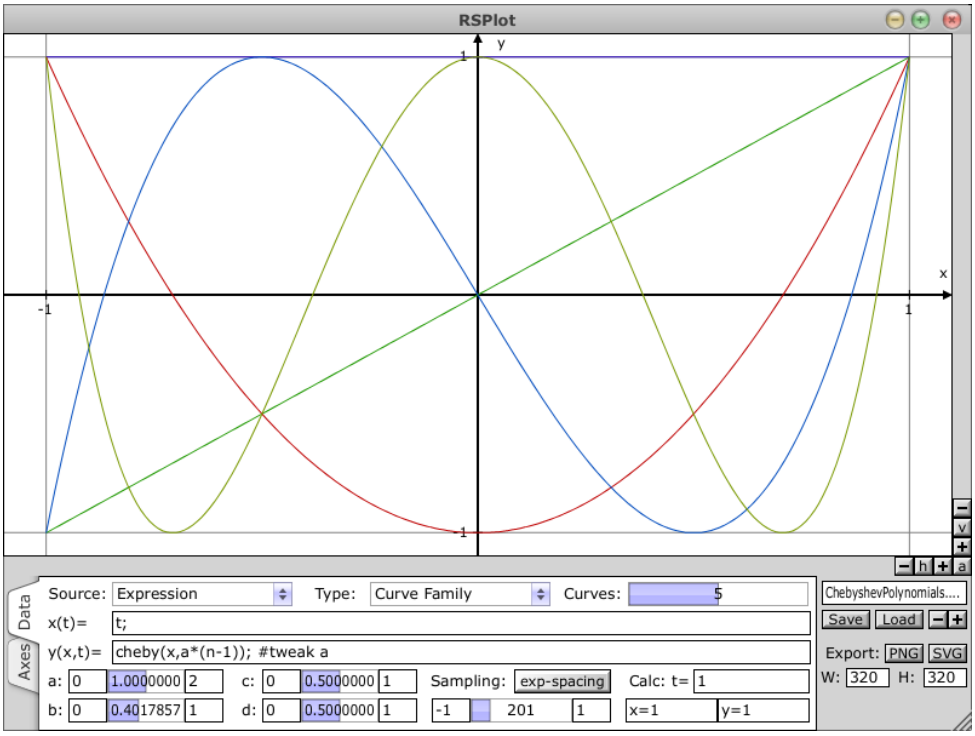
RSPlot - User Manual



What is RSPlot?

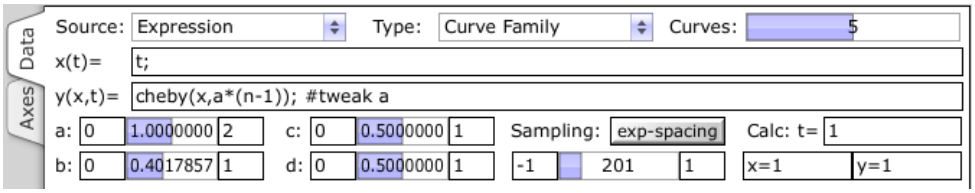
RSPlot is a mathematical curve and function plotting application. It supports plotting of single curves or functions, parametrized curve/function families and plotting of several unrelated curves/functions in one single plot. The result can be exported to an .svg vector graphics file or to a .png bitmap graphics file.

The File Loading/Saving Section

In the bottom right corner of the window is the section for saving and loading the current settings and exporting the plot to an image/graphics file. The 'Save' and 'Load' buttons are for saving and loading of the program settings (save works actually as 'save as') and the +,- buttons are for quickly skipping through the setting-files inside the currently chosen directory. The 'SVG' and 'PNG' buttons open a dialog window to export the resulting plot to a .png or .svg file respectively. The 'W', 'H' fields below select the width and height in pixels of the exported file. Both, file format and pixel resolution can be changed later in the dialog as well.

The 'Data' Tab

Left to the file section there is a tabbed field, with the first tab being the setup section for the data which is to be plotted. The data consists of samples of a curve or function family which can be either generated from a mathematical expression by the built in expression evaluator or alternatively be imported from a



