

PocketCAS 3.3 User's Manual

Daniel Alm

November 5, 2013

Contents

Introduction	3
1 Getting Started	4
1.1 The document view	4
1.1.1 Entry types	4
1.2 Editing entries	5
1.3 The main menu	6
1.3.1 Application settings	6
1.3.2 Document settings	7
1.3.3 iCloud sync	7
1.4 Tutorials	8
1.5 Additional concepts & tips	8
1.5.1 Variables	8
1.5.2 Functions	8
1.5.3 Export capabilities	8
1.5.4 Undo	9
1.5.5 Implicit multiplication	9
1.5.6 Recalculating the whole document	9
2 Features of the Entry Editor	10
2.1 The keyboard	10
2.2 The info bar	11
2.3 The matrix editor	12
2.4 The entry type selector	12
2.5 The function list	13
2.5.1 The detail view	14
2.6 The variable list	14
3 Plotting	15
3.1 Using the graph view	15

3.2	Plot settings	15
3.2.1	Plot ranges	15
3.3	Plot types	17
3.3.1	Cartesian plots	17
3.3.2	Polar plots	18
3.3.3	Parametric plots	19
3.3.4	Implicit plots	19
3.3.5	Run charts	20
3.3.6	Cartesian 3D plots	21
3.3.7	Parametric 3D curve plots	21
3.3.8	Parametric 3D area plots	22
4	Advanced Capabilities	23
4.1	Linear algebra	23
4.1.1	Matrix and vector formats	23
4.1.2	Submatrix access	24
4.2	Solving	25
4.2.1	Curve sketching	25
4.3	Scripting	25

Introduction

PocketCAS is a great tool for performing sophisticated calculations on your mobile device. Unfortunately, this flexibility comes at a cost: It may not always be obvious how a specific problem can be approached best with PocketCAS. The tutorials shipped with the application can show you some approaches for common problems, but they only cover a small portion of what PocketCAS is capable of. This document is intended to introduce you to every feature of the application's interface and to provide some insight into its core concepts.

This manual does not provide a reference of every function available in PocketCAS. The application already contains a (searchable) function list with short descriptions and examples for every available function. There is a reference manual for the Xcas desktop software, which uses the same computation system as PocketCAS, available at <http://pocketcas.com/reference>. Please keep in mind that some functions of the desktop version (especially with regard to plotting) may not be available in PocketCAS.

If you find areas that should be covered in more detail, please let me know! This is the first version of the manual and feedback is very much appreciated.

Most of the descriptions here are given for the iPad version of PocketCAS, but things are very similar on the iPhone. If an interface feature is significantly different in the iPhone version, it will be mentioned separately.

Chapter 1

Getting Started

This chapter will introduce you into the main interface elements of the application. For an introduction on how to perform simple calculations, please refer to the *Introduction* tutorial, which can be reached from the main menu (cf. Section 1.4).

1.1 The document view

The first thing you'll see when starting PocketCAS is the *document view*. It shows an overview of all the *entries* in the current document. Each entry represents a different set of expressions that belong together. Pressing the *Edit* button allows you to rearrange, delete and insert entries, while tapping an entry lets you edit its contents.

1.1.1 Entry types

There are several distinct types of entries that are each handled differently by PocketCAS:

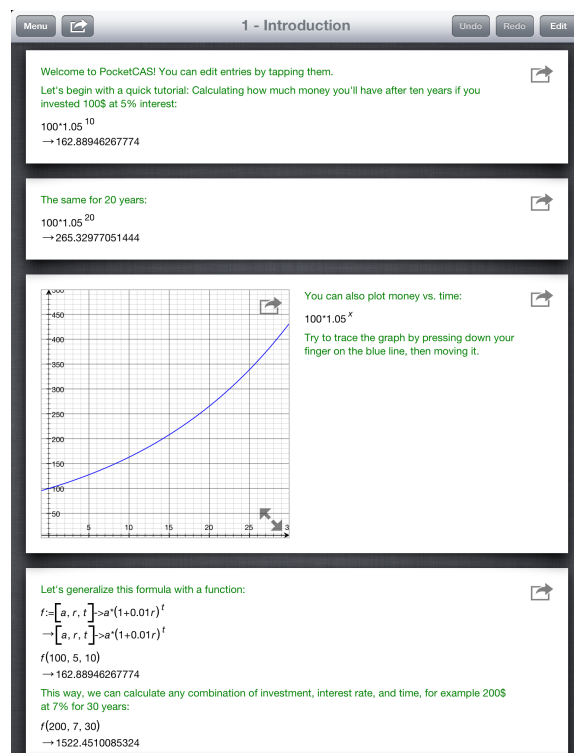


Figure 1.1: Screenshot of the document view

Calculations

A *calculation* consists of several *lines* whose contents are executed consecutively. This for example allows you to define a variable in the first line, then perform calculations with it in the subsequent lines. In the document view, the result of executing each line will be displayed right below it: Symbolic results are shown with an \rightarrow arrow, approximations with an \approx approximation sign.

Plots

Plot entries contain a special view that shows graphs of the expressions it contains. See Chapter 3 for details.

3D Plots


3D plot entries are different from regular plots as they contain expressions that need to be plotted in three (rather than two) dimensions. See Section 3.3.6.

