Introductory Tutorial to *Mathematica* 8

Author: Harald Höller
version: English version v1.0
last modified: 12.01.2012

This introductory tutorial to *Mathematica* 8 is designed as interactive course material and should ideally be worked through together with students at PCs. Each student should progress sequentially through all Input/Output. Duration: approx. 90min.

Getting Started with Mathematica

Mathematica Basics

■ *Some Basic Remarks*

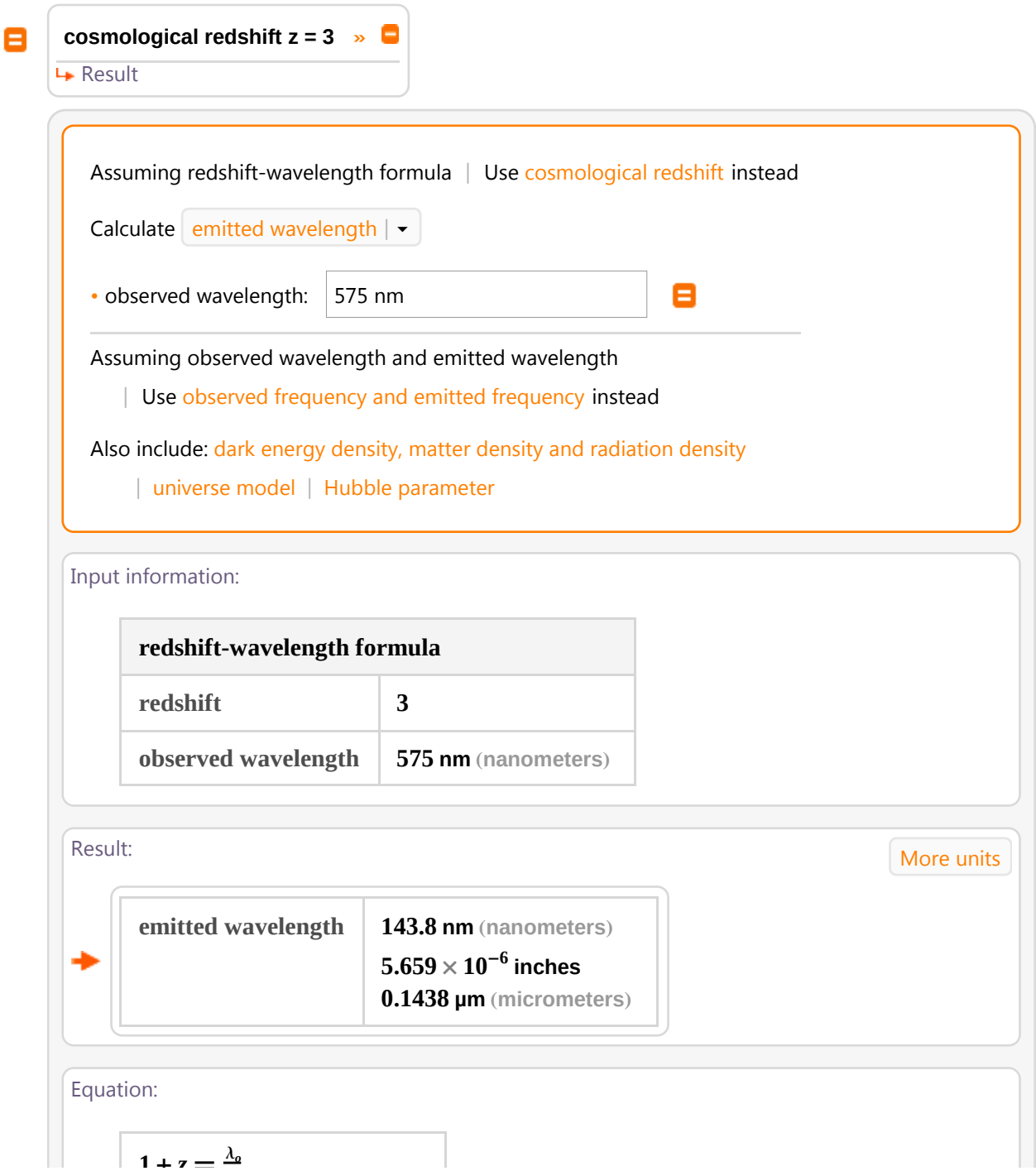
Mathematica allows analytical, numerical and grafical calculations. *Mathematica* offers a relatively simple and instructional logical syntax. Most of the known functions do have a syntactically similar correspondent in *Mathematica*. The argument of a function always is to set inside square brackets (e.g. Sin[x]). If you operate on some mathematical expression, also the operation is executed onto brackets and the arguments are separated by commas (e.g. Integrate[f[x],x]).

