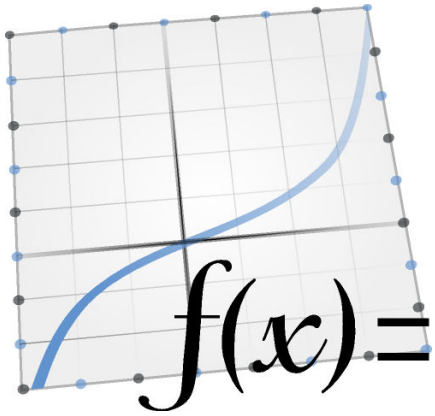
The KmPlot Handbook

Klaus-Dieter Möller
Philip Rodrigues
David Saxton



Contents

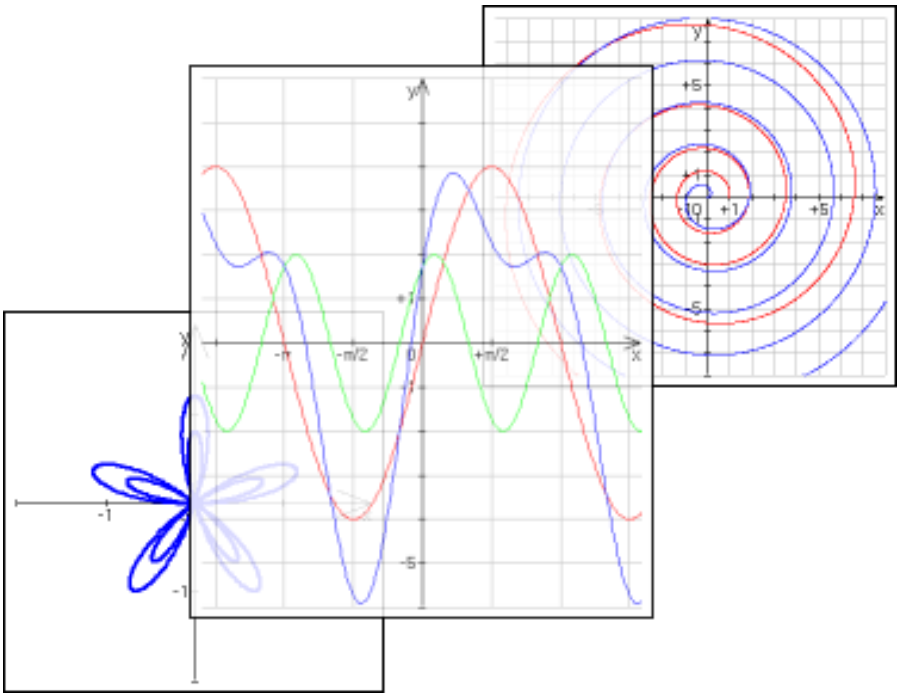
Abstract

KmPlot is a mathematical function plotter by KDE.
KmPlot is part of the KDE-EDU Project: http://edu.kde.org/

Chapter 1

Introduction

KmPlot is a mathematical function plotter by KDE. It has a powerful built-in parser. You can plot different functions simultaneously and combine them to build new functions.



KmPlot supports several different types of plots:

- Explicit cartesian plots of the form y = f(x).
- Parametric plots, where the x and y components are specified as functions of an independent variable.
- Polar plots of the form r = r(ϑ).
- Implicit plots, where the x and y coordinates are related by an expression.
- Explicit differential plots.

KmPlot also provides some numerical and visual features like:

- Filling and calculating the area between the plot and the first axis

- Finding maximum and minimum values
- Changing function parameters dynamically
- Plotting derivatives and integral functions.

These features help in learning the relationship between mathematical functions and their graphical representation in a coordinate system.

Chapter 2

First Steps With KmPlot

2.1 Simple Function Plot

In the sidebar on the left, there is the **Create** button with a drop down menu for creating new plots. Click on it, and select **Cartesian Plot**. The text box for editing the current equation will be focused. Replace the default text with

```
y = x^2
```

and press **Enter**. This will draw the plot of $y = x^2$ in the coordinate system. Clicking on the **Create** button again, select **Cartesian Plot**, and this time enter the text

```
y = 5sin(x)
```

to get another plot.

Click on one of the lines you have just plotted. Now the crosshair becomes the color of the current plot and is attached to the it. You can use the mouse to move the crosshair along the plot. In the status bar at the bottom of the window the coordinates of the current position is displayed. Note that if the plot touches the horizontal axis the root will be displayed in the status bar, too.

Click the mouse again and the crosshair will be detached from the plot.

2.2 Edit Properties

Let us make some changes to the function and change the color of the plot.

The **Functions** sidebar lists all the functions that you have plotted. If **y = x^2** isn't already selected, select it. Here you have access to a lot of options. Let us rename the function and move the plot 5 units down. Change the function equation to

```
parabola(x) = x^2 - 5
```

and hit enter. To select another color for the plot, click the **Color** button in the section **Appearance** at the bottom of the function sidebar and select a new color.

> NOTE
> All changes can be undone via **Edit** → **Undo**.