Scripts

A *script's* lines are evaluated all at once in order to allow for multi-line expressions (e.g. complicated functions, loops, etc.). Thus, a script entry will show just one result line no matter how many text lines it consists of. See Section 4.3 for details.

1.2 Editing entries

After tapping an entry (or the *New Entry* button), the *entry editor* will be shown. It is described in more detail in Chapter 2. When you are done editing an entry, press the blue *Calculate* button in the top right of the editor or the *go!* key on the keyboard.

You can also add an image to a calculation or script entry via the  button. This image will be shown next to the corresponding entry. You can use this feature to e.g. illustrate a calculation with a sketch. Note that images will be scaled down so that the longer side is at most 1000 pixels long.

1.3 The main menu

The *main menu* is available from the *Menu* button on the top left of the document view. It allows you to create, load and save documents, access tutorials and change the settings, which are explained below.

In PocketCAS for iPhone, you can also export the current document from here (cf. Section 1.5.3).

1.3.1 Application settings

Auto-Approximate

When enabled, PocketCAS will check if it's suitable to show an approximation of the result, and if so, show it alongside the symbolic result. For example, if a calculation returned $\sqrt{2}$, PocketCAS would also output its numerical approximation (i.e. 1.4142...).

Default entry type

Lets you set a default type for new entries rather than have PocketCAS ask you each time. See Section 1.1.1 for an explanation of the differences between entry types.

Decimal notation

Gives you the option to have decimal numbers displayed in default (e.g. as 123456.0), scientific (1.23456E+05) or engineering (123.456E+03) notation.

Formula font size

Lets you change the font size for formulas. This can for example be useful for presentations using an external display.



Figure 1.2: Screenshot of the settings menu

Show function info bar

Sets whether the function info bar (see Section 2.2) should be displayed.

Portrait/Landscape editor font size

Determines the font size of the entry editor in portrait and landscape mode, respectively.

Keystroke sound

Determines whether or not to play a sound when you hit the keyboard.

Portrait/Landscape keyboard size

Lets you set the height of the mathematical keyboard.

1.3.2 Document settings

These settings will only apply to the current document.

If you prefer to use trigonometric functions with degrees rather than radians, you can do so by disabling the *Angles in Radian* setting here. When you enable *Complex Mode*, PocketCAS will include complex-valued values when solving and factoring. You can also have PocketCAS show all solutions for equations with trigonometric functions, rather than just those within one period.

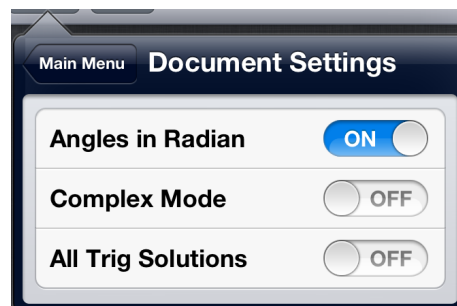


Figure 1.3: Screenshot of the document settings menu

1.3.3 iCloud sync

The main menu also holds the option to enable iCloud syncing. Upon activation, your documents will be transferred to iCloud and automatically synced to all your other devices where iCloud syncing is enabled in PocketCAS.

Note that your documents stored in iCloud will become unavailable if you decide to disable iCloud again at a later time.

iCloud syncing is still considered experimental. Make sure to back up your data before activating it!

1.4 Tutorials

There are several tutorials available from the main menu. Among other content, they offer examples for each possible plot type (cf. Section 3.3), show you how to work with units and introduce you to calculus and linear algebra with PocketCAS.

1.5 Additional concepts & tips

1.5.1 Variables

PocketCAS allows you to create custom variables. You do so by using the $:=$ operator. For example, in order to store the value 2 into the variable a , you'd enter $a := 2$. After assigning, you can use variables in other calculations and they will automatically be substituted with their corresponding values. This makes it possible to influence many calculation results at once by just changing the value of the variable they depend on.

While it's technically possible to use x as a variable, doing so is not recommended, because it is e.g. also used as the default variable by which derivatives and integrals are taken.


1.5.2 Functions

You can also define custom functions (which also might depend on variables). There are two ways to define e.g. a function $f(x, y)$ that returns the sum of x and y :

$$\begin{aligned} f(x, y) &:= x + y \\ f &:= (x, y) \rightarrow x + y \end{aligned}$$

1.5.3 Export capabilities

Several objects in PocketCAS can be exported in a variety of different formats.

Whenever you encounter an  icon, you will be able to select an export method from the resulting menu.

The following kinds of objects can currently be exported:

- The whole document
- All lines of a single entry
- The plot view of plot and 3D plot entries
- A single line of an entry (press and hold a finger on the line you wish to export to show a menu that lets you export it or copy it as text)

Possible export options include printing, saving an image to the photo roll, or creating a PDF file for emailing or usage in other applications. Not all kinds of objects support all export options, however.

1.5.4 Undo

PocketCAS features undo support for the entry editor as well as the document view. On the iPad, there are dedicated *Undo* and *Redo* buttons available on the respective toolbars. On the iPhone, shaking the device calls up the undo menu.

1.5.5 Implicit multiplication

PocketCAS has limited support for implicit multiplication: You can omit the multiplication sign between numbers and variable names.