■ *Data types*

The standard data type, the Notebook (.nb) provides an interactive interface in which calculations are possible. Under *File -> Save As* there are some further data formats to which the notebooks can be exported (.tex, .html, .txt, .pdf). Also single expressions can be copied as LaTeX code and be pasted into a .tex-file. Notebooks can be opened by *Mathematica* (interactive) or the *MathReader* (read only) respectively the *Wolfram CDF Player* which is available as free internet browser plugin.
The forward- and backward-compatibility is given partially, which means that notebooks created in *Mathematica* 7 will usually run in *Mathematica* 8, the opposite direction will rather cause problems.

■ *Wolfram|Alpha*

Wolfram|Alpha is a net-based "frontend" for *Mathematica* usage with a huge database of knowledge. Describing as "scientific Google" does not really cover its whole power. Either you access it via the website www.wolframalpha.com or there is even an integration to *Mathematica*. Beginning an Input with the "=" sign is interpreted as Wolfram|Alpha query. E.g: "= cosmological redshift z=3"



$1 + z = \frac{\lambda_o}{\lambda_e}$

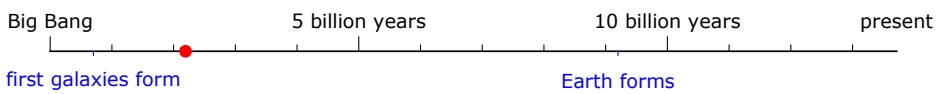
λ_e	**emitted wavelength**
z	**redshift**
λ_o	**observed wavelength**

Cosmological results: More

time ago (lookback time)	**11.5 billion years**
time since big bang	**2.19 billion years**
distance (comoving)	**21.1 billion ly** (light years) **6480 Mpc** (megaparsecs) **2×10^{23} km** (kilometers) **1.24×10^{23} miles**
fraction of total observable radius	**0.453**
scale factor	**0.25** × current value
epoch	**matter dominated, post-recombination**
radiation temperature	**10.9 K** (kelvins)

(based on 5–year WMAP data and Lambda–CDM model; current universe age: 13.7 billion years)

Timeline:



emitted wavelength	**143.8 nm** (nanometers) **5.659×10^{-6} inches** **0.1438 μm** (micrometers)

■ *Input/Output I*

The input of commands in *Mathematica* is done via the Enter key (not Return), which is either located at the numeric keypad or it can be typed in as Shift + Return. Each Output and Input is numbered by *Mathematica*.

```
3
```

```
3
```

In this way one can refer to previous In- and Output

```
In[2] + Out[2]
```

```
6
```

respectively one can refer to the last output via %.

```
%
```

```
6
```

■ *Help / Documentation Center*

Mathematica provides a compresehsive help and documentation center which gets completed by the online-documentation. By entering the F1 key, the documentation center is opened; one can even mark an expression in the notebook and will get to the help entry for this term via F1.
For a brief summary of syntax usage and functionality of a function, one can also just enter question mark + function to get an overwiev.

```
? Integrate
```

Integrate[f, x] gives the indefinite integral $\int f\, d\, x$.

Integrate[f, {x, x_{min}, x_{max}}] gives the definite integral $\int_{x_{min}}^{x_{max}} f\, d\, x$.

Integrate[f, {x, x_{min}, x_{max}}, {y, y_{min}, y_{max}}, ...] gives the multiple integral $\int_{x_{min}}^{x_{max}} d\, x \int_{y_{min}}^{y_{max}} d\, y \ldots f$. ≫

With ?xyz* one gets a list of functions starting with xyz.

```
?Integ*
```

▼ System`

Integer	IntegerExponent	IntegerPart	IntegerQ	IntegerString	Integrate
IntegerDigits	IntegerLength	IntegerPartitions	Integers	Integral	

■ *The Kernel*

Mathematica "remembers" all definitions, In- and Outputs inside a Notebook. As described some lines before, one can refer to previous terms easily. As long as the Kernel is running, all these definitions are available.

```
?Random
```

Random[] gives a uniformly distributed pseudorandom Real in the range 0 to 1.
Random[*type*, *range*] gives a pseudorandom number of the specified type, lying in the specified range. Possible types are: Integer, Real and Complex. The default range is 0 to 1. You can give the range {*min*, *max*} explicitly; a range specification of *max* is equivalent to {0, *max*}. ≫

```
a = Random[Integer, 100]
b = Random[Integer, 100]
c = Random[Integer, 100]
```

```
22
```

```
53
```

```
3
```

```
a + b + c
```

```
78
```

If you want to delete entires in this memory of definitions, the Clear[Arguments] command can be used. If you want to clear all kernel memory, i.e. kill and restart the kernel, then go to *Evaluation -> Quit Kernel -> Local*. This also terminates running calculations which comes in handy, when you realize that a calculation seems to have got stuck and you don't want to wait any longer.

```
?Clear
```