Chapter 3

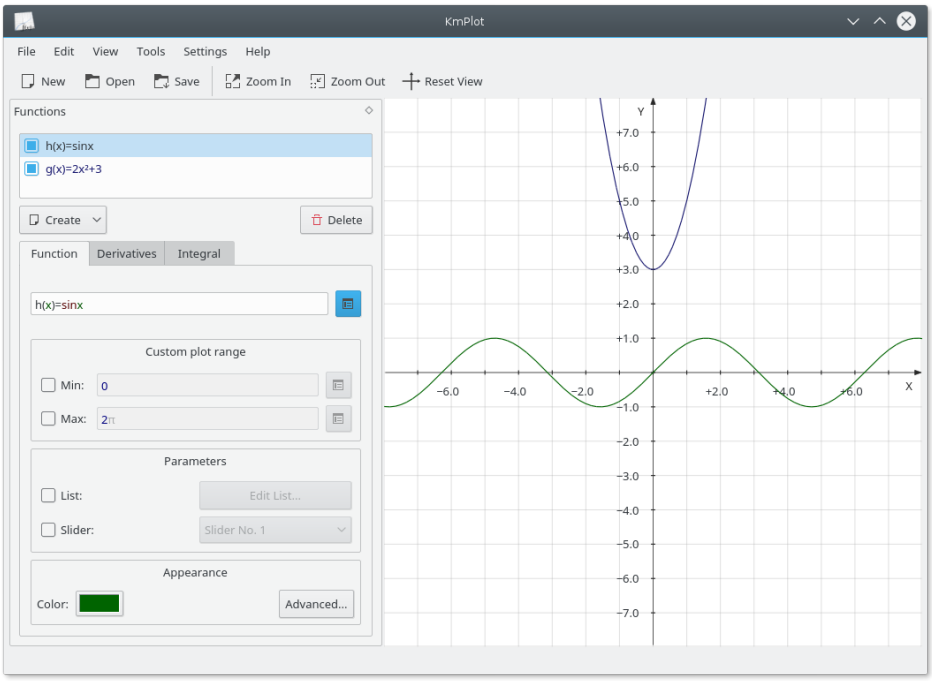
Using KmPlot

KmPlot deals with several different types of functions, which can be written in function form or as an equation:

- Cartesian plots can either be written as e.g. 'y = x^2', where x has to be used as the variable; or as e.g. 'f(a) = a^2', where the name of the variable is arbitrary.
- Parametric plots are similar to Cartesian plots. The x and y coordinates can be entered as equations in t, e.g. 'x = sin(t)', 'y = cos(t)', or as functions, e.g. 'f_x(s) = sin(s)', 'f_y(s) = cos(s)'.
- Polar plots are also similar to Cartesian plots. They can be either be entered as an equation in ϑ, e.g. 'r = ϑ', or as a function, e.g. 'f(x) = x'.
- For implicit plots, the name of the function is entered separately from the expression relating the x and y coordinates. If the x and y variables are specified via the function name (by entering e.g.'f(a,b)' as the function name), then these variables will be used. Otherwise, the letters x and y will be used for the variables.
- Explicit differential plots are differential equations whereby the highest derivative is given in terms of the lower derivatives. Differentiation is denoted by a prime ('). In function form, the equation will look like 'f''(x) = f' − f'. In equation form, it will look like 'y'' = y' − y'. Note that in both cases, the '(x)' part is not added to the lower order differential terms (so you would enter 'f'(x) = −f' and not 'f'(x) = −f(x)').

All the equation entry boxes come with a button on the right. Clicking this invokes the advanced **Equation Editor** dialog, which provides:

- A variety of mathematical symbols that can be used in equations, but aren't found on normal keyboards.
- The list of user constants and a button for editing them.
- The list of predefined functions. Note that if you have text already selected, it will be used as the function argument when a function is inserted. For example, if '1 + x' is selected in the equation 'y = 1 + x', and the sine function is chosen, then the equation will become ' y = sin(1+x)'.



3.1 Function Types

3.1.1 Cartesian Functions

To enter an explicit function (i.e., a function in the form y=f(x)) into KmPlot, just enter it in the following form:

```
f(x) = expression
```

where:

- *f* is the name of the function, and can be any string of letters and numbers.
- *x* is the horizontal coordinate, to be used in the expression following the equals sign. It is a dummy variable, so you can use any variable name you like to achieve the same effect.
- *expression* is the expression to be plotted, given in the appropriate syntax for KmPlot. See Section 5.4.

3.1.2 Parametric Functions

Parametric functions are those in which the x and y coordinates are defined by separate functions of another variable, often called t. To enter a parametric function in KmPlot, follow the procedure as for a Cartesian function for each of the x and y functions. As with Cartesian functions, you may use any variable name you wish for the parameter.

As an example, suppose you want to draw a circle, which has parametric equations x = sin(t), y = cos(t). After creating a parametric plot, enter the appropriate equations in the x and y boxes, i.e., **`f_x(t)=sin(t)`** and **`f_y(t)=cos(t)`**.

You can set some further options for the plot in the function editor:

Min, Max

These options control the range of the parameter t for which the function is plotted.

3.1.3 Functions in Polar Coordinates

Polar coordinates represent a point by its distance from the origin (usually called r), and the angle a line from the origin to the point makes with the horizontal axis (usually represented by ϑ the Greek letter theta). To enter functions in polar coordinates, click the **Create** button and select **Polar Plot** from the list. In the definition box, complete the function definition, including the name of the theta variable you want to use, e.g., to draw the Archimedes' spiral r = ϑ, enter:

```
r(ϑ) = ϑ
```

Note that you can use any name for the theta variable, so 'r(t) = t' or 'f(x) = x' will produce exactly the same output.

3.1.4 Implicit Functions

An implicit expression relates the x and y coordinates as an equality. To create a circle, for example, click the **Create** button and select **Implicit Plot** from the list. Then, enter into the equation box (below the function name box) the following:

```
x^2 + y^2 = 25
```

3.1.5 Differential Functions

KmPlot can plot explicit differential equations. These are equations of the form $y^{(n)} = F(x,y',y'',...,y^{(n-1)})$, where y^k is the k^{th} derivative of y(x). KmPlot can only interpret the derivative order as the number of primes following the function name. To draw a sinusoidal curve, for example, you would use the differential equation **y'' = − y** or **f''(x) = −f**.