But this is the only way you should use implicit multiplication in PocketCAS: The input $x(y)$, for example, will be interpreted as “the function x applied to y ”, not as $x * y$. And terms consisting of multiple letters, like xy , will be interpreted as independent variable names and not be converted to $x * y$ automatically.

1.5.6 Recalculating the whole document

On some occasions, you may want to re-evaluate every entry in the current document. You can do so by scrolling to the top and then pulling down the document view even further. A message will appear that tells you to release your finger to recalculate.

Chapter 2

Features of the Entry Editor

This chapter will explain the features of the entry editor.

2.1 The keyboard



Figure 2.1: Screenshot of the main mathematical keyboard

PocketCAS makes use of a customized keyboard to help you input mathematical expressions more quickly. If you need to access the default alphanumeric keyboard, just swipe up with one finger on the keyboard or tap the *ABC* button (iPad only).

Additional sections of the keyboard can be accessed via the leftmost column of buttons (or by swiping left and right if you are using PocketCAS in portrait mode on an iPhone).



Figure 2.2: Example of a key with alternative functions



Figure 2.3: The menu that appears when pressing such a key

Several keys have a slanted top right edge (see Figure 2.2). When you press and hold your finger on one of them, a menu will appear that lets you access related functions (Figure 2.3). For example, you can access the *asin*, *sinh* and *asinh* functions from the regular *sin* button. This is also useful for e.g. entering braces and brackets – just press your finger down on one of the parenthesis keys, then select the kind of parenthesis you'd like to insert.

If you keep your finger pressed on a function key for a short while, the info bar (see below) will expand to show a short description of the corresponding function.

Tip: If you have an external keyboard connected to your device, you can start a calculation by pressing the Page-Down key. The Page-Up key cancels editing.

2.2 The info bar



Figure 2.4: Screenshot of the info bar

When you tap a function button on the keyboard or select a function in the text field, the info bar will show which arguments it accepts. Pressing the blue button on the right will show more information.

Please note that the info bar will be hidden in landscape mode on the iPhone in order to save space.

You can press and hold your finger on the info bar in order to insert the current function. On the iPhone, double-tapping the info bar will call up the function list.

2.3 The matrix editor

The *matrix editor* can be invoked by tapping anywhere in the text field and then selecting the *Insert Matrix* option from the menu that appears. If you tapped inside an existing matrix, you can instead edit it.

If you are editing a one-row or one-column matrix (i.e. a vector), you can choose to create the vector as a list instead. See Section 4.1.1 for an explanation of the differences.

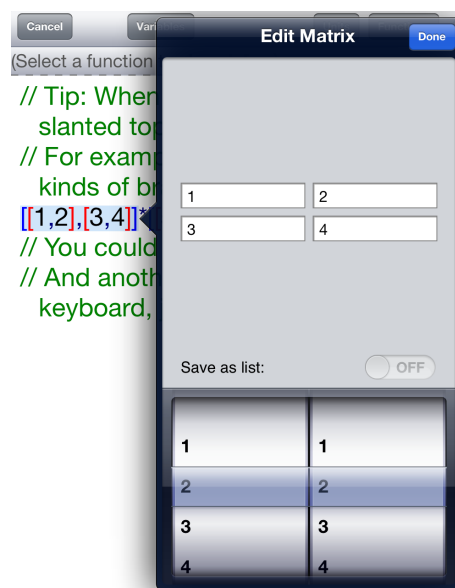


Figure 2.5: Screenshot of the matrix editor

2.4 The entry type selector

This control allows you to change the type of an entry while editing. Doing so also affects the interface: PocketCAS adapts the keyboard for the current entry type. On the iPad, selecting the *Calculation* or *Script* type will show an additional button on the top toolbar that lets you insert units or scripting functions, respectively. On the iPhone, the portrait keyboard is expanded with an appropriate section instead.



Figure 2.6: Screenshot of the entry type selector

In PocketCAS for iPad, the following features are available from the toolbars. In the iPhone version, you can access them by pressing the *Tools* button instead.

