = $(f(x_i)-f(x_{i-1})) / (x_i-x_{i-1})$ is a numerical derivative of the expression f(x) in x.

Integral(xmin, xmax, n1, n2, nInc, powerX, powerY) integrates curve n1 over its x-coordinate from xmin to xmax. Applying an optional n2 the integrals from curve n1 to n2 with increment nInc are summed. The two power arguments change the integrand from y to x^powerX * y^powerY. The defaults for the optional arguments n2, nInc, powerX, powerY are n1, 1, 0, 1 resp..

avgI(nStart, nEnd, nInc, iStart, iEnd, iInc, pI, pY, nY2, p2, n3, p3, n4, p4) is the sum of data values divided by its number. Included are the curves from nStart to nEnd with increment nInc and rows from iStart to iEnd with increment iInc. The defaults for the optional arguments nEnd, nInc, iStart, iEnd, iInc, pI, pY are nStart, 1, 0, inf, 1, 0, 1 resp. . In the case of both pI and pY being 0 the result is the number of summands N.

$$\frac{1}{N}\sum_{\substack{i=i_s,i_s+i_{inc},\cdots \\ j=n_s,n_s+n_{inc},\cdots}}^{\substack{n_E \\ i_E}} i^{p_i}\cdot Y(i,j)^{p_y}\cdot Y(i,n_2)^{p_2}\cdot Y(i,n_3)^{p_3}\cdot Y(i,n_4)^{p_4}$$

Integrate(f(x),x) = $\mathrm{Sum}(f(x_i)+f(x_{i-1}))\ *\ (x_i-x_{i-1})\ /2$ is a numerical Integration. Example: 1/sqrt(2*pi)*Int(exp(-0.5*x^2),x) is the Erf-function.

EigenValue(num,A11,A21,…,An1,A22,…A2n,…Ann) calculates Eigenvalues (JACOBI-Algorithms) of the real-symmetric-matrix A. Only the upper triangular part of A is to be specified. The eigenvalues are sorted from small to great and num=1 will return the smallest. It is possible to enter expressions for all Aij. The order n of A is recognized by the number of Aij's =(n/2*(n+1)).

PointsInRange(n, min, max) returns the number of data points of the curve n in the range min to max. The curve number n is displayed in the "Inspector"-Panel in the form Y(n,x).

Histogram(n, x, offset, width) is the same as **PointsInRange** but the range is one of:

(..., [-1/2*width/2+offset, 1/2*width+offset], [1/2*width+offset, 3/2*width +offset], ...)

in which x is inside.

Spline(at_x, x1,y1, x2,y2, x3,y3, …) returns the y-value at at_x of a spline function defined by **n** points (2<**n**<50).

Akima(at_x, x1,y1, x2,y2, x3,y3, …) see **Spline**

LinInterpol(at_x, x1,y1, x2,y2, …) see **Spline**

maxdata(n), **mindata**(n) return the largest / smallest value of the curve n

max(v1, v2, …), **min**() return the largest / smallest argument

fourierSin(icurve, xmin, xmax, freq), **fourierCos**() return the integral form xmin to xmax over x of the curve icurve multiplied by sin(2p freq x) and cos() resp.

Line Profile Functions

G(x,x0,a,w)=a*exp(-((x-x0)/w)^2 * 4 * ln(2)) => **G**(0,0,1,1)=1, **G**(0.5,0,1,1)=0.5, G() is the Gaussian-function

L(x,x0,a,w)=a/(1+4*((x-x0)/w)^2) is a Lorenz-function

Voigt(x,x0,a,wg,wl) is a Voigt-function which is approximately calculated. "wg" is the Gaussian-FWHM, "wl" the Lorenzian-FWHM. Ref.: Journal Quant. Spectrosc. Radiat. Transfer. Vol 21, 1979, pp. 309-313

Galatry(x,x0,a,wd,wc,z, eps) is a Galatry-function which is approximately calculated. "wd" is the Doppler-FWHM, "wc" the collisional-FWHM, "z" the standardized narrowing parameter, eps is an accuracy parameter (typ 1e-10). Ref.: Applied Optics Vol 23, No. 14, 1984, pp. 2376-2385 (A direct calculation of Galatry-function is used instead of using the routines represented in the reference)