Clear[$symbol_1$, $symbol_2$, …] clears values and definitions for the $symbol_i$.
Clear["$form_1$", "$form_2$", …] clears values and
definitions for all symbols whose names match any of the string patterns $form_i$. ≫

```
Clear[a, b, c]
```

```
a + b + c
```

```
a + b + c
```

If one wishes not to quit ther kernel but delete all previous definitions, (of all currently opened *Mathematica*-Notebooks) the syntax is as follows.

```
Clear["Global`*"]
```

■ *Comments*

Comments (i.e. non to be executed, read-only text parts) inside Input-lines can by typed inside round brackets and stars.

```
Sqrt[2] ==
 (* Square root of 2 can also be typed via "Ctrl + 2": *) √2
```

```
True
```

■ *Shortcuts*

There is a number of shortcuts and hotkeys in order to simplify typing formulas to the notebook. Some examples:

```
a_i (* Subscript via "Ctrl + -" *);
1/2 (* Fraction lines via "Ctrl + /" *);
∫ (*"Esc + int + Esc" *)f[x] dx (* "Esc + dd + Esc" *);
∞ (* "Esc + inf + Esc" *);
α (* Also greek letters via "Esc + Letter + Esc" *);
```

■ *Syntax Highlighting*

When typing formulae in a *Mathematica* notebook, syntax highlighting is very convenient. Known Functions, trailing brackets, arguments etc. are recognoized and highlighted; either as positive feedback that everything is alright or - usually even more helpful - as negative feedback that there is some syntactic problem. In case the syntax is faulty, klick on the plus-sign on the right hand side of the input line to get more information about what might be wrong.

```
Solve[Sin[x_ + 3 + Cos[x_] = 1, y]
```

Symbolic Calculus in *Mathematica* - Basics

■ *Input/Output II*

The decimal separator is the dot (.) verwendet, not the comma. A space character is interpreted as scalar multiplicator (“times”) and replaced by a cross.

```
3.14 × 3.14
```

```
9.8596
```

■ *Definitions and Assignments*

The simplest form of assigning a right hand side to a left hand side was already presented, namely the equal sign.

```
?=
```

lhs = *rhs* evaluates *rhs* and assigns the result
to be the value of *lhs*. From then on, *lhs* is replaced by *rhs* whenever it appears.
{l_1, l_2, …} = {r_1, r_2, …} evaluates the r_i, and assigns the results to be the values of the corresponding l_i. ≫

```
d = 1
```

```
1
```

One can surpress the otput of an assignment by adding a colon. This assigment is evaluated not until the definition is being used somewhere later in the Notebook. This assignment is especially useful when the right hand side is somewhat elongate and evaluation costs computation time.

```
? :=
```

lhs := *rhs* assigns *rhs* to be the delayed value of *lhs*. *rhs* is maintained in an unevaluated form. When *lhs* appears, it is replaced by *rhs*, evaluated afresh each time. ≫

```
f := 1
```

■ *Functions + Pattern*

One main powerfulness of CAS such as *Mathematica* is symbolic calculus, i.e. working with functions. There is one little syntax peculiarity of *Mathematica* concerning this topic. Objects with arguments in the form f[x] are static; if you want the argument to be variable, then definition of the function must add an underscore (see *Mathematica* documentation center under keyword Pattern) to each argument that should be a variable.

DONT :

```
h[x] := Sin[x]
```

```
h[1]
```

```
h[1]
```

DO :

```
g[x_] := Sin[x]
```

```
g[1]
```

```
Sin[1]
```

As you can see, *Mathematica* does not necessarily output numerical values. In this case, the result is an irrational number and one needs to specify further if and what numerical outoput is wanted.

■ *Input/Output III*

In order to force numerical output, one needs to type N[Argument,Digits].

```
N[g[1], 100]
```

```
0.84147098480789650665250232163029899962256306079837106567275170999
 19191040439123966894863974354305269 59
```

There is also the possibility to append functions and objetcs that operate on the whole expression by a double slash.

```
g[1] // N
```

```
0.841471
```

■ *Basic Calculus (Scalars)*

An addition of objetcs (scalars as well as vectors and matrices) is typed - as expected - via the plus (+) key, subtraction via minus (-). Scalar multiplication can be set via the star (*) or a blank character, division is typed viaone slash.

```
2 * 2
```

```
4
```

```
2 × 2
```

```
4
```

```
2 / 2
```

```
1
```

Nested expressions are typed as “with pen and paper” using round brackets.

```
(1 + (4 - 3) * 2) / (1 + 1) + 1
```

$$\frac{5}{2}$$

```
Clear["Global`*"]
(* please enter this input to delete all previous definitions
 which would partially conflict with the upcoming section *)
```

■ *Basic Commands - I/O III*

Until now we have seen *Mathematica* mainly as somewhat sophisticated calculator. However its true power abounds when applying such a computer algebra system to complex problems such as integrals, differential equations or algebraic system of equations that are hardly or even not solvable at all with pen and paper.