2.5 The function list

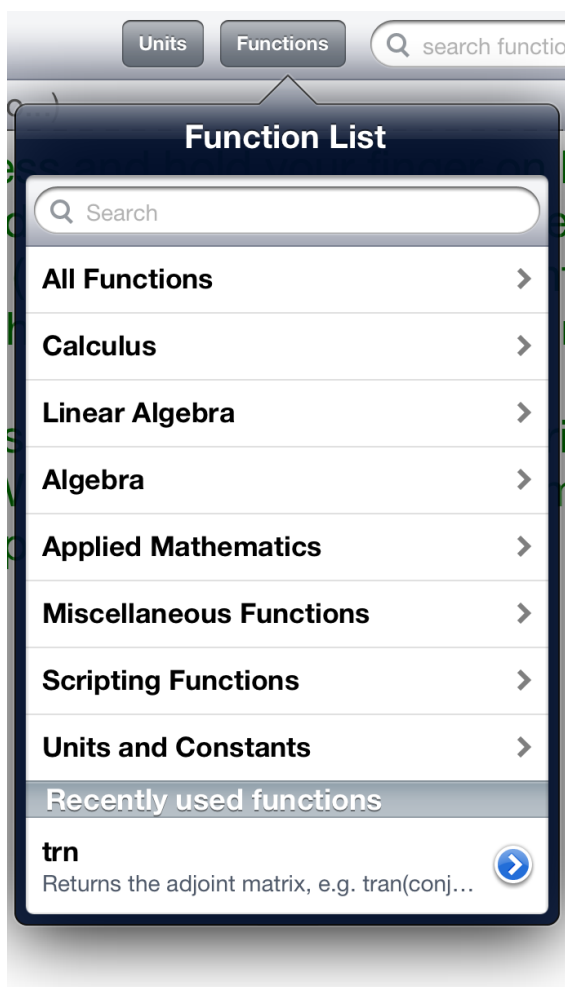


Figure 2.7: The main function list

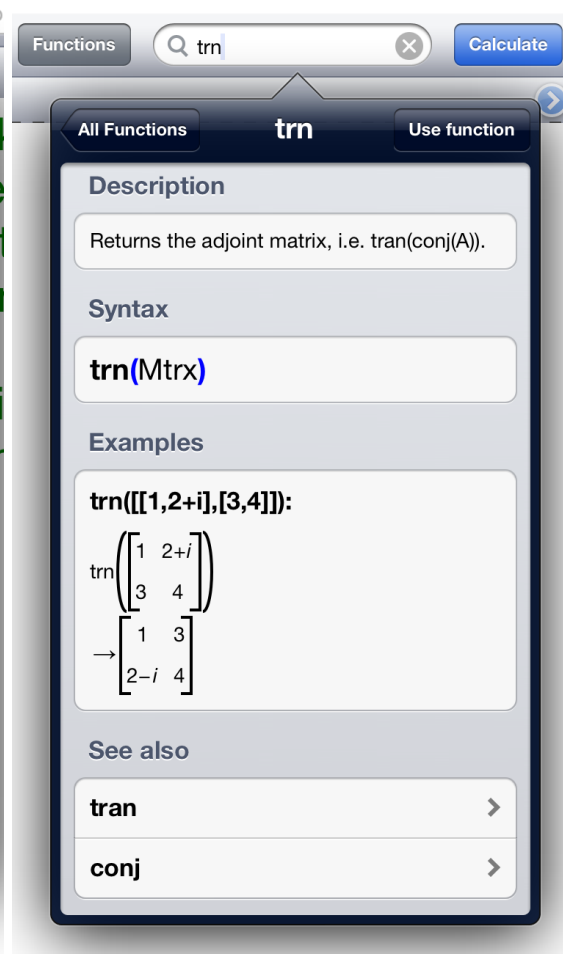


Figure 2.8: Screenshot of a function's detail view

The function list (see Figure 2.7) lets you browse all the functions and units available in PocketCAS. There are several categories to help you find the function you need. The main list also shows the functions that have been inserted most recently. Pressing the blue button

next to a function will show additional information:

2.5.1 The detail view

This view shows a short description of the function, its arguments, and possible examples and related functions (cf. Figure 2.8). For some examples, the corresponding result will also be shown.

2.6 The variable list

All the variables you've declared (cf. Section 1.5.1) will be listed here. You can insert them with a tap or delete them with a swipe.

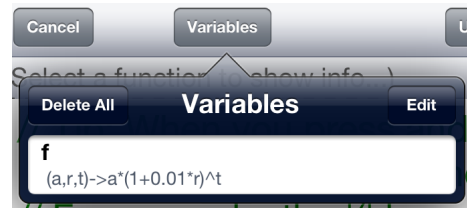





Figure 2.9: Screenshot of the variable list

Chapter 3

Plotting

3.1 Using the graph view

When you create an entry of the *Plot* type, it will contain a *graph view* next to the expressions you entered. You can move the visible area around by panning with one finger and zoom by pinching two fingers. The  button enables fullscreen mode, while  presents a menu where you can export the graph (see Section 1.5.3), show a value table (Section 3.3.1). The  button allows you to change the plot settings (see below).

When you press and hold your finger on a specific graph, you can trace a point along the graph's curve, showing the corresponding coordinates.

3.2 Plot settings

3.2.1 Plot ranges

The first section of the plot settings screen (shown in Figure 3.1) allows you to precisely specify which area of the graph should be visible. Via the *Reset Axis Ranges* button, you can either reset it to the default or (for 2D graphs) have PocketCAS automatically determine the Y range for an 1:1 aspect ratio. You can also enable logarithmic scaling for the X or Y axes.

In addition, you have the option to hide the grid and have the legend show up always or never. By default, it will automatically hide after a few seconds.