Rautian(x,x0,a,wd,wc,z) is a Rautian-function which is approximately calculated. "wd" is the Doppler-FWHM, "wc" the collisional-FWHM, "z" the standardized narrowing parameter. Ref.: Applied Optics Vol 23, No. 14, 1984, pp. 2376-2385

HumlicekReal(x,y) is the real part of the complex probability function which is approximately calculated. "x" and "y" are the real and imaginary part of the corresponding complex argument of this function. Ref.: Journal Quant. Spectrosc. Radiat. Transfer. Vol 21, 1979, pp. 309-313

HumlicekImaginary(x,y) is the imaginary part of the complex probability function which is approximately calculated. "x" and "y" are the real and

imaginary part of the corresponding complex argument of this function. Ref.: Journal Quant. Spectrosc. Radiat. Transfer. Vol 21, 1979, pp. 309-313

The following operators are defined:

`+, *, -, /, ^, (), <, >, =, != , |, &`

`'|'` is the „or“ operator, **`'&'`** is the „and“ operator, the relation `a=b` will be analyzed by testing fabs(a-b) <= 2e-15*(fabs(a)+fabs(b)) to avoid rounding errors; '<=' is not implemented

The following constants are defined:

`e, pi, deg`=pi/180, `NAN`=not a number

Built-In Variables

`x` runs on the x-axis from xmin to xmax or on x-data of a curve.

`i` is the index of the data point and runs form 0 to N-1. (used to address curve data point `Yi(…,i)`)

`targetY` is the number of the curve which is the current target of the "modify" command (e.g. returns 13 for `Y(13,x)`). This is useful if you want to apply a function to several selected curves. (modify)

`targetX` is the number of the x-data column that belongs to the current **`targetY`** curve. You can also use the Yi(...)-function to access these x-data (e.g. `Yi(targetX, i)`).

`targetLength` is the number of data points of the curve to be modifed.

`targetTable` is the index of the table of the curve to be modifed.

`Xmin, Xmax, Ymin` and **`Ymax`** are the values of the Axis panel.

`FitMin` and **`FitMax`** have the value of the fit range if it is defined. Else they are `Xmin` and `Xmax` respectively.

`fitCurve` is the index of the fit-data column.

Curve Data Access Functions

You can get the y-value of a curve at an x-value by the function **`Y (curveNumber, x-value)`** e.g. **`Y(3,x)`**. If the x-value does not match an existing x-data-point the y-value is calculated by means of a linear interpolation. Therefore sorted x-values are necessary. If you apply it on unsorted data you get a runtime error message. To go on perform the "sort by x" command . To access or modify data of unsorted curves use the function **`Yi(curveNumber, index)`** e.g. **`Yi(3,i)`**

Run-time Loaded Functions

If the available set of functions is not sufficient to solve your problem you can write your own C-code and link at run-time. An example C code "functionsForAbscissa" can be found in the distribution folder. You will need XCode to compile the code. The compiler will install the bundle in "$(HOME)/Library/Application Support/Abscissa/". Unfortunately, Abscissa has to be restarted to access the recompiled bundle. To use the new functions open the formula panel and enter `"addIn0(1, -1.5, x)"` and press on show. This function simply adds the three arguments.

A more complex example is a gaussian filter function. First make sure that your plot document contains any kind of data to be filtered. Coming from the previous paragraph you only have a temporary curve called tmp in your plot. If this is the case press the fix button in the formula panel in order to have a permanent filter data source. Then enter the formula `"addIn1(curve, x, 1, width)"`. Apply values for the parameters i.e. `curve=1` and `width=0.1` and press on show. You will notice a new tmp curve looking smoother than to source curve.

Examples

The formula `"sqrt(Y(3, x+3.3))"` produces a curve that is the square-root of the leftward shifted curve 3.