■ E.g.: Differentiation

Just to get used to the syntax we regard a simple example. The differentiation symbol is the capital D:

```
?D
```

D[f, x] gives the partial derivative $\partial f/\partial x$.
D[f, {x, n}] gives the multiple derivative $\partial^n f/\partial x^n$.
D[f, x, y, ...] differentiates f successively with respect to x, y, ...
D[f, {{x_1, x_2, ...}}] for a scalar f gives the vector derivative $(\partial f/\partial x_1, \partial f/\partial x_2, ...)$.
D[f, {*array*}] gives a tensor derivative. ≫

When we compute the first derivative of a general function F[x], we get the output F'[x]. As described in the *Mathematica* documentation, the arguments (function and variable) of this operation are separated by a comma.

```
D[F[x], x]
```

```
F'[x]
```

We have to further specify the function to be derived in order to get a calculated output of course. As an example we define a function G in and compute ther first partial derivative with respecto to one variable,

```
G[x_, y_] := x * y
```

```
D[G[x, y], y]
```

```
x
```

the second partial derivative

```
D[G[x, y], {x, 2}]
```

```
0
```

or subsequent derivatives with respect to a list of variables.

```
D[G[x, y], x, y]
```

```
1
```

■ *Simplify und Expand*

```
?Simplify
```

Simplify[*expr*] performs a sequence of algebraic and other transformations on *expr*, and returns the simplest form it finds.
Simplify[*expr*, *assum*] does simplification using assumptions. ≫

```
?FullSimplify
```

FullSimplify[*expr*] tries a wide range of transformations on
expr involving elementary and special functions, and returns the simplest form it finds.
FullSimplify[*expr*, *assum*] does simplification using assumptions. ≫

```
?Expand
```

Expand[*expr*] expands out products and positive integer powers in *expr*.
Expand[*expr*, *patt*] leaves unexpanded any parts of *expr* that are free of the pattern *patt*. ≫

■ E.g.: Polynom

```
x * (x + 1)^3 + (x^2 - 3 * x)^5 + x + 1 + x^2 + (x - x^2 + (x - 1)^2)^2
```

$$1 + x + x^2 + x\,(1 + x)^3 + \left((-1 + x)^2 + x - x^2\right)^2 + \left(-3\,x + x^2\right)^5$$

```
FullSimplify[%]
```

$$2 + x^2 \left(5 + x \left(3 + x \left(1 + (-3 + x)^5 x\right)\right)\right)$$

```
Expand[%]
```

$$2 + 5\,x^2 + 3\,x^3 + x^4 - 243\,x^5 + 405\,x^6 - 270\,x^7 + 90\,x^8 - 15\,x^9 + x^{10}$$

Plots / Graphics / Figures - Basics

■ *Plot*

A major scientific tool in *Mathematica* is the manifold of plotting routines that help you to depict mathematical results grafically. The basic syntax for 2D graphs (as they are of course also done by muich simpler programs like gnuplot) is Plot[Argument, {Variables, Boundaries}]. These boundaries define the abscissa, the ordinate is scaled with the additional argument, namely PlotRange -> {Boundaries}.