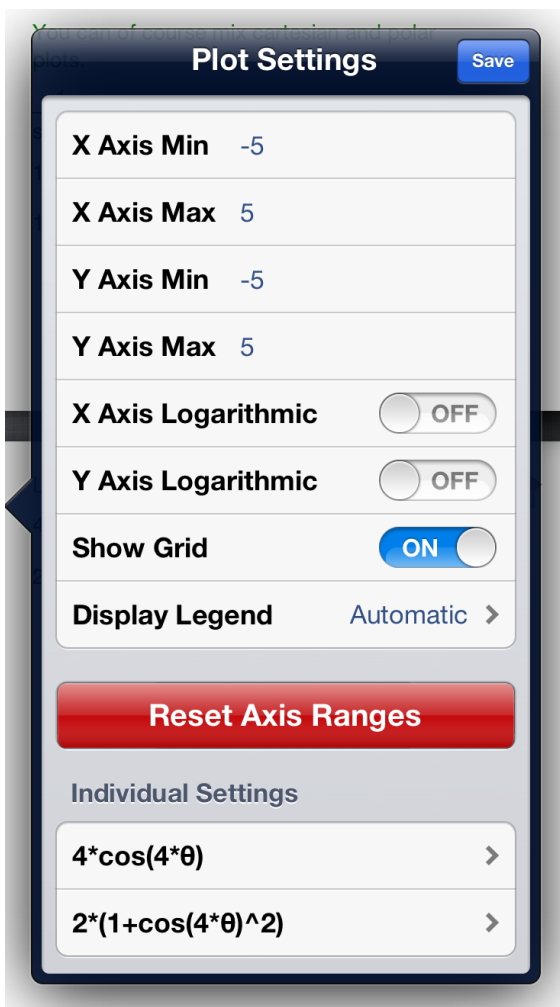


Figure 3.1: Screenshot of the Plot settings screen

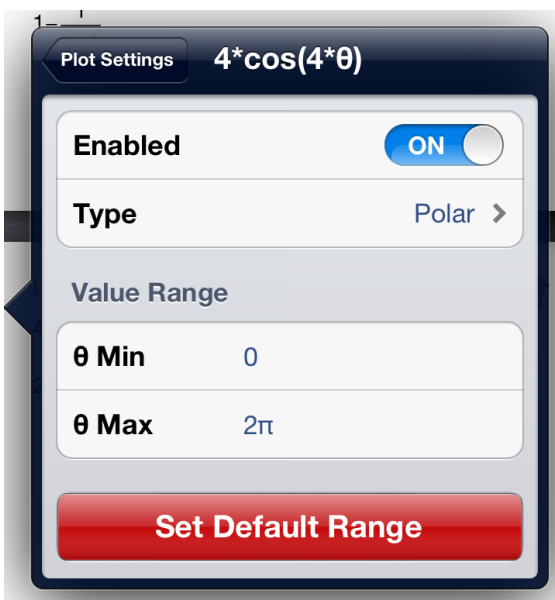


Figure 3.2: Screenshot of an individual plot's settings screen

For 3D graphs, this section also lets you choose the number of points that will be calculated in each space dimension. Note that PocketCAS will automatically calculate twice as many points when the graph is shown in fullscreen mode.

The *Individual Settings* section lets you edit each line on an individual basis (see Figure 3.2 for an example). You can exclude lines from plotting, change the plot type (see below) and change the corresponding plot variable's value range (if applicable).

3.3 Plot types

PocketCAS will try to determine the intended plot type automatically. If this fails, you can manually specify it in the corresponding plot's individual settings.

Please see the plotting tutorial for interactive examples of each plot type.

3.3.1 Cartesian plots

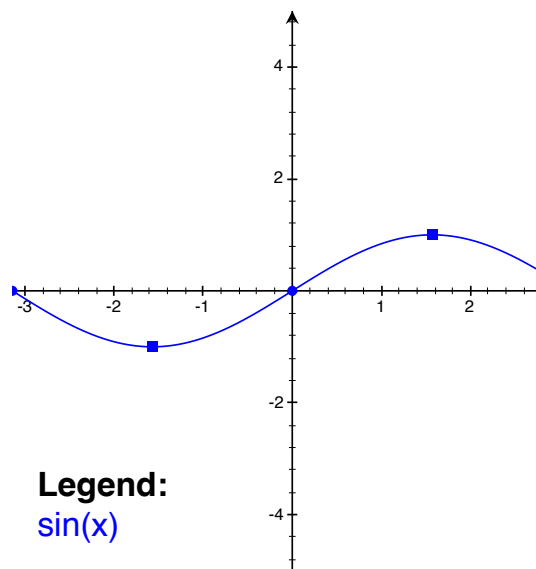


Figure 3.3: Cartesian plot of a function

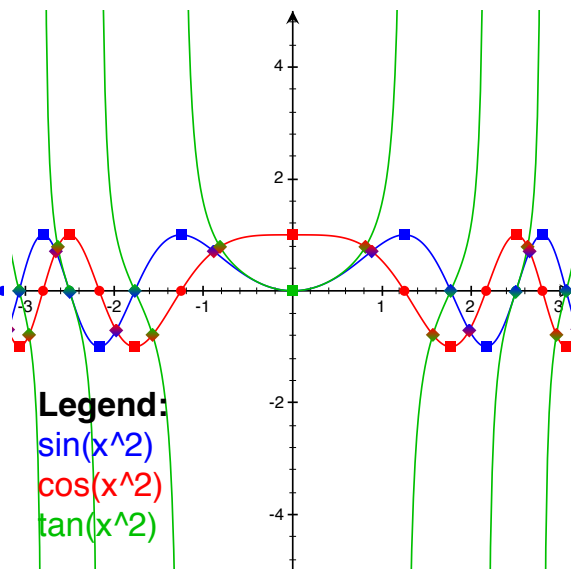


Figure 3.4: Three functions in a cartesian plot with zeroes, extremes and intersections

Cartesian plots (Figures 3.3 and 3.4) are the most “classic” kind of plot: A function that depends on x is plotted as the value of the y coordinate.


Note that for plotting e.g. $y = f(x) = x^2$, you'd only enter x^2 into the input.