However, a differential equation on its own isn't enough to determine a plot. Each curve in the diagram is generated by a combination of the differential equation and the initial conditions. You can edit the initial conditions by clicking on the **Initial Conditions** tab when a differential equation is selected. The number of columns provided for editing the initial conditions is dependent on the order of the differential equation.

You can set some further options for the plot in the function editor:

Step

The step value in the precision box is used in numerically solving the differential equation (using the Runge Kutta method). Its value is the maximum step size used; a smaller step size may be used if part of the differential plot is zoomed in close enough.

3.2 Combining Functions

Functions can be combined to produce new ones. Simply enter the functions after the equals sign in an expression as if the functions were variables. For example, if you have defined functions f(x) and g(x), you can plot the sum of f and g with:

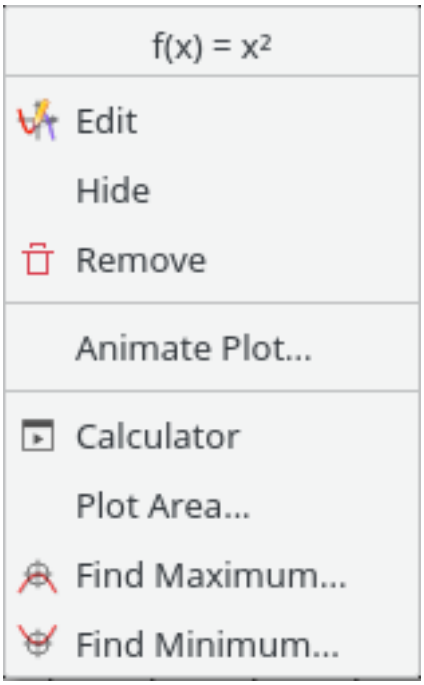
```
sum(x) = f(x) + g(x)
```

3.3 Changing the appearance of functions

To change the appearance of a function's graph on the main plot window, select the function in the **Functions** sidebar. You can change the plot's line width, color and many other aspects by clicking on the **Color** or **Advanced...** button at the bottom of the section **Appearance**.

If you are editing a Cartesian function, the function editor will have three tabs. In the first one you specify the equation of the function. The **Derivatives** tab lets you draw the first and second derivative to the function. With the **Integral** tab you can draw the integral of the function.

3.4 Popup menu



When right-clicking on a plot function or a single-point parametric plot function a popup menu will appear. In the menu there are three items available:

Edit
: Selects the function in the **Functions** sidebar for editing.

Hide
: Hides the selected graph. Other plots of the graph's function will still be shown.

Remove
: Removes the function. All its graphs will disappear.

Animate Plot...
: Displays the **Parameter Animator** dialog.

Calculator
: Opens the **Calculator** dialog.

Depending on the plot type, there will also be up to four tools available:

Plot Area...

Select the minimum and maximum horizontal values for the graph in the new dialog that appears. Calculates the integral and draws the area between the graph and the horizontal axis in the selected range in the color of the graph.

Find Minimum...

Find the minimum value of the graph in a specified range. The selected graph will be highlighted in the dialog that appears. Enter the lower and upper boundaries of the region in which you want to search for a minimum.

Note: You can also tell the plot to visually show the extreme points in the **Plot Appearance** dialog, accessible in the **Functions** sidebar by clicking on **Advanced...**.

Find Maximum...

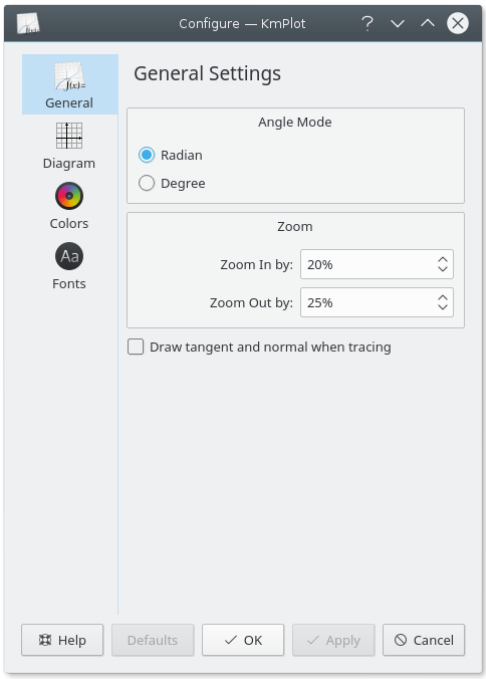
This is the same as **Find Minimum...** above, but finds the maximum value instead of the minimum value.

Chapter 4

Configuring KmPlot

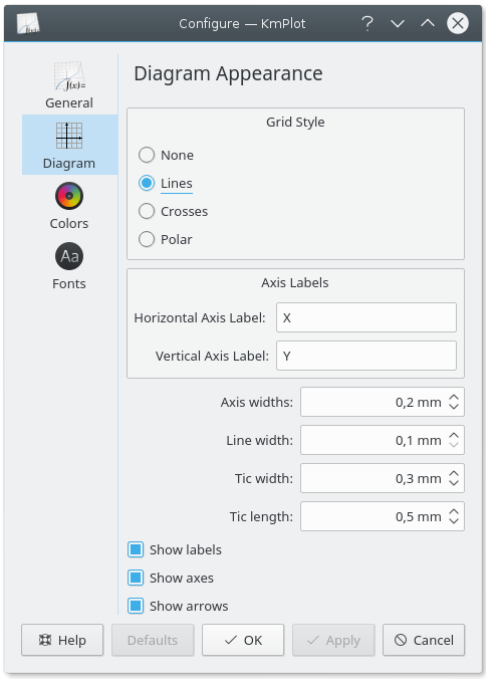
To access the KmPlot configuration dialog, select **Settings** → **Configure KmPlot...** The settings for **Constants...** can only be changed from the **Edit** menu and the **Coordinate System...** only from the **View** menu.