If you want to add a value to each y-value of a curve then type:

```
Yi(1,i)+value
```

and use the **modify**-command

Parameter

Parameters are values with names in a formula. They are useful because they can appear several times in a formula and can be changed quickly. All parameters of the formula are listed with name and value. At the top of the parameter list is a selector switch which decides which parameter property are displayed.

- "**val**" value of the parameter
- "**err**" error of the value (will be set after the fit)
- "**coor**" correlation of the parameter (will be set after the fit)
- "**min**" lower limit (only used for fit)
- "**max**" upper limit (only used for fit)

In addition, the parameter can be locked and the fit procdure will not change its value.

Note: The parameter list can be printed in the "Formula Output" panel.

Modifying Data by Means of a Formula

modify y-data

You can use the "**modify**" button in the formula-panel to modify the selected curves y-data with the given formula. Therefore first type a formula like this one `"2*Y(targetY,x) - x/100"` and press **"set"**. The `"targetY"` in the formula stands for the number of the selected curve that should be modified. Now when you press the "modify"-button a dialog appears where you can choose what should be modified. To modify y-data press on "**y-data only**". You can select more than one curve or tables. If a table is selected its x-data will not be changed.

modify x-data

One or more tables should be selected. The formula `"2*x+1"` for instance will change the x-data in a matter that the curves will be stretched and move rightward on the x-axis. Therefore press "modify" and then "**x-data only**". This will only change the first column of each selected table.

Examples

The formula

```
Y(targetY,x) * LinInterpol(targetY,1,a1, 3,a3, 5,a3, …)
```

will scale each of the selected curves by specific factors.

The following function does the same with factors that are given by the curve data of curve 99.

```
Y(targetY,x) * Yi(99, (targetY-1)/2)
```

Parameter Fit

Abscissa has the ability to change the values of Parameters of a formula so that the function of the formula fits the given data as best as possible. Therefore a least-square-fit algorithm is used. The algorithm calculates the numerical derivation of each free parameter and changes the parameters in the direction of the gradient. This implies that the result depends on the starting values of these parameters. If the fit does not reach the minimum you are searching for, change the starting values. The aim of the fit is to minimize the sum of squares. It is defined by:

1/(2*N) Sum i=1..N of (((**y_i** - **f**(**x_i**)) / Δ**y_i**)^2)

- **N** number of data point in the fit range
- **x_i** x-values of the fit-data curve in the fit range i=1..N
- **y_i** y-values of the fit-data curve in the fit range i=1..N
- Δ**y_i** errors of yi (see "y-error bar")
- **f(..)** the formula

Usage:

First select one curve in the Browser and choose the "**mark as fit-data**" command. Then choose an x-range by <**alt**><**command**>-drag as fit range. This range will be marked by a gray frame. Now it is obvious that the x-data must be sorted. If not, sort the table. Then type the formula in the text field of the "Formula"-panel and click **"set"**. Provide values for all parameters and check your setting by displaying the formula curve by pressing "**show**". Some formulas like Gauss- and Lorentz- curves with parameters like height, width, position and ground can get their values from a mouse-selection . Now you can press on "**fit**" and the program will vary all parameters. If you want to exclude some parameters from optimization set them to be fixed . The "Formula"-panel shows the run: "**iter**" is the number of actual iteration (variation of parameter set). The limit for "**iter**" 60. Only in rare cases is it useful to restart the fit after an iteration-limit-error. "**1/2 y2**" is the above-defined sum of squares. It is minimal after a successful fit. "**lam**" is a step size parameter that runs in the range of 0.0 to 1.0. A successful fit should end with "**lam**"=1.0 (if not see). Now press "**show**" to verify the result. You can use the "**fit-Parameter**"-button in the "**Formula Output**" to save the parameter values. This save set can be reimported with the "**set Parameter**"-button. "**Marker**" can also be used to store parameters.

Details of Fitting

Fit Method

The method used here works like a gradient-method. The source is called "fumili" in the CERN-Library. It is necessary that the fit-formula be continuous around the minimum.