Zeroes, extremes and intersections

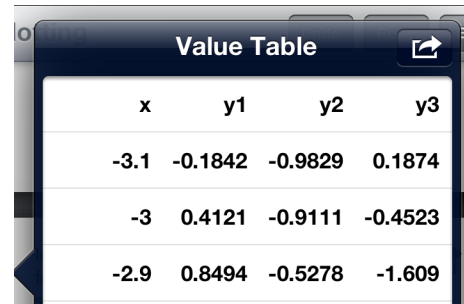
For cartesian plots, PocketCAS will automatically highlight special points of the graph (see Figure 3.4 for examples). Zeroes are denoted by a dot, extremes by a square and intersections between two graphs are shown with a tilted square

Tapping such a point will show its coordinates. Due to the finite resolution of the calculation, these coordinates may differ slightly from the exact value. You can increase accuracy by enabling fullscreen mode (as PocketCAS then plots at a higher resolution) or zooming in.

Value table

The graph view's  menu gives you the option to show a value table that you can then mail as a CSV file. The value table's range and step size are determined automatically depending on the currently visible range of the x coordinate.

This feature is currently only available for cartesian plots.



x	y1	y2	y3
-3.1	-0.1842	-0.9829	0.1874
-3	0.4121	-0.9111	-0.4523
-2.9	0.8494	-0.5278	-1.609

Figure 3.5: Screenshot of the value table screen

3.3.2 Polar plots

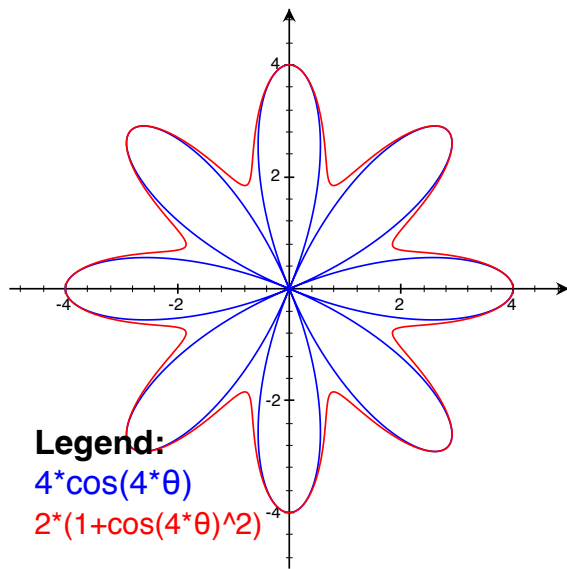


Figure 3.6: Polar plot of two functions

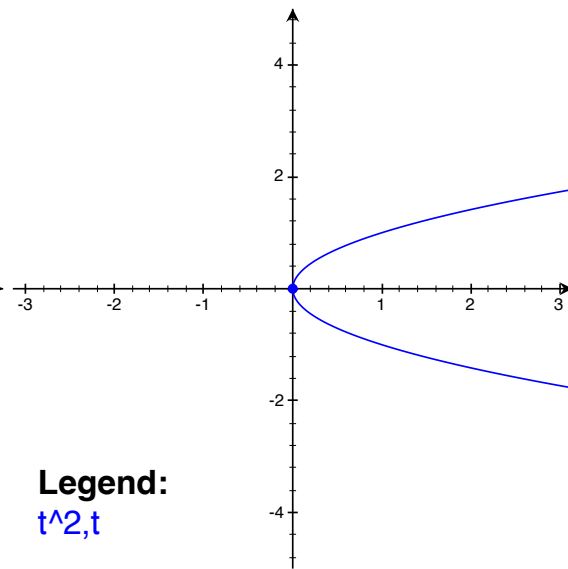


Figure 3.7: A parabola, drawn by means of a parametric plot

For a polar plot, you define a function $r = f(\theta)$ of the angle θ from the positive x axis in counter-clockwise direction. The value of this function is then used as the distance from the

origin. i.e. the final coordinates of the resulting point are determined by

$$\begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} r \sin(\theta) \\ r \cos(\theta) \end{pmatrix}.$$

The range of θ can be changed in the plot's individual settings.

An example is depicted in Figure 3.6.

3.3.3 Parametric plots

In parametric mode, you specify two functions of t (separated by a comma) that will be used for the x and y coordinate, respectively. For example, entering t^2, t will plot a curve where the points satisfy the equation

$$\begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} t^2 \\ t \end{pmatrix},$$

which represents a parabola opened in the direction of the positive x axis, as shown in Figure 3.7.

3.3.4 Implicit plots

Since version 3.2, PocketCAS also supports implicit plots. I.e., it plots the solutions of an equation of the form

$$f(x, y) = 0$$

for some function f . As a convenience, the right-hand side doesn't need to be zero, so entering e.g.

$$x^2 = 1 - y^2$$

would also plot the unit circle. Figure 3.8 shows a vertical line drawn via

$$x = 1$$

as well as an ellipsis defined as the solution set of

$$2 \cdot (x + 1.5)^2 + (y - 2.5)^2 = 3.$$

On the other hand, Figure 3.9 shows the hyperbolae defined by

$$x^2 - y^2 = 1$$

and

$$x^2 - y^2 = -1,$$

respectively.