4.1 General Configuration



Here you can set global settings which automatic will be saved when you exit KmPlot. you can set angle-mode (radians and degrees), zoom in and zoom out factors, and whether to show advanced plot tracing.

4.2 Diagram Configuration



You can set the **Grid Style** to one of four options:

None
: No gridlines are drawn on the plot area

Lines
: Straight lines form a grid of squares on the plot area.

Crosses
: Crosses are drawn to indicate points where x and y have integer values (e.g., (1,1), (4,2) etc.).

Polar
: Lines of constant radius and of constant angle are drawn on the plot area.

Other options for the diagram appearance can also be configured:

Axis Labels
: Sets labels for the horizontal and vertical axes.

Axis width:
: Sets the width of the lines representing the axes.

Line width:
: Sets the width of the lines used for drawing the grid.

Tic width:
: Sets the width of the lines representing tics on the axes.

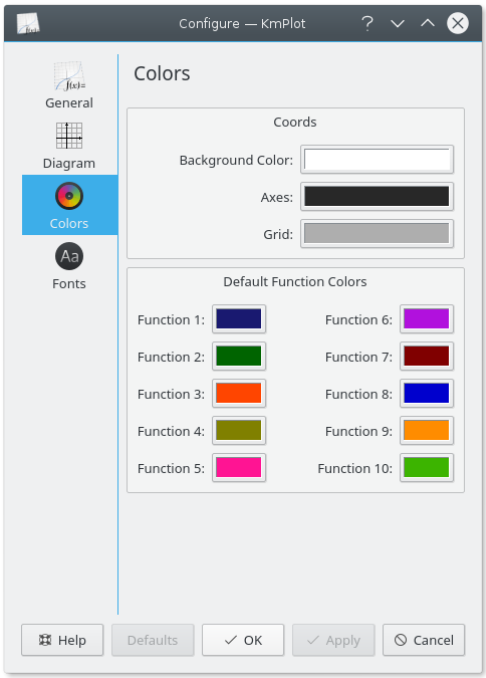
Tic length:
: Sets the length of the lines representing tics on the axes.

Show labels
If checked, the names of the axes are shown on the plot and the axes' tics are labeled.

Show axes
If checked, the axes are visible.

Show arrows
If checked, the axes are displayed with arrows at their ends.

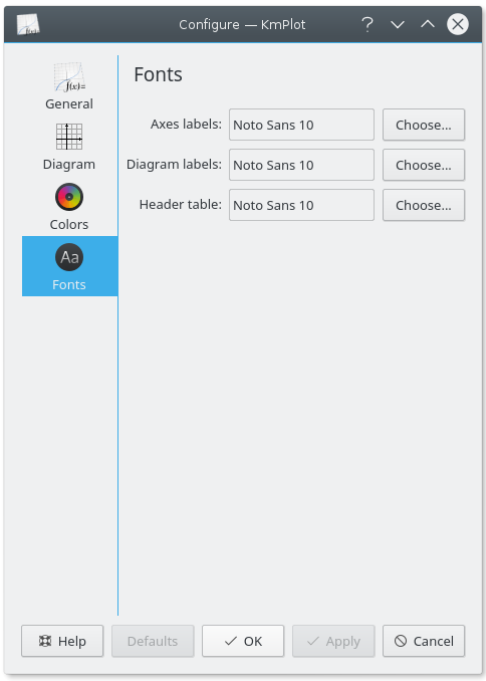
4.3 Colors Configuration



In the **Coords** section of the **Colors** configuration dialog, you can change the colors of the axes, the grid and the background of the main KmPlot area.

The **Default Function Colors** control which colors are cycled through when creating new functions.

4.4 Fonts Configuration



Axis labels
: The font used for drawing the axis numbers and x/y labels.

Diagram label
: The font used for drawing diagram labels (e.g., those showing the plot name or extreme points).

Header table
: The font used for drawing the header when printing a plot.

Chapter 5

KmPlot Reference

5.1 Function Syntax

Some syntax rules must be complied with:

```
name(var1[, var2])=term [;extensions]
```

name
The function name. If the first character is 'r' the parser assumes that you are using polar coordinates. If the first character is 'x' (for instance 'xfunc') the parser expects a second function with a leading 'y' (here 'yfunc') to define the function in parametric form.

var1
The function's variable

var2
The function 'group parameter'. It must be separated from the function's variable by a comma. You can use the group parameter to, for example, plot a number of graphs from one function. The parameter values can be selected manually or you can choose to have a slider bar that controls one parameter. By changing the value of the slider the value parameter will be changed. The slider can be set to an integer between 0 and 100.

term
The expression defining the function.

5.2 Predefined Function Names and Constants

All the predefined functions and constants that KmPlot knows can be shown by selecting **Help** → **Predefined Math Functions**, which displays this page of KmPlot's handbook.

These functions and constants and even all user defined functions can be used to determine the axes settings as well. See Section 5.7.1.