Step Size Parameter "lam"

This parameter expresses how well the fit-routine adapts the problem. "lam" is 1.0 if the formula result depends linearly on all parameters. This is mostly true at the minimum. "lam" will be 1.0 at the end of a good fit. In rare cases "lam" is 1 but the end criteria are not reached (a precision problem). Then "lam" falls rapidly. You can use the result with caution. But if "lam" never comes up to1.0 you should not use the fit result. The reason is mostly a correlation of parameters (corr>20.0).

Correlation of parameters

If after a fit the correlation value of a parameter is greater than ~20.0 then there is another parameter that has a similar or same effect in the formula for the fit range. Such parameters can not be fit. Sometimes a better formula or fixing some parameters is a solution. Formulas that are built by a good theory mostly have no correlation. Power series with many parameters are difficultly to fit, because they are mostly correlated and have precision problems.

An example for a good (without parameter correlation) and a bad (strong parameter correlation) formula:

Data:

x	y
1000	1.1
1001	1.9
1002	3.1

inappropriate formula: a*x+b -> corr(a) = 1.5e6

appropriate formula:a*(x-1000)+b -> corr(a) = 2.5

Multi Dimension Fit

The normal curve fit works with formulas that have only **one** independent variable ("x"). A "Multi Dimension Fit" can optimize parameters of formulas that have **several** independent variables ("Yi(2,i)", "Yi(3,i)", ...). The following expression will be minimized by the fit:

1/(2*N) Sum i=1..N (((**f**(**Yi(2,i)**, **Yi(3,i)**,...)-$\mathbf{x_i}$) / **Yi(1,i)**)^2)

- **N** number of lines of the fit-table
- $\mathbf{x_i}$ fit value that f() should reach (first column of the table)
- **Yi(1,i)** error of xi (second column of the table)
- **Yi(2,i)** first coordinate of xi (third column of the table)
- **Yi(3,i)** second coordinate
- ...

It is confusing that "x" and "Y" have the opposite meaning than in normal fit. This implies, that the plot of such data has no meaning and can be ignored.

Usage:

- read in data ("append Read" deactive is helpful for often reload)
- the fit-table should be first table in the "Browser"
- use "mark as fit data" on this table
- activate "**multi dim Fit**" in the Inspector->Document panel
- type your formula and set starting values of its parameters
- press "**fit**"
- if the fit has finished well press "**diff**" and open the "DataViewer" to check the result

A hypothetical problem:

How long does it take a coffee maker to make coffee?

The result of a measurement:

Time[sec]	error[sec]	water[liter]	coffee[gram]
351	10	0.5	1
522	12	0.7	3
546	12	0.7	10
479	20	0.6	12

....

Question:

Is the formula `"Yi(2,i)/power + Yi(2,i)*Yi(3,i)*cw"` (`Yi(2,i)`=water; `Yi(3,i)`=coffee) sufficient to calculate the result within its error and what are the values of the parameters `power` (heating power of the machine) and `cw` (water conductivity of the coffee ground)?

Result: (see "Formula Output" panel)

```
power= 0.00140942381071 +- 3.30476e-05
cw= 7.37316313193 +- 2.366044
chi= 0.158085 iter=46 lam=1
```

Answer:

The formula is good enough because chi is less then 1.0.

Mouse Modes in the Main Window

You can change the mode of the mouse by either selecting a tool in tool panel (Menu: Tools->Tool Box) or by holding down a modifier key. In the order of the tool panel the mouse modes are:

- Zoom In: drag the mouse and a rectangle of the new range is displayed
- Zoom Out: like "Zoom In" but with <**alternate**> key down
- extend selection in inspector window: click with <**shift**> down
- Fit-Range: drag mouse with <**alternate**><**command**> down
- Fit-Range remove: click with <**alternate**><**command**> down
- Set Parameter: drag mouse with <**command**> down
- show x-y-values of the nearest data point: click with <**control**> down
- write x-y-values of the nearest data point in "Formula-Output"-panel: click with <**control**><**shift**> down
- write x-y-values in "Formula-Output"-panel: click with <**control**><**shift**><**alternate**> down
- delete a curve point: click with <**control**><**command**> down
- clicks with <**shift**><**command**> or <**shift**><**alternate**> down are passed to external programs. see xxx

Mouse Modes in the Inspector Panel

In the Inspector Window the <**shift**> and <**alternate**> key act like in a regular browser. The selection will be extended. In addition to these keys the <**control**> key has a special meaning. It remembers the subselection and applies it to the additional selected table or source folder in the browser. This is useful to select a specific set of columns in many tables and easily change their curve parameters all at once.