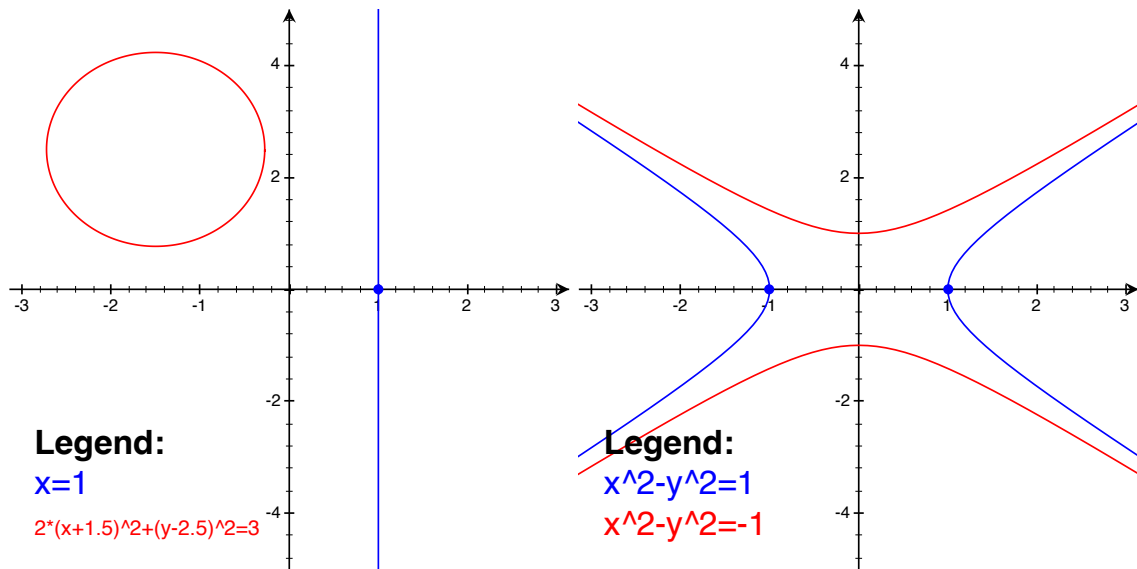


Figure 3.8: A vertical line and an ellipsis as an implicit plot Figure 3.9: Two implicitly defined hyperbolae implicit plot

3.3.5 Run charts

Run charts make it possible to manually specify the points which should be plotted. Just enter a matrix (cf. Section 4.1.1) with the x -coordinates in the first and the y -coordinates in the second row, such that each column of the matrix represents one point:

$$\begin{pmatrix} x_1 & x_2 & x_3 \\ y_1 & y_2 & y_3 \end{pmatrix}$$

By default, consecutive points are connected by a straight line. You can disable this in the run chart's individual settings.

See Figure 3.10 for an example.

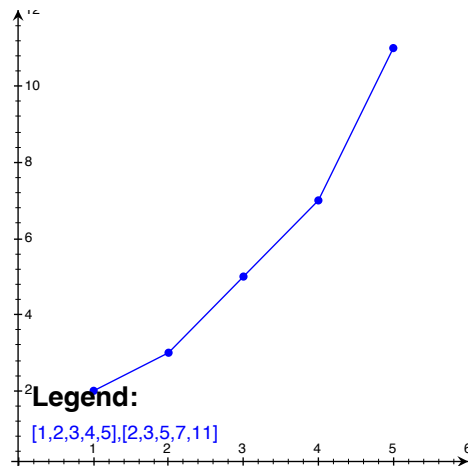


Figure 3.10: A run chart

3.3.6 Cartesian 3D plots

All plot types mentioned so far related to a curve in the two-dimensional plane. If you instead wish to plot the height field of a function depending on two variables, 3D plots are the right choice: The results of a function depending on the variables x and y are taken as the z -coordinate of each point (Figure 3.11 shows a plot of $z = \cos(\sqrt{x^2 + y^2})$).

As two- and three-dimensional plots are inherently different, you can not combine them. Thus, there's a distinct entry type for 3D plots.

Manipulating the graph view is different, too: Panning one finger rotates the graph, and pinching changes the scale of the z -axis. If you wish to change the range of the x and y variables, you have to do so in the plot settings.

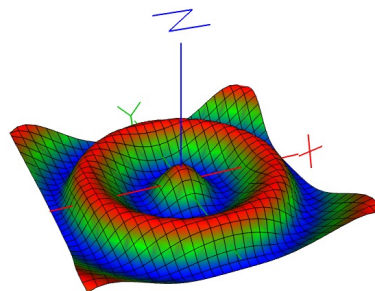


Figure 3.11: A 3D plot

3.3.7 Parametric 3D curve plots

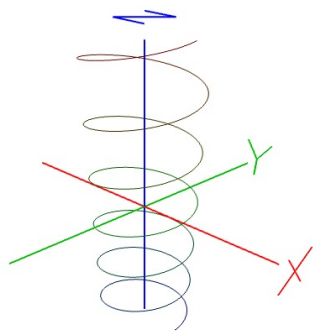


Figure 3.12: A helix, drawn as a parametric 3D curve plot

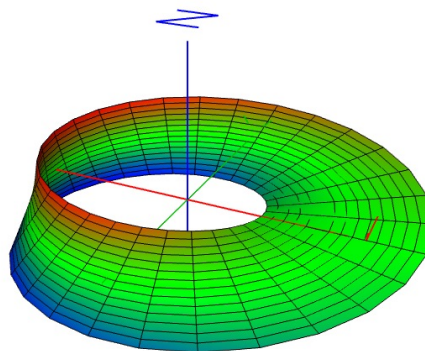


Figure 3.13: A Möbius strip as an example for a parametric 3D area plot

These are similar to parametric 2D plots, but you have to specify a third component. Figure 3.12 is described by

$$\begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} \cos(3t) \\ \sin(3t) \\ t \end{pmatrix}.$$