5.2.1 Trigonometric Functions

By default, the trigonometric functions work in radians. However, this can be changed via **Settings** → **Configure KmPlot**.

sin(x), arcsin(x), cosec(x), arccosec(x)
The sine, inverse sine, cosecant and inverse cosecant respectively.

cos(x), arccos(x), sec(x), arcsec(x)
The cosine, inverse cosine, secant and inverse secant respectively.

tan(x), arctan(x), cot(x), arccot(x)
The tangent, inverse tangent, cotangent and inverse cotangent respectively.

5.2.2 Hyperbolic Functions

The Hyperbolic Functions.

sinh(x), arcsinh(x), cosech(x), arccosech(x)
The hyperbolic sine, inverse sine, cosecant and inverse cosecant respectively.

cosh(x), arccosh(x), sech(x), arcsech(x)
The hyperbolic cosine, inverse cosine, secant and inverse secant respectively.

tanh(x), arctanh(x), coth(x), arccoth(x)
The hyperbolic tangent, inverse tangent, cotangent and inverse cotangent respectively.

5.2.3 Other Functions

sqr(x)
The square x^2 of x.

sqrt(x)
The square root of x.

sign(x)
The sign of x. Returns 1 if x is positive, 0 if x is zero, or -1 if x is negative.

H(x)
The Heaviside Step Function. Returns 1 if x is positive, 0.5 if x is zero, or 0 if x is negative.

exp(x)
The exponent e^x of x.

ln(x)
The natural logarithm (inverse exponent) of x.

log(x)
The logarithm of x to base 10.

abs(x)
The absolute value of x.

floor(x)
Rounds x to closest integer less than or equal to x.

ceil(x)
Rounds x to the closest integer greater than or equal to x.

round(x)

Rounds x to the closest integer.

gamma(x)

The gamma function.

factorial(x)

The factorial of x.

min(x_1,x_2,...,x_n)

Returns the minimum of the set of numbers {x_1,x_2,...,x_n}.

max(x_1,x_2,...,x_n)

Returns the maximum of the set of numbers {x_1,x_2,...,x_n}.

mod(x_1,x_2,...,x_n)

Returns the modulus (Euclidean length) of the set of numbers {x_1,x_2,...,x_n}.

5.2.4 Predefined Constants

pi, π

Constants representing π (3.14159...).

e

Constant representing Euler's Number e (2.71828...).

5.3 Extensions

An extension for a function is specified by entering a semicolon, followed by the extension, after the function definition. The extension can be entered by using the D-Bus method parser addFunction. None of the extensions are available for parametric functions but N and D[a,b] work for polar functions too. For example:

```
f(x)=x^2; A1
```

will show the graph $y=x^2$ with its first derivative. Supported extensions are described below:

N

The function will be stored but not be drawn. It can be used like any other user-defined or predefined function.

A1

The graph of the derivative of the function will be drawn additionally with the same color but less line width.

A2

The graph of the second derivative of the function will be drawn additionally with the same color but less line width.

D[a,b]

Sets the domain for which the function will be displayed.

P[a{,b...}]

Give a set of values of a group parameter for which the function should be displayed. For example: **`f(x,k)=k*x;P[1,2,3]`** will plot the functions f(x)=x, f(x)=2*x and f(x)=3*x. You can also use functions as the arguments to the P option.

Please note that you can do all of these operations by editing the items in the **Derivates** tab, the **Custom plot range** section and the **Parameters** section in the **Functions** sidebar too.

5.4 Mathematical Syntax

KmPlot uses a common way of expressing mathematical functions, so you should have no trouble working it out. The operators KmPlot understands are, in order of decreasing precedence:

^

The caret symbol performs exponentiation. e.g., **`2^4`** returns 16.

***, /**

The asterisk and slash symbols perform multiplication and division . e.g., **`3*4/2`** returns 6.

+, −

The plus and minus symbols perform addition and subtraction. e.g., **`1+3−2`** returns 2.

<, >, ≤, ≥

Comparison operators. They return 1 if the expression is true, otherwise they return 0. e.g., **`1 ≤ 2`** returns 1.

√

The square root of a number. e.g., **√`4`** returns 2.

|x|

The absolute value of x. e.g., **`|−4|`** returns 4.

±,

Each plus-minus sign gives two sets of plots: one in which the plus is taken, and one in which the minus is taken.e.g.. **`y = ±sqrt(1−x^2)`** will draw a circle. These, therefore, cannot be used in constants.

Note the precedence, which means that if parentheses are not used, exponentiation is performed before multiplication/division, which is performed before addition/subtraction. So **`1+2*4^2`** returns 33, and not, say 144. To override this, use parentheses. To use the above example, **`(1+2)*4)^2`** *will* return 144.

5.5 Plotting Area

By default, explicitly given functions are plotted for the whole of the visible part of the horizontal axis. You can specify an other range in the edit-dialog for the function. If the plotting area contains the resulting point it is connected to the last drawn point by a line.

Parametric and polar functions have a default plotting range of 0 to 2π. This plotting range can also be changed in the **Functions** sidebar.