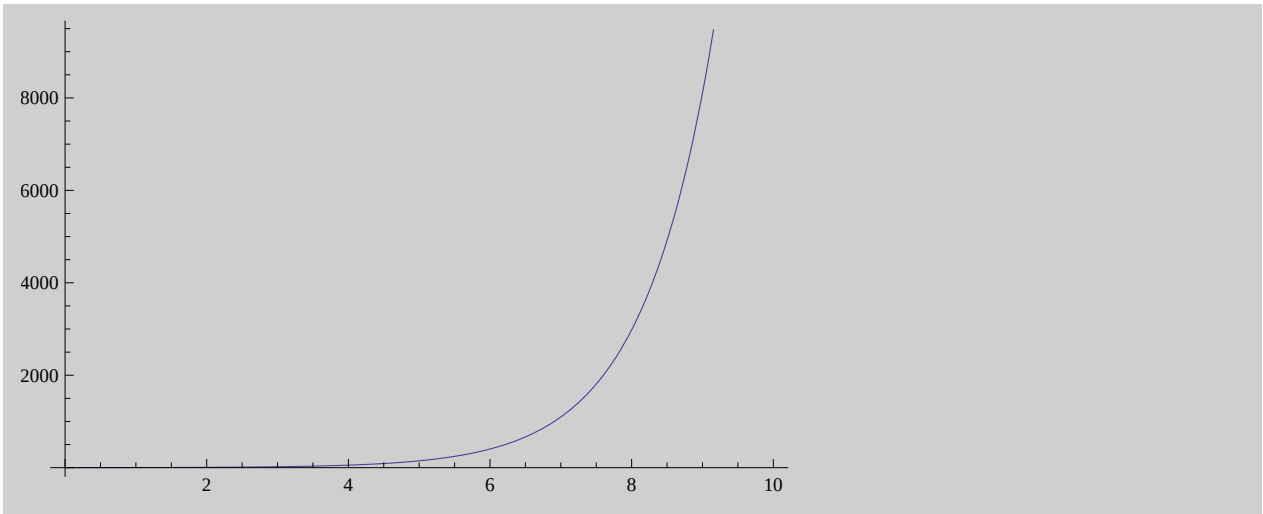
```
?Plot*
```

▼ System`

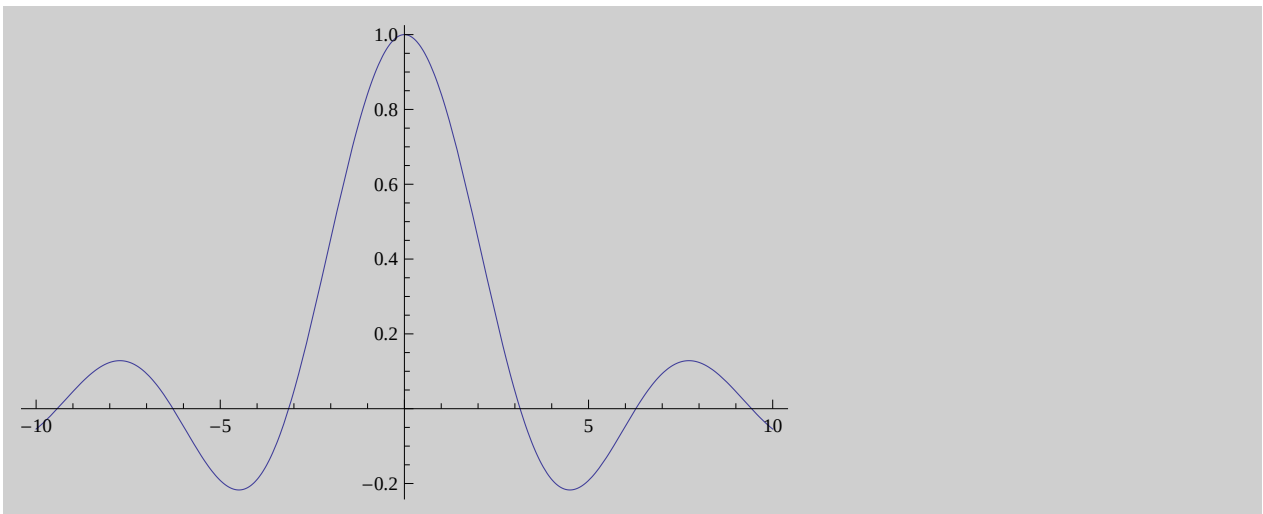
Plot	Plot3Matrix	PlotJoined	PlotLayout	PlotPoints	PlotRange-Clipping	PlotRegion
Plot3D	PlotDivisio-n	PlotLabel	PlotMarke-rs	PlotRange	PlotRange-Padding	PlotStyle

■ *Plot Functions*

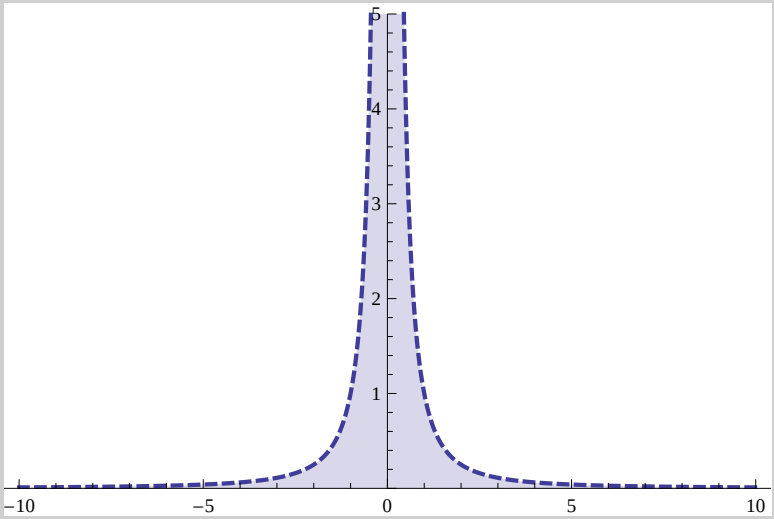
```
Plot[Exp[x], {x, 0, 10}]
```




```
Plot[Sin[x] / x, {x, -10, 10}]
```



```
Plot[1 / x^2, {x, -10, 10}, PlotRange → {0, 5},
 PlotStyle -> {Thick, Dashed}, Background → White, Filling → Automatic]
```