.dat file (not yet implemented). The 'Source' menu chooses between those two different data sources. The 'Type' menu lets you select, what kind of data it is - this will affect the interpretation of the data and the way in which it will be presented - at the moment only the 'Curve Family' option is implemented. Right to this there is the 'Curves' slider - this lets you select, how many curves or functions will be plotted. Below these three widgets, you find the formula entry fields for the x-value and the y-value. It is assumed, that you want to plot a curve which is parametrized by the parameter 't' (you may think of it as representing time), so use the character 't' in the x-entry field to represent this parameter. The field itself describes, how x changes with respect to t in form of a mathematical expression. If you want to plot a function $y = f(x)$ (as opposed to the more general case of a curve), simply use the identity function here - that is: $x(t) = t$. In the field below, you enter the formula for y in a similar way - but here you can use the variable x as well. Think of it that way: first, the program calculates the x-values according to the x-formula. After the x-values have been assigned, it calculates the corresponding y-values. In the formulas, you can also use the constant parameters a, b, c, d, the numerical value of which can be assigned by the 4 sliders in the bottom left part of the 'Data'-tab. The number entry fields to both sides of those sliders select between which minimum and maximum value the slider can be adjusted. Next to these 4 sliders, you find the 'Sampling' section. Here you can define the minimum and maximum values for the t-parameter and how may samples should be taken in between this interval. These settings will affect, which part of the (potentially infinitely long) curve will actually be plotted (at which concrete values of t the expressions will be evaluated) and how precise the plot is. For a function (family), it often makes most sense to choose the minimum value for t equal to the minimum visible x-value, and likewise for the maximum. The exp-spacing button chooses an exponential spacing between successive t-values. This makes most sense for functions which should be drawn on a logarithmically scaled x-axis. The 'Calc' section lets you enter one single particular value for t and the corresponding result-fields will show you, what the corresponding x- and y-values would be. This function allows this plotting-application to serve as a calculator-application as well.

Intermediate Variables

Inside an expression for x or y you may have several subexpressions, each of which must end with a semicolon ';'. Via these subexpressions, you can create intermediate variables - the result of the whole expression will then be the result of the last subexpression. For example, the expression '$z = 3 * x; y = tanh(z);$' is the same as '$y = tanh(3{*}x);$'. The final '$y =$' is not strictly required, you could have just written '$tanh(z);$' instead of '$y = tanh(z);$' as well. That is to say: the last subexpression does not necessarily need a left hand side of the equation. When you use this feature, avoid using names for your intermediate variables which collide with the internal names, that is: don't use $t, x, x1, x2, \ldots y, y1, y2, \ldots, a, b, c, d, n$ for intermediate results. The significance of $x1, x2, \ldots y1, y2, \ldots$ and n will be described below.

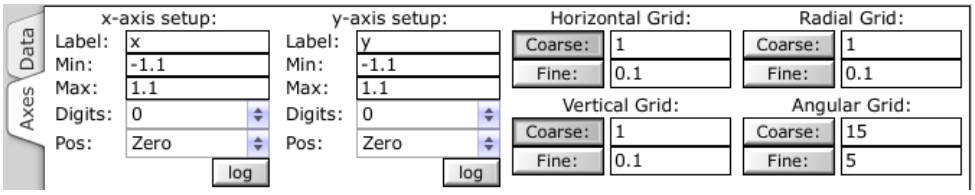
Curve Families

RSPlot allows for plotting several curves into one single plot. These curves may or may not be related with one another. For a (parameterized) curve or function family, you may use the character n inside the equations which is always assigned to the index of the respective curve which is currently drawn. The minimum value for n is always 1 and the maximum value is given by the number of curves which should be drawn (as selected via the 'Curves' slider). For example, the expressions $x(t) = t$; and $y(x, t) = cheby(x, n - 1)$; will draw all Chebyshev polynomials from order 0 to $N - 1$ where N is the number of curves. You may use subexpressions as decsribed above to paramterize the curves by different values than the curve index, for example, you could have been written: $k = n - 1; cheby(x, k)$; in the y-expression field without changing the result. And of course you can use more complex expressions to derive the curve-parameter k from the curve-index.

Several Curves

For several unrelated curves, you need to enter separate expressions for each of the curves in both formula entry fields. The different expressions in the x-formula entry field must follow the convention '$x1 = \ldots$;§ $x2 = \ldots$;§ $x3 = \ldots$; where the '§' indicates the separation between two expressions for $x(t)$. Likewise for the y-expressions. If you want to plot several functions (as opposed to curves), you do not enter $x1 = t$; § $x2 = t$; etc into the x-field - a simple t; will suffice here. The reason is, that the resulting data of the last expression will be used for all subsequent data-arrays as well.

The 'Axes' Tab



The second tab in the bottom of the window lets you set up the the scaling, range, position and annotation of the x and y axes. The 'Label' fields set up the annotation for the axes, when you want something different than x and y to appear there. 'Min' and 'Max' are used to select the visible range. 'Digits' selects how many decimal digits should be shown for the axis number annotation (not yet implemented). The numbers on the axis will appear at intervals determined by the coarse grid intervals which will be described below. 'Pos' lets you select where the axes should appear and finally the 'Log' buttons enable logarithmic scaling of the respective axis. The horizontal and vertical grid buttons switch on/off the drawing of coarse and fine grids, the corresponding number entry fields next to the respective buttons determine the spacing of the grid lines. You can also activate radial and angular grids in a similar manner.