5.6 Crosshair Cursor

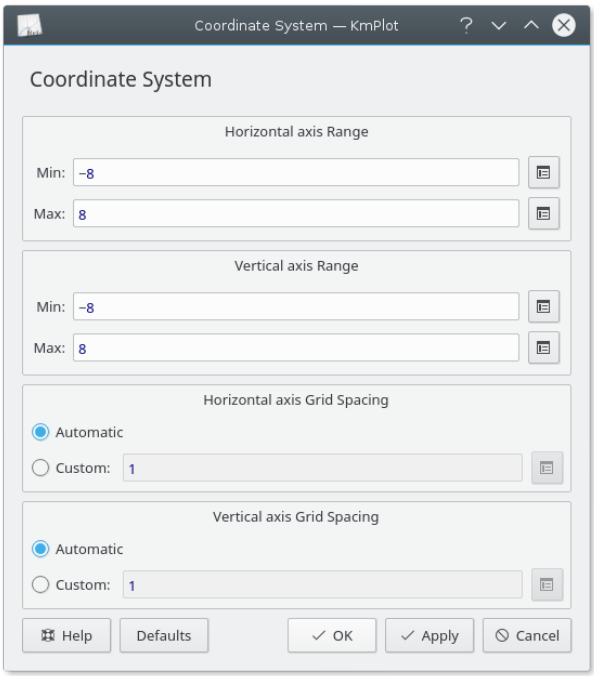
While the mouse cursor is over the plotting area the cursor changes to a crosshair. The current coordinates can be seen at the intersections with the coordinate axes and also in the status bar at the bottom of the main window.

You can trace a function's values more precisely by clicking onto or next to a graph. The selected function is shown in the status bar in the right column. The crosshair then will be caught and be colored in the same color as the graph. If the graph has the same color as the background color, the crosshair will have the inverted color of the background. When moving the mouse or pressing the keys Left or Right the crosshair will follow the function and you see the current horizontal and vertical value. If the crosshair is close to vertical axis, the root-value is shown in the statusbar. You can switch function with the Up and Down keys. A second click anywhere in the window or pressing any non-navigating key will leave this trace mode.

For more advanced tracing, open up the configuration dialog, and select **Draw tangent and normal when tracing** from the **General Settings** page. This option will draw the tangent, normal and oscillating circle of the plot currently being traced.

5.7 Coordinate System Configuration

To open this dialog select **View** → **Coordinate System...** from the menubar.



5.7.1 Axes Configuration

Horizontal axis Range

Sets the range for the horizontal axis scale. Note that you can use the predefined functions and constants (see Section 5.2) as the extremes of the range (e.g., set **Min:** to `2*pi`). You can even use functions you have defined to set the extremes of the axis range. For example, if you have defined a function `f(x) = x^2`, you could set **Min:** to `f(3)`, which would make the lower end of the range equal to 9.

Vertical axis Range

Sets the range for the vertical axis. See 'Horizontal axis Range' above.

Horizontal axis Grid Spacing

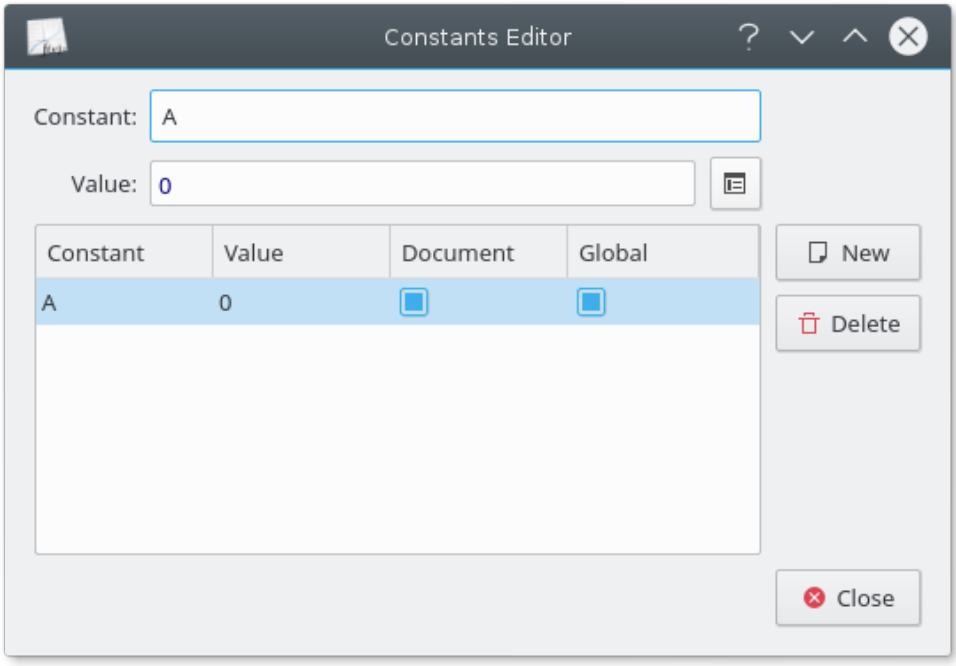
This controls the spacing between grid lines in the horizontal direction. If **Automatic** is selected, then KmPlot will try to find a grid line spacing of about two centimeters that is also numerically nice. If **Custom** is selected, then you can enter the horizontal grid spacing. This value will be used regardless of the zoom. For example, if a value of 0.5 is entered, and the x range is 0 to 8, then 16 grid lines will be shown.

Vertical axis Grid Spacing

This controls the spacing between grid lines in the vertical direction. See 'Horizontal axis Grid Spacing' above.

5.8 Constants Configuration

To open this dialog select **Edit** → **Constants...** from the menubar.



Constants can be used as part of an expression anywhere inside of KmPlot. Each constant must have a name and a value. Some names are invalid, however, such as existing function names or existing constants.

There are two options that control the scope of a constant:

Document

If you select the **Document** checkbox, then the Constant will be saved along with the current diagram when you save it to file. However, unless you have also selected the **Global** option, the constant will not be available between instances of KmPlot.

Global

If you select the **Global** checkbox, then the Constant's name and value will be written to KDE settings (where it can also be used by KCalc). The constant will not be lost when KmPlot is closed, and will be available again for use when KmPlot is started again.