■ *Dynamic Graphics*

The two major commands for interactice plotting are:

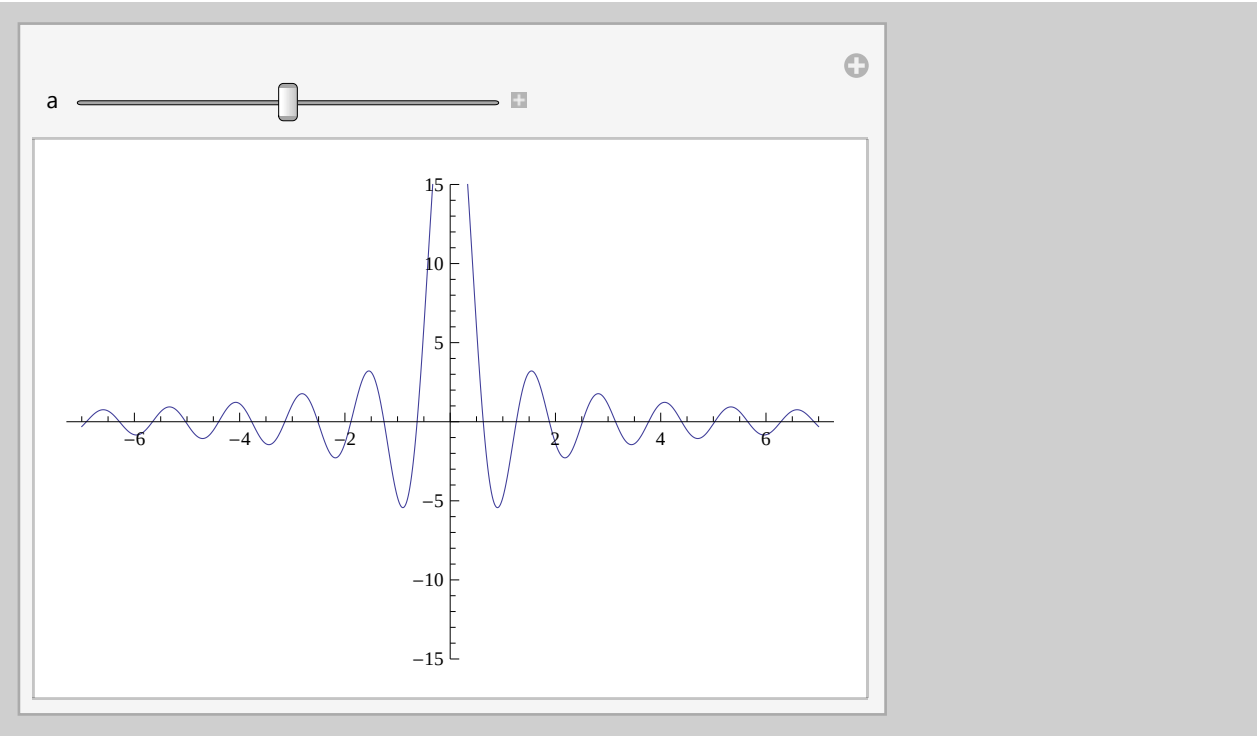
```
? Dynamic
```

Dynamic[*expr*] represents an object that displays as the dynamically updated current value of *expr*. If the displayed form of Dynamic[*expr*] is interactively changed or edited, an assignment *expr* = *val* is done to give *expr* the new value *val* that corresponds to the displayed form.
Dynamic[*expr*, None] does not allow interactive changing or editing.
Dynamic[*expr*, *f*] continually evaluates *f*[*val*, *expr*] during interactive changing or editing of *val*.
Dynamic[*expr*, {*f*, f_{end}}] also evaluates f_{end}[*val*, *expr*] when interactive changing or editing is complete.
Dynamic[*expr*, {f_{start}, *f*, f_{end}}] also evaluates f_{start}[*val*, *expr*] when interactive changing or editing begins. ≫

```
? Manipulate
```