Logarithmic Plots

If you want to have a logarithmic scaling for one or both axes, you should make sure that the 'Min' value for that axis is strictly positive because logarithms make only sense for strictly positive numbers (at least when we constrain ourselves to the real numbers, which this application assumes). And - obviously - when 'Min'

must greater than zero, so must be 'Max' because 'Max' always must be greater than 'Min'. The spacing of the grid-lines will be exponentially in case of a logarithmic scaling - that is: the (x or y) coordinate of one grid-line will be determined by a multiplication of the coordinate of the previous grid-line and some constant factor. Let's illustrate that by means of an example: let's say you want a logarithmically scaled x-axis with a minimum value of $x = 1$. Then - if you choose 2 as the coarse grid interval, the coarse grid lines will appear at $x = 1, x = 2, x = 4, x = 8, \ldots$. But despite their exponentially increasing spacing, they will appear at equal distances in the plot because the axis is logarithmically scaled. This also implies that for logarithmic plots, we will need to have the grid spacing intervals (both, coarse and fine) to be strictly greater than one to make sense.

Appendix: Function Reference

Arithmetic Operators

- +, -, *, /, ^: addition, subtraction, multiplication, division and raise-to-power operation (works only for integer powers)

Elementary Transcendental Functions

- exp(x): exponential function (raises Euler's number $e = 2.71828...$ to the power of x)
- pow(x,y): power function: x raised to the power of y
- sqrt(x): square root
- ln(x), log(x), logn(x,n): natural, base 10, and base n logarithm
- sin(x), cos(x), tan(x): trigonometric functions of x (expressed in radians)
- asin(x), acos(x), atan(x): inverse trigonometric functions, output is in radians
- sinh(x), cosh(x), tanh(x): hyperbolic functions

Special Mathematical Functions:

- cheby(x,n): Chebyshev polynomial of order n of the input argument x
- gauss(x,m,v): Gaussian distribution function with mean m and variance v (both parameters optional with default values of m=0 and v=1 respectively).

Number Manipulation Functions:

- ipart(x), fpart(x): integer and fractional part
- floor(x), ceil(x): floor and ceiling (closest integer below or above)
- abs(x): absolute value
- sign(x): sign function: $+1$ for $x > 0$, -1 for $x < 0$, 0 for $x = 0$
- mod(x,y): modulo operation: remainder of x divided by y
- quant(x,q): quantize x with quantization interval q
- quantToBits(x,b): quantizes the range between -1...+1 with a quantization interval which corresponds to a resolution of b bits
- clip(x, min, max): clips the input argument x to the range min...max
- softClip(x, min, max, s1, s2): clips the input argument x to the range min...max with tanh-shaped transition regions determined the 'softness' parameters s1 and s2 (assumed to be in the range 0...1). The softness parameters are optional with default values of 0.1.
- clamp(x, min, max): clamps x to the range between min and max, wrapping as needed

Comparison Functions:

- min(x,y,z,...), max(x,y,z,...): returns the value of the smallest or largest of the input arguments
- if(c,t,f): returns the value/result of 't' if 'c' is nonzero, the value/result of 'f' if 'c' is zero
- select(c,n,z,p): returns n for $c < 0$, z for $c = 0$ and p for $c > 0$
- equal(x,y): returns 1 for $x = y$, 0 otherwise
- below(x,y), above(x,y): below returns 1 for $x < y$, 0 otherwise, vice versa for above

Logical Functions:

- and(x,y): returns 1 if both x and y are nonzero, else returns 0
- or(x,y): returns 1 if either x or y is nonzero, else returns 0
- not(x): returns 1 if $x = 0$, else returns 0

Miscellaneous Functions:

- deg(x): converts x from radians to degrees
- rad(x): converts x from degrees to radians
- rescale(x, o1, o2, n1, n2): rescales a number x from the range o1...o2 to the range n1...n2
- poly(x, ...): evaluates the polynomial of x with polynomial coefficients given by the subsequent parameters which are assumed to be in decreasing order (the leftmost coefficient is the multiplier for the highest power of x, the rightmost coefficient is the constant term).

References

For deeper information on waveshaping, you might want to have a look at the article:
http://www.rs-met.com/documents/tutorials/Waveshaping.pdf

Credits

Thanks to Brian A. Vanderburg II for the ExprEval C++ library which empowers the expression evaluator engine used in this plugin. The project can be found on: http://sourceforge.net/projects/expreval