Chapter 6

Command Reference

6.1 Menu Items

Apart from the common KDE menus described in the Menu chapter of the KDE Fundamentals documentation KmPlot has these application specific menu entries:

6.1.1 The File Menu

File → Export...

Exports the plotted graphs to an image file in all formats supported by KDE.

6.1.2 The Edit Menu

Edit → Constants...

Displays the **Constants** dialog box. See Section 5.8.

6.1.3 The View Menu

The first three items in the menu are related to zooming.

View → Zoom In (Ctrl+1)

This tool can be operator in two different manners. To zoom in on a point on the graph, click on it. To zoom in on a specific section of the graph, hold and drag the mouse to form a rectangle, which will be the new axes ranges when the mouse button is released.

View → Zoom Out (Ctrl+2)

The tool can also be used in two different manners. To zoom out and center on a point, click on that point. To fit the existing view into a rectangle, hold and drag the mouse to form that rectangle.

View → Fit Widget to Trigonometric Functions

The scale will be adapted to trigonometric functions. This works both for radians and degrees.

View → **Reset View**
Resets the view.

View → **Coordinate System...**
Displays the **Coordinate System** dialog box. See Section 5.7.

View → **Show Sliders**
Toggles the visibility of the slider dialog. In the dialog move a slider to change the parameter of the function plot connected to it.

Enable this on the Function tab and select one of the sliders to change the parameter value dynamically. The values vary from 0 (left) to 10 (right) by default, but can be changed in the slider dialog.

For a small tutorial see Using Sliders.

6.1.4 The Tools Menu

This menu contains some tools for the functions that can be useful:

Tools → **Calculator**
Opens the **Calculator** dialog.

Tools → **Plot Area...**
Select a graph and the values of the horizontal axis in the new dialog that appears. Calculates the integral and draws the area between the graph and the horizontal axis in the range of the selected values in the color of the graph.

Tools → **Find Minimum...**
Find the minimum value of the graph in a specified range.

Tools → **Find Maximum...**
Find the maximum value of the graph in a specified range.

6.1.5 The Help Menu

KmPlot has a standard KDE **Help** with one addition:

Help → **Predefined Math Functions...**
Opens this handbook with a list of the predefined function names and constants that KmPlot knows.

Chapter 7

Scripting KmPlot

You can write scripts for KmPlot using D-Bus. For example, if you want to define a new function **`f(x)=2sin x+3cos x`**, set its line width to 20 and then draw it, you type in a console:

qdbus org.kde.kmplot-PID /parser org.kde.kmplot.Parser.addFunction ˝f(x)=2sin x+3cos x˝ ˝˝ As a result, the new function's id number will be returned, or -1 if the function could not be defined.

qdbus org.kde.kmplot-PID /parser org.kde.kmplot.Parser.setFunctionFLineWidth ID 20 This command sets the function with the id number ID the line width to 20.

qdbus org.kde.kmplot-PID /view org.kde.kmplot.View.drawPlot This command repaints the window so that the function get visible.

A list of the available functions:

/kmplot org.kde.kmplot.KmPlot.fileOpen url

Load the file *url*.

/maindlg org.kde.kmplot.MainDlg.isModified

Returns true if any changes are done.

/maindlg org.kde.kmplot.MainDlg.checkModified

If there are any unsaved changes, a dialog appears to save, discard or cancel the plots.

/maindlg org.kde.kmplot.MainDlg.editAxes

Opens the coordinate system edit dialog.

/maindlg org.kde.kmplot.MainDlg.toggleShowSlider

Shows/hides parameter slider window.

/maindlg org.kde.kmplot.MainDlg.slotSave

Saves the functions (opens the save dialog if it is a new file).

/maindlg org.kde.kmplot.MainDlg.slotSaveas

The same as choosing **File** → **Save As** in the menu.

/maindlg org.kde.kmplot.MainDlg.slotPrint

Opens the print dialog.

/maindlg org.kde.kmplot.MainDlg.slotResetView

The same as choosing **View** → **Reset View** in the menu.

/maindlg org.kde.kmplot.MainDlg.slotExport

Opens the export dialog.

/maindlg org.kde.kmplot.MainDlg.slotSettings

Opens the settings dialog.

/maindlg org.kde.kmplot.MainDlg.slotNames

Shows the predefined math functions in the handbook.

/maindlg org.kde.kmplot.MainDlg.findMinimumValue

The same as choosing **Tools** → **Minimum Value...** in the menu.

/maindlg org.kde.kmplot.MainDlg.findMaximumValue

The same as choosing **Tools** → **Maximum Value...** in the menu.

/maindlg org.kde.kmplot.MainDlg.graphArea

The same as choosing **Tools** → **Plot Area** in the menu.

/maindlg org.kde.kmplot.MainDlg.calculator

The same as choosing **Tools** → **Calculator** in the menu.

/parser org.kde.kmplot.Parser.addFunction f_str0 f_fstr1

Adds a new function with the expressions *f_str0* and *f_str1*. If the expression does not contain a function name, it will be auto-generated. The id number of the new function is returned, or -1 if the function could not be defined.

/parser org.kde.kmplot.Parser.removeFunction id

Removes the function with the id number *id*. If the function could not be deleted, false is returned, otherwise true.

/parser org.kde.kmplot.Parser.setFunctionExpression id eq f_str

Sets the expression for the function with the id number *id* to *f_str*. Returns true if it succeed, otherwise false.

/parser org.kde.kmplot.Parser.countFunctions

Returns the number of functions (parametric functions are calculated as two).

/parser org.kde.kmplot.Parser.listFunctionNames

Returns a list with all functions.

/parser org.kde.kmplot.Parser.fnameToID f_str

Returns the id number of *f_str* or -1 if the function name *f_str* was not found.