Manipulate[*expr*, {*u*, u_{min}, u_{max}}] generates a version of *expr* with controls added to allow interactive manipulation of the value of *u*.
Manipulate[*expr*, {*u*, u_{min}, u_{max}, *du*}] allows the value of *u* to vary between u_{min} and u_{max} in steps *du*.
Manipulate[*expr*, {{*u*, u_{init}}, u_{min}, u_{max}, …}] takes the initial value of *u* to be u_{init}.
Manipulate[*expr*, {{*u*, u_{init}, u_{lbl}}, …}] labels the controls for *u* with u_{lbl}.
Manipulate[*expr*, {*u*, {u_1, u_2, …}}] allows *u* to take on discrete values u_1, u_2, ….
Manipulate[*expr*, {*u*, …}, {*v*, …}, …] provides controls to manipulate each of the *u*, *v*, ….
Manipulate[*expr*, c_u –> {*u*, …}, c_v –> {*v*, …}, …]
links the controls to the specified controllers on an external device. ≫

```
Manipulate[Plot[a * Sin[a * x] / x,
  {x, -7, 7}, PlotRange → {-15, 15}], {a, 0, 10, 1}]
```



■ *Graphics und Show*

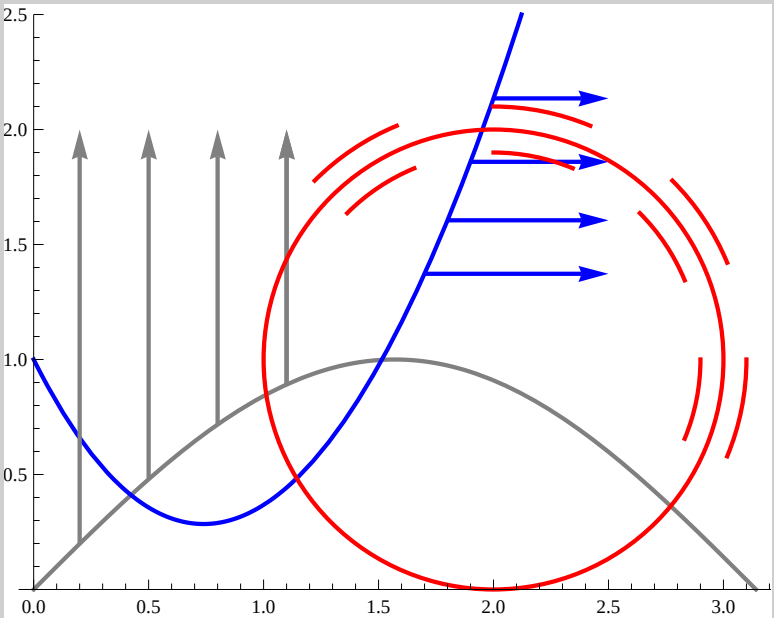
```
? Graphics
```

Graphics[*primitives, options*] represents a two–dimensional graphical image. ≫

```
? Show
```

Show[*graphics, options*] shows graphics with the specified options added.
Show[g_1, g_2, …] shows several graphics combined. ≫

```
Show[Plot[Sin[x], {x, 0, Pi},
  AspectRatio → Automatic, PlotStyle → {Thick, Gray},
  Background → White, PlotRange → {0, 2.5}],
 Plot[Exp[-x] - x + x^2, {x, 0, Pi}, AspectRatio → Automatic,
  PlotStyle → {Thick, Blue},
  Background → White, PlotRange → {0, 2.5}],
 Graphics[{Thick, Gray, Arrow[{{0.2, Sin[0.2]}, {0.2, 2}}]}],
 Graphics[{Thick, Gray, Arrow[{{.5, Sin[.5]}, {.5, 2}}]}],
 Graphics[{Thick, Gray, Arrow[{{.8, Sin[.8]}, {.8, 2}}]}],
 Graphics[{Thick, Gray, Arrow[{{1.1, Sin[1.1]}, {1.1, 2}}]}],
 Graphics[{Thick, Gray, Arrow[{{1.1, Sin[1.1]}, {1.1, 2}}]}],
 Graphics[{Thick, Blue, Arrow[{{1.7, Exp[-1.7] - 1.7 + 1.7^2},
     {2.5, Exp[-1.7] - 1.7 + 1.7^2}}]}],
 Graphics[{Thick, Blue, Arrow[{{1.8, Exp[-1.8] - 1.8 + 1.8^2},
     {2.5, Exp[-1.8] - 1.8 + 1.8^2}}]}],
 Graphics[{Thick, Blue, Arrow[{{1.9, Exp[-1.9] - 1.9 + 1.9^2},
     {2.5, Exp[-1.9] - 1.9 + 1.9^2}}]}],
 Graphics[{Thick, Blue, Arrow[{{2, Exp[-2] - 2 + 2^2},
     {2.5, Exp[-2] - 2 + 2^2}}]}],
 Graphics[{Thick, Red, Circle[{2, 1}]}],
 Graphics[{Thick, Red, Circle[{2, 1}, 1.1, {-Pi / 8, 0}]}],
 Graphics[{Thick, Red, Circle[{2, 1}, 1.1, {Pi / 8, 2 * Pi / 8}]}],
 Graphics[{Thick, Red, Circle[{2, 1}, 1.1, {3 * Pi / 8, 4 * Pi / 8}]}],
 Graphics[{Thick, Red, Circle[{2, 1}, 1.1, {5 * Pi / 8, 6 * Pi / 8}]}],
 Graphics[{Thick, Red, Circle[{2, 1}, .9, {-Pi / 8, 0}]}],
 Graphics[{Thick, Red, Circle[{2, 1}, .9, {Pi / 8, 2 * Pi / 8}]}],
 Graphics[{Thick, Red, Circle[{2, 1}, .9, {3 * Pi / 8, 4 * Pi / 8}]}],
 Graphics[{Thick, Red, Circle[{2, 1}, .9, {5 * Pi / 8, 6 * Pi / 8}]}]
]
```