A double click hides all not selected curves for about one second. To extend this period hold down the <**command**> key while double clicking.

Markers

With the switch "**marker**" in the "Inspector" panel you can choose whether the selected tables should be displayed as curves or as marker. A marker is a set of positions at the x-axis that have a set of properties (only values). In a marker table the first column holds the x-data and the other columns are the properties of the x-position. The position will be displayed by a short vertical line. This line is selectable and the properties are displayed in the "marker"-inspector. An example for a marker is the set of maxima of a spectrum. A new marker comes up at half height of the window and can then be moved by dragging its handle.

Inspector

The values (properties) in the inspector are editable (don't forget <enter>)

- "**clear**" makes all values of one position to 0.0
- "**delete**" deletes the selected position
- "**insert**" inserts one (or more) position and sets the properties with values of the parameter list in the "**Formula**" panel. Therefore the names of the marker-table columns are compared with the names of the parameters. Parameter names can have an additional digit that is not relevant for comparing. An example: If the parameter list is (`x0, x1, x2, a0, a1, a2`) and the marker names are (`x, a`) then there three positions will be inserted.
- "**to funct**" copies values in opposite direction than "**insert**"
- "**fit all**" starts a loop for all marker position behind the selected. The loop consists of: 1. "goto", 2. "to funct", 3. "fit", 4. "Insert". Here all marker properties will be interpreted as fit-parameters that should be optimized. The loop can be stopped with <command>-. . This is useful in combination with "Maxima".
- "**goto**" centers the x-range so that the selected marker position is in the middle of the window.
- A marker can be saved to disk with "**save**".

A new marker can be created with the "**create**"-command in the "Inspector" panel.

Maxima

If you press "**search**" a new marker will be created which consists of positions of maxima/minima of the selected curve. The entries "**height**" and "**width**" are used to suppress small and thin maxima/minima. The "width" value is noted in

numbers of data points (x-data is not used). The algorithm is fast but gives values for width and height that are not exact (see "fit all").

Control Abscissa from a Remote Program

Abscissa has an interface to be controlled by a remote program via PDO messages. The interface is defined in "AbscissaControl.h". All messages have a parameter "document" which is the title of the receiving document. Most messages are one way without return values and the sender will not be blocked until execution.

Two simple unix programs ("welcome" and "running_sine") show how to use the interface. Welcome features the following:

- start Abscissa if not already running
- open a new blank document
- set x and y range
- add same curve data
- add a text object

Running_sine demonstrates Abscissa‘s capability as a data acquisition front-end. It features the following:

- start Abscissa if not already running
- open a new blank document
- adding attributed axis titles (font size 22, greek letters, color)
- adding curve data consecutively in an incremental manner using very little CPU time
- changing curve color
- capturing mouse clicks with either <SHIFT>+<ALT> or <SHIFT> +<COMMAND> held down

You can stop all connections if you disable the "**enable connections**" preference. If the "**private connections only**" switch is enabled, remote programs can only access documents that were created via connection and save commands are forbidden. This is important because everyone on the internet can connect to Abscissa.

Recipes, the Script Language

This feature records a sequence of user actions to repeat them at a later time. The resulting script can be edited to add loops etc. The underlying interpreter is **perl**. All of its commands can be used. Perl opens a pipe to the program `"callAbs"` which transmits or retrieves data and commands to and from Abscissa. This pipe is called `TO_ABS` and `FROM_ABS` for writing and reading, respectively. For instance the perl command to let the x-axis start at 10 and end at 20 is:

```
print TO_ABS "-x setXRange -s 10 -e 20 \n";
```

To get the current axis range the perl script looks like:

```
print TO_ABS "-x getXRange \n";  # send the command
$text = <FROM_ABS>;              # read its output
chomp $text;                     # remove the \n
($dummy,$xmin,$dummy,$xmax) = split(' ', $text, 4);
                                 # parse the output
```

Most of these commands you don't need to learn since scripts can be build by recording the user action. However, some commands can't be recorded because they don't have a corresponding user action.