/parser org.kde.kmplot.Parser.functionFVisible id

Returns true if the function with the ID *id* is visible, otherwise false.

/parser org.kde.kmplot.Parser.functionF1Visible id

Returns true if the first derivative of the function with the ID *id* is visible, otherwise false.

/parser org.kde.kmplot.Parser.functionF2Visible id

Returns true if the second derivative of the function with the ID *id* is visible, otherwise false.

/parser org.kde.kmplot.Parser.functionIntVisible id

Returns true if the integral of the function with the ID *id* is visible, otherwise false.

/parser org.kde.kmplot.Parser.setFunctionFVisible id visible

Shows the function with the ID *id* if *visible* is true. If *visible* is false, the function will be hidden. True is returned if the function exists, otherwise false

/parser org.kde.kmplot.Parser.setFunctionF1Visible id visible

Shows the first derivative of the function with the ID *id* if *visible* is true. If *visible* is false, the function will be hidden. True is returned if the function exists, otherwise false.

/parser org.kde.kmplot.Parser.setFunctionF2Visible id visible

Shows the second derivative of the function with the ID *id* if *visible* is true. If *visible* is false, the function will be hidden. True is returned if the function exists, otherwise false.

/parser org.kde.kmplot.Parser.setFunctionIntVisible id visible

Shows the integral of the function with the ID *id* if *visible* is true. If *visible* is false, the function will be hidden. True is returned if the function exists, otherwise false.

/parser org.kde.kmplot.Parser.functionStr id eq

Returns the function expression of the function with the ID *id*. If the function not exists, an empty string is returned instead.

/parser org.kde.kmplot.Parser.functionFLineWidth id

Returns the line width of the function with the ID *id*. If the function not exists, 0 is returned.

/parser org.kde.kmplot.Parser.functionF1LineWidth id

Returns the line width of the first derivative of the function with the ID *id*. If the function not exists, 0 is returned.

/parser org.kde.kmplot.Parser.functionF2LineWidth id

Returns the line width of the second derivative of the function with the ID *id*. If the function not exists, 0 is returned.

/parser org.kde.kmplot.Parser.functionIntLineWidth id

Returns the line width of the integral of the function with the ID *id*. If the function not exists, 0 is returned.

/parser org.kde.kmplot.Parser.setFunctionFLineWidth id linewidth

Sets the line width of the function with the ID *id* to *linewidth*. True is returned if the function exists, otherwise false.

/parser org.kde.kmplot.Parser.setFunctionF1LineWidth id linewidth

Sets the line width of the first derivative of the function with the ID *id* to *linewidth*. True is returned if the function exists, otherwise false.

/parser org.kde.kmplot.Parser.setFunctionF2LineWidth id linewidth

Sets the line width of the second derivative of the function with the ID *id* to *linewidth*. True is returned if the function exists, otherwise false.

/parser org.kde.kmplot.Parser.setFunctionIntLineWidth id linewidth

Sets the line width of the integral of the function with the ID *id* to *linewidth*. True is returned if the function exists, otherwise false.

/parser org.kde.kmplot.Parser.functionParameterList id

Returns a list with all the parameter values for the function with the ID *id*.

/parser org.kde.kmplot.Parser.functionAddParameter id new_parameter

Adds the parameter value *new_parameter* to the function with the ID *id*. True is returned if the operation succeed, otherwise false.

/parser org.kde.kmplot.Parser.functionRemoveParameter id remove_parameter

Removes the parameter value *remove_parameter* from the function with the ID *id*. True is returned if the operation succeed, otherwise false.

/parser org.kde.kmplot.Parser.functionMinValue id

Returns the minimum plot range value of the function with the ID *id*. If the function not exists or if the minimum value is not defined, an empty string is returned.

/parser org.kde.kmplot.Parser.functionMaxValue id

Returns the maximum plot range value of the function with the ID *id*. If the function not exists or if the maximum value is not defined, an empty string is returned.

/parser org.kde.kmplot.Parser.setFunctionMinValue id min

Sets the minimum plot range value of the function with the ID *id* to *min*. True is returned if the function exists and the expression is valid, otherwise false.

/parser org.kde.kmplot.Parser.setFunctionMaxValue id max

Sets the maximum plot range value of the function with the ID *id* to *max*. True is returned if the function exists and the expression is valid, otherwise false.

/parser org.kde.kmplot.Parser.functionStartXValue id

Returns the initial x point for the integral of the function with the ID *id*. If the function not exists or if the x-point-expression is not defined, an empty string is returned.

/parser org.kde.kmplot.Parser.functionStartYValue id

Returns the initial y point for the integral of the function with the ID *id*. If the function not exists or if the y-point-expression is not defined, an empty string is returned.

/parser org.kde.kmplot.Parser.setFunctionStartValue id x y

Sets the initial x and y point for the integral of the function with the ID *id* to *x* and *y*. True is returned if the function exists and the expression is valid, otherwise false.

/view org.kde.kmplot.View.stopDrawing

If KmPlot currently is drawing a function, the procedure will stop.

/view org.kde.kmplot.View.drawPlot

Redraws all functions.

Chapter 8

Credits and License

KmPlot

Program copyright 2000-2002 Klaus-Dieter Möller kd.moeller@t-online.de

CONTRIBUTORS

- CVS: Robert Gogolok mail@robert-gogoloh.de
- Porting GUI to KDE 3 and Translating: Matthias Messmer bmlmessmer@web.de
- Various improvements: Fredrik Edemar f_edemar@linux.se
- Porting to Qt 4, UI improvements, features: David Saxton david@bluehaze.org