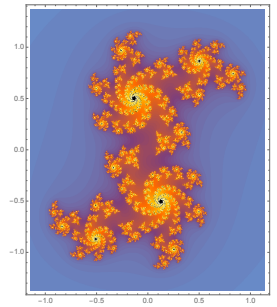


Mathematica: a powerful calculator*

Bahareh Afshari

11th September 2017





Apart from all standard arithmetic calculations, Mathematica can perform a multitude of complex mathematical operations including solving equations, maximising and minimising functions, integration and differentiation, as well as plotting and visualising curves and surfaces. Of course, there are other programs with similar aims such as Maple and Matlab. One advantage of Mathematica is its friendly user interface.

This document contains commands and tips to help you get started with Mathematica. In the introductory lecture you will have also seen

- how to create a notebook
- what cells are and how you can evaluate calculations in separate cells
- how to abort a computation
- what the kernel is and what you need to know about it

1 Getting started: calculations, defining variables and more

To evaluate a calculation you need to press  + .

► Try $1-1$, $2(3+5)$ and $17/7$.

► Try these one after another: $a=2$, $b=3$, $a+b$, $a=4$, $a+b$, $a+b+c$

If you assign different values to the same variable it is always the last assignment that counts. For example, the value for a will be 4 for all the following calculations in our notebook (unless we give a a new value).

In fact, if we open a new notebook and start using a or b we will find out that they already have values 3 and 4 respectively. This is because all calculations happen in the background, in Mathematica's *kernel* and until you quit Mathematica the assignments are all remembered.

*<https://www.wolfram.com/mathematica/>

So, if you want to use the same variable name for other purposes you need to first “free” the variable by using commands such as `Remove[b]` or `Remove[Global`*]`.

► Try `a==2` and `a==4`.

Notice the difference between `a=2` and `a==2`: the former is an assignment (setting the value of `a` to be 2) and the latter is testing whether the value of `a` is equal to 2. We will see another use of `==` later.

2 Built-in functions

Mathematica has all the functions you can dream of. An easy place to find notation for them is `Palettes` \leftrightarrow `Basic Math Assistant`.

► Find the Mathematica version of the following: $\sqrt[3]{125}$, $\log e$, $\sin \pi$

Note that,

1. all built-in functions and constants start with a capital letter;
2. function arguments are enclosed in `[]` and *not* `()` – parentheses are only used for grouping.

Some special functions There are some special Mathematica functions that are handy to know:

► Try `N[17/7]` and `N[17/7,30]`. `N` means ‘numerical’ and returns the numerical value of its argument. The optional second argument is the number of digits of precision to use.

► `Expand[(x+1)^2]`. Mathematica does not expand expressions until it needs to. The function `Expand` opens out products and powers.

3 Defining your own functions

A piece of advice: when you want to use something that you haven’t seen an example of before, don’t guess! It can go terribly wrong and take you a long time to figure out the mistake.

► Try `r[x]=x+4` and then `r[1]`. Can you figure out what went wrong?

It is very easy to find out the correct way. In the menu, go to `Help` \leftrightarrow `Wolfram Documentation`, search for “defining functions”. Then copy an example into your notebook and evaluate.

► Try `f[x_]:=x^2` and then `f[3]` and `f[a+b+c]`.

► Try `r[x_]:=x+4` and then `r(5)` and `r[5]`. What is going on?

There are three important things to remember about the syntax when defining functions: the ‘blank’ `_`, square brackets `[]`, and ‘colon equals’ `:=`.

► Try `?f` and `?r`

If we are interested in calculating an expression on a particular value, there is no need to define a function as we can use the substitution operation `/.`

► Try `(x^2 + 2) /. x -> 15`

► Try `y=(x+1)(x-2)` and then `y /. x -> b-1`

4 Defining and using lists

► Try `L={3,5,hello,11}`, then `L+2`

► Try `Range[6]`

► Try `mylist=Range[2,50,2]`

► Try `mylist==L` and `mylist[[6]]`

Enter `?Range` or look up `Range` in the documentation to read more about it.

5 Solving equations

Another important use for `==` is in solving equations.

► Try `Solve[5x == 25, x]`

Look up ‘`Solve`’ in the documentation. Remember, you don’t need to read all the text there – just skim through and look for what you are after.

► Try `Solve[{x + y == 2, y - x == 1}, {x,y}]`

6 Plotting data

No matter what your data is, whether it is a list, an (explicit) function or parametric, Mathematica has the tools to visualise it. Below we see some examples.

6.1 Plotting lists

In Mathematica, a pair (x, y) is represented as $\{x, y\}$.

► Try `data = { {2,4} , {3,8} , {4,15} } and ListPlot[data]`

► Try `ListPlot[data , PlotJoined -> True]`

Options such as `PlotJoined` can be easily found in the documentation.

6.2 Plotting functions

► Try `Plot[Tan[x] , { x , 0 , 6 Pi }]`

► Try `Plot[Tan[x] , { x , 0 , 6 Pi } , PlotRange -> { {-5,10} , {-4,4} }]`

`PlotRange` is a useful option!

6.3 Plotting two or more functions at the same time

► Try `Plot[{ Tan[x] , Sin[x] } , { x , 0 , 6 Pi }]`. See `?Plot` for the general form and further options.

Keep a close eye on the syntax! For instance, if you leave out the braces in the above example you will get an error which is hard to detect. Copying an existing example is a good strategy.

► Try `Plot3D[{ x^2 + y^2 } , { x , -2 , 2 } , { y , -2 , 2 }]`

All the examples in the Functions and Equations lecture notes were plotted using the above tools. Try some of them for yourself.

7 Fitting data to lines, curves, ...

► Try `myline = Fit[data , { 1 , x } , x] and then ?myline`

► Try `mycurve = Fit[data , { 1 , x , x^2 } , x]`

► `Plot[{myline , mycurve}, {x,0,5}]`

If you want to combine `ListPlot` and `Plot`, `Show` is your friend:

► `Show[ListPlot[data] , Plot[{ myline , mycurve } , {x, 0, 5}]]`

8 Final words

1. After you quit Mathematica all variable assignments, function definitions, etc, are erased. So next time you open your notebook, if you need them you must evaluate the cells which contain the definitions.

- 2.** If you start an evaluation that takes too long or has an infinite loop, it can be aborted from the menu `Evaluation` \leftrightarrow `Abort Evaluation`.
- 3.** If you can't find the mistakes in your code, try reloading Mathematica as this resets the kernel.