AppleScript is not support by Abscissa directly but since it can call `"callAbs"` all scripting can be done with Applescript as well. An example:

```
do shell script "/usr/local/bin/callAbs -x setAttributes2 -w 3"
```

It might be necessary to copy `callAbs` from within Abscissa.app to an appropriate place like `/usr/local/bin`.

Some Explanatory and Useful Scripts

Fast Fourier Transformation

The script `fft` will perform a fourier transformation on the selected data series and create a new plot window which displays the result. It utilizes the Unix command `"uniFFT"` which resides inside the application bundle Absissa.app. The available options will be listed by calling `"uniFFT -h"`. This little program is based on the great library from http://fftw.org .

Colorize Selected Curves

This script will give each selected curve a different color. To try it out copy the sample data below into a plot window. Goto *Tools->Recipe* and choose the script „*color sel. curves*“. After selecting the some curves in the inspector, start the script by hitting the *Play* button. Now the curves have colors with different hue. The script consists of the following parts.

- store the selection flag with the „ `markSelectedCurves`“ command
- run a loop over all just marked curves
- determine whether the curve (or better data column) is x or y data
- if its y data, select the data column and apply a calculated color

```
x2    d21   d22   d23   d24
11    12    13    14    15
17    18    19    16    17
```

Curve from Mouse Clicks

This script creates a polygon with corner points at each mouse click. To see it in action, open a new document window and run the script„ curveFromClicks“ in the *Tools->Recipe* Panel. Make sure that the documnent window is active and press the mouse button at different positions while <shift> and <cmd> is held down. The script consists of the following parts.

- create data if less than two columns of data already exist.
- run a loop for each data point doing the following
- wait 0.25 sec
- ask Abscissa for the next mouse click with <shift>+<cmd> or <shift> +<alt>
- if the answer is not „none“ do the following
- extract all information from the answer which are: x-y-corrdinates of the mouse down and mouse up event and wether <alt> or <cmd> was held
- use the x and y value of the down event to set the data of the curve
- redraw the window

Statistics of many Columns

This script simply asks Abscissa for statistical information of the first two data columns and prints the average.

Shift Certain Curves by different Amount

This script will move curves along the x axis. For each line that it reads form standard input, which is the light yellow field below the script display, the curve with index taken from the first value of the line is moved. The second value of the line determines the amount of the movement. The script consists of the following parts.

- install the formula „`x+move`“
- run a loop over all input lines doing the following
- set the variables `i` and `move` from the input line
- select the target x data column
- apply a new value to the `move` parameter of the formula
- run the formula acting on x data

Watch for Changes of a File and Plot It

This script tries to keep a plot up to date with a file as it changes from time to time. To see it in action simply create a file (in this case „`/tmp/xx`“) with TextEdit.app and save the file with different data while the script is running. The script consists of the following parts.

- run an infinite loop doing the following
- wait 1 second
- if the file in question exists do the following
- select and delete all data columns associated to the file
- open the file and redraw the plot with an appropriate range
- let the perl script wait till Abscissa has finished the above
- delete the file on the disk
- select the new data and apply a red color

Mosaic Plot

You can place several plots in a single PDF document by means of the script command `addPDFtoComposition`. This command accepts five optional parameters namely `-X, -Y, -w, -h,` and `-a` which define the position, size, and angle respectively. The units are cm and degree. Before plots can be placed on a page, the PDF document has to be initialized with `newComposition`. With to optional parameters `-X, -Y, -w,` and `-h` a document rectangle can be defined which is used by the subsequent `addPDFtoComposition`. Usually, the final PDF document will be saved on disk with `savePDFofComposition`. Here, the `-f` parameter defines the file name and the optional `-p` parameter lets Abscissa open the PDF document with the specified program. For example

```
print TO_ABS "-x savePDFofComposition -f /tmp/xx.pdf -a
Intaglio \n";
```

will let you see and edit the new composition. Without the a parameter the program Preview is used. Suppling "none" as program name won't not use any program.