3.3.8 Parametric 3D area plots

Parametric area plots are similar to parametric curves, but they span whole areas in three-dimensional space and thus require functions of two variables (u and v). For the Möbius strip seen in Figure 3.13, you'd use the equation

$$\begin{pmatrix} x \\ y \\ z \end{pmatrix} = 3 \begin{pmatrix} (1 + 0.5v \cos(0.5u)) \cos u \\ (1 + 0.5v \cos(0.5u)) \sin u \\ 0.5v \sin(0.5u) \end{pmatrix}.$$

Chapter 4

Advanced Capabilities

This chapter describes some caveats for using the more advanced computations available in PocketCAS. For examples on how to perform some tasks in the corresponding areas, please refer to the tutorials and the function reference. The function list's categories should help you find the function you are looking for.

4.1 Linear algebra

This section describes some more technical matrix operations. For an overview of basic matrix and vector operations and PocketCAS' linear algebra functions, please refer to the *Linear Algebra* tutorial and function list.

4.1.1 Matrix and vector formats

When using the matrix editor (described in Section 2.3), you don't need to worry about PocketCAS' matrix formats. But if you prefer to enter matrices manually, this section describes the corresponding syntax.

Lists

The basic multi-value type in PocketCAS is a list. It is entered as a sequence of values separated by a comma that's surrounded by square braces. For example, the input

$$[a, b, c]$$

returns a list containing the elements a , b and c . Lists are very similar to arrays in other programming languages, but some linear algebra operations in PocketCAS require you to use vectors.

Matrices and vectors

A *matrix* is simply a list of lists. Each sub-list represents one row of the matrix. For example, the input

$$[[a, b, c], [d, e, f]]$$

will be interpreted as the matrix

$$\begin{bmatrix} a & b & c \\ d & e & f \end{bmatrix}.$$

Row- and column-*Vectors* are matrices consisting of just one row or one column, respectively:

$$[[a, b, c]] \rightarrow [a \quad b \quad c] \qquad [[a], [b], [c]] \rightarrow \begin{bmatrix} a \\ b \\ c \end{bmatrix}.$$

4.1.2 Submatrix access

Say we have defined a matrix

$$A := [[a, b, c], [d, e, f]] \rightarrow \begin{bmatrix} a & b & c \\ d & e & f \end{bmatrix}.$$

$A[j, k]$ returns the entry in the $(j+1)$ -row and the $(k+1)$ -th column. In this example, $A[1, 2]$ would return f .

You can access entire submatrices by specifying ranges:

$$A[0..1, 1..2] \rightarrow \begin{bmatrix} b & c \\ e & f \end{bmatrix}.$$

Note that a one-row or one-column sub-matrix will return a *list* rather than a *vector*:

$$A[0..1, 1] \rightarrow [b, e].$$

4.2 Solving

Please have a look at the solving tutorial for examples on how to solve equations with PocketCAS.

When solving an equation, please make sure that none of the variables occurring in the equation have been assigned a value. Otherwise, the variables might get substituted for their values before the actual solving takes place, which can lead to unexpected results.

To show all numerical solutions in an interval, you can use the `fsolve()` function, passing the interval as its third parameter:

```
fsolve(exp(x) - 2x^2 = 0, x = -10..10)
→ [-0.53983527690282, 1.4879620654982, 2.6178666130668].
```

4.2.1 Curve sketching

Although PocketCAS does not have a complete curve sketching feature, there are ways to easily obtain most of the corresponding information. In cartesian plots, PocketCAS will already show the function's zeroes and extremes (see Section 3.3.1).

In addition, you can obtain these values yourself by finding the zeroes of the function or its derivative, respectively. For example, you could first define a function and its derivative:

$$\begin{aligned}f(x) &:= 2x^3 - 5x^2 + 2x + 1 \\g(x) &:= \text{diff}(f(x))\end{aligned}$$

and then calculate the locations of their zeroes via `solve(f(x) = 0)` and `solve(g(x) = 0)`. Note that `solve()` only returns symbolically computable zeroes, so you might need to use `fsolve()` for some functions instead.

4.3 Scripting

Since version 3, PocketCAS supports the *Script* entry type which allows you to create multi-line entries. This e.g. makes it possible to create small programs with control flow statements. For instance, PocketCAS supports *if-else*-statements and *for*- and *while*-loops with a C-style syntax.

To familiarize yourself with PocketCAS' programming language, please study the examples in the *Script* tutorial.

Note: PocketCAS currently does not support *else if*-statements. You need to create an *else* block and put the new *if* block inside. I.e., the following won't work:

```
if (a) {  
    code;  
} else if (b) {  
    code;  
}
```

Instead, you should write:

```
if (a) {  
    code;  
} else {  
    if (b) {  
        code;  
    }  
}
```