Since almost all properties of a plot can be changed by scripting, most wishes for a composition of diagrams can be fulfilled. As an example the document "mosaicDemo.absd" produces a 3 by 3 mosaic plot of the same data with different plot ranges. The 3 plots on the lower row and on the left column feature numbers on the abscissa and ordinate respectively. To see the script in action, open the "mosaicDemo.absd" document, open the Tools->Recipe panel, choose Scripts->mosaic, and press on Play.

Implemented script commands

The following command are meant to be used in perl's print command like:

```
print TO_ABS "-x COMMAND \n";
```

Where `COMMAND` is one item of the long listing below:

The type of the argument is color coded:

- numbers in red like `123` or `1.25`
- booleans in green which can either be `Y` or `N`
- tristates in yellow can either be `Y` , `N` , or `K`
- strings in blue like `blabla`

- `getXRange`
  output e.g.: "xmin= 1.23 xmax= 2.34 \n"
- `getYRange`
  output e.g.: "ymin= 1.23 ymax= 2.34 \n"
- `getX2Range`
  output e.g.: "xmin= 1.23 xmax= 2.34 \n"
- `getY2Range`
  output e.g.: "ymin= 1.23 ymax= 2.34 \n"
- `selectCurveByIndex -i indexFirst [-j indexLast -a appendToSelection ]`
  no output
- `selectCurve [-s "sourceName" -t table -c "columnTitle" -a appendSelection ]`
  no output
- `indexOfCurveName [-s "sourceName" -t table -c "column"]`
  output e.g.: "0\n"
- `display [-r doRangeOverAll]`
  no output
- `getFormulaParameters`
  note: output is a single line
- `getVar -s "variableName"`
  note: see text object for valid variable names
- `getFormula`
  note: output is a single line
- `inventory [-s onlyInSelSource -t onlyInSelTable]`
  output: number of columns, tables, and sources
- `activeDoc`
  output: name of the active document
- `activate -d "docname"`
  no output
- `getFitRange`
  output e.g.: "xmin= 1.23 xmax= 2.34\n"

- `YatX -a x_value -s curve_index`
  output e.g.: "2.34\n"
- `Yi -i line_index -s curve_index`
  output e.g.: "2.34\n"
- `iAtX -a x -s curve_index`
  output: line index of the closest data point of given curve
  output e.g.: "0\n"
- `iAtXY -a x -b y`
  output: curve index, line index, x and y of closest data point
  output e.g.: "1 0 1.58 0.57\n"
- `setYi -i line_index -s curve_index -a value`
  no output
- `selectedIndex`
  output e.g.: "0\n", returns -1 for no selection
- `fitCurveIndex`
  output e.g.: "1\n", returns -1 for no selection
- `mouseClick`
  note: Mouse clicks and drags with the shift and command (or shift and alt) key held down are dedicated to script programming. `mouseClick` asks for the oldest event of those.
  output is: x1 y1 x2 y2 keyMode
- `replaceData -p "data" [-s "source" -r doRangeOverAll]`
  no output
- `readData -p "data" [-s "source" -r doRangeOverAll -a appendToExistingData]`
  example: callabs -x readData -p "0 0 \n 2 5"
  no output

Commands that are known by the recorder:

- `selectCurveByPosition -s sourceIndexFirst [-e sourceIndexLast] -t tableIndexFirst [-u tableIndexEnd] -c columnIndexFirst [-f columnIndexLast -a appendSelection]`
  note:-1 as index means all
  no output
- `searchMaxima -w minWidth -h minHeight -g serachMaxima -s searchMinima`
  no output

- `createCurveAtSelection [-s xStart -e xEnd -i xInc] -y yStart -j yInc`

  no output

- `nextMarkedCurve`

  output is: "index [x|y] sourceIndex tableIndex columnIndex columnName sourceName"

  e.g.: "0 x 1 0 1 col0 file1\n"
  returns: "-1 x\n"at the end

- `markSelectedCurves`

  returns number of marked columns, e.g.: "2\n"
  note: to be called once before sequence of `nextMarkedCurve`

- `modifyX`

  no output

- `modifyY`

  no output

- `fit`

  no output

- `delete`

  no output

- `show`

  no output

- `sortByX`

  no output

- `swapWithX`

  no output

- `sortBySelection [-r reverse]`

  no output

- `markSelectionAsFitData`

  no output

- `newDocument -d "docname"`

  no output

- `open -f "fileName" [-r doRangeOverAll]`

  no output

- `saveDocument -f "fileName" [-b binary -k keepOrigialDocName]`
  output is either: "save succeeded\n" or "save ERROR\n"
- `savePDF -f "fileName"`
  output is either: "save succeeded\n" or "save ERROR\n"
- `saveEPS -f "fileName"`
  output is either: "save succeeded\n" or "save ERROR\n"
- `saveTIFF -f "fileName"`
  output is either: "save succeeded\n" or "save ERROR\n"
- `savePNG -f "fileName"`
  output is either: "save succeeded\n" or "save ERROR\n"
- `copyPDF`
  no output
- `newComposition [-X x0 -Y y0 -w width -h height]`
  no output
- `addPDFtoComposition -X xCenter -Y yCenter [-w width -h height -a angle]`
  no output
- `savePDFofComposition -f "fileName.pdf" [-a "viewerProgram"]`
  no output
- `setFormulaParameters -p "parameter list"`
  parameter list e.g.:`"a=2.2 fixed b=3.3"`
  no output
- `setXRange [-s xStart -e xEnd]`
  no output
- `setYRange [-s xStart -e xEnd]`
  no output
- `setX2Range [-s xStart -e xEnd]`
  no output
- `setY2Range [-s xStart -e xEnd]`
  no output
- `setAttributes [-l line -s stepped -y yerr -e xerr -a dash -f fill -t topX -r rightY -m marker -S`

symboleSize -N breakAtNAN -b breakAtStepBack]

no output

- setAttributes2 [-s sum -p points -t interpolationType -y symbolSize -c symbolInc -m symbol1 -n symbol2 -w lineThickness -h dashLength1 -i dashLength2 -j dashLength3 -k dashLength4 -r lineColorRed -g lineColorGreen -b lineColorBlue -o lineColorOpacity -R fillColorRed -G fillColorGreen -B fillColorBlue -O fillColorOpacity]

  no output

- setDocAttributes [-w xAxisLength -h yAxisLength -t tickLength -a axisThickness -e yAxisPos -y xAxisPos -m multiDimFit]

  no output

- setProgAttributes [-a antialias -c regardTAB -t transparentBitmap -i dpi]

  no output

- setAxis [-a axisCode -i inc -t ticks -h prec -l log -u insideticks -w outsideticks -m majorgrid -n minorgrid -e expnumbers -r labelAngle -o noline -j numbersinside -k nonumbers -q arrow -b logticksAt -c lognumberAt -s reverse -g labelDensity]

  no output

- changeNameOfSelection -t "newName"

  no output

- placeText -t "text" -r xCoordinate -u yCoordinate [-a angle -f floatingX -g floatingY]

  no output

  note: if the "floating" parameter is set the axis coordinate system is used, if not the coordinate should run within 0 to 1.

- setFormula -f "text"

  no output

- copy [-i includeX]

  no output

- duplicateAsPlotted

  no output

- close [-a askForSave]

  no output

- `diff`
  no output
- `pasteData`
  no output
- `setFitRange [-s xStart -e xEnd]`
  no output
- `statistic`
  output is "points min max sum std area name\n"
- `print`
  no output
- `setDecimalSeparator -s 0_1_2_for_auto_dot_comma`
  no output
- `sync`
  wait until previous commands finished
  output: `"synced\n"`

Usually commands apply to the active document. However you can apply them to a specific document by means of the `-d` option like in:

```
print TO_ABS "-x getXRange -d MyDoc \n";
```