Linear Algebra - Calculus on Lists, Vectors, Matrices etc.

On The Difference Between Lists and Vectors

■ *Handling Vectors, Matrices and Tensors*

Whenever working with CAS respectively even more basic level of data processing such as using common programming languages, one has to keep in mind that a list of entries does not necessarily make a vector, nor does a two dimensional list make a tensor etc. From algebra we know that all tensorial quantities are only defined properly when also defining a base. This standard base is - as expected - the cartesian n-dimensional unit base if nothing else is specified.
A list of functions, numbers or otherwise entries is input via curly brackets and the comma as separator.

```
vec = {a, b, c, d}
```

```
{a, b, c, d}
```

```
? Dimensions
```

Dimensions[*expr*] gives a list of the dimensions of *expr*.
Dimensions[*expr*, *n*] gives a list of the dimensions of *expr* down to level *n*. ≫

```
Dimensions[vec]
```

```
{4}
```

Of course this concept can be used to arbitrarily interlace lists of lists etc.

```
tensor := {{{a, b, c}, {a, b, c}, {a, b, c}, {a, b, c}},
  {{a, b, c}, {a, b, c}, {a, b, c}, {a, b, c}}}
```

```
Dimensions[tensor]
```

```
{2, 4, 3}
```

- **E.g.: The Identity Matrix**

```
? IdentityMatrix
```

IdentityMatrix[*n*] gives the $n \times n$ identity matrix. ≫

```
M := IdentityMatrix[10]
```

```
Dimensions[M]
```

```
{10, 10}
```

■ E.g.: Automatic Filling of Lists

```
? Table
```

Table[*expr*, {i_{max}}] generates a list of i_{max} copies of *expr*.
Table[*expr*, {*i*, i_{max}}] generates a list of the values of *expr* when *i* runs from 1 to i_{max}.
Table[*expr*, {*i*, i_{min}, i_{max}}] starts with $i = i_{min}$.
Table[*expr*, {*i*, i_{min}, i_{max}, *di*}] uses steps *di*.
Table[*expr*, {*i*, {i_1, i_2, …}}] uses the successive values i_1, i_2, ….
Table[*expr*, {*i*, i_{min}, i_{max}}, {*j*, j_{min}, j_{max}}, …] gives a nested list. The list associated with *i* is outermost. ≫

```
H = Table[1 / (i + j - 1), {i, 1, 15}, {j, 1, 15}];
% // MatrixForm
```

$$\begin{pmatrix}
1 & \frac{1}{2} & \frac{1}{3} & \frac{1}{4} & \frac{1}{5} & \frac{1}{6} & \frac{1}{7} & \frac{1}{8} & \frac{1}{9} & \frac{1}{10} & \frac{1}{11} & \frac{1}{12} & \frac{1}{13} & \frac{1}{14} & \frac{1}{15} \\
\frac{1}{2} & \frac{1}{3} & \frac{1}{4} & \frac{1}{5} & \frac{1}{6} & \frac{1}{7} & \frac{1}{8} & \frac{1}{9} & \frac{1}{10} & \frac{1}{11} & \frac{1}{12} & \frac{1}{13} & \frac{1}{14} & \frac{1}{15} & \frac{1}{16} \\
\frac{1}{3} & \frac{1}{4} & \frac{1}{5} & \frac{1}{6} & \frac{1}{7} & \frac{1}{8} & \frac{1}{9} & \frac{1}{10} & \frac{1}{11} & \frac{1}{12} & \frac{1}{13} & \frac{1}{14} & \frac{1}{15} & \frac{1}{16} & \frac{1}{17} \\
\frac{1}{4} & \frac{1}{5} & \frac{1}{6} & \frac{1}{7} & \frac{1}{8} & \frac{1}{9} & \frac{1}{10} & \frac{1}{11} & \frac{1}{12} & \frac{1}{13} & \frac{1}{14} & \frac{1}{15} & \frac{1}{16} & \frac{1}{17} & \frac{1}{18} \\
\frac{1}{5} & \frac{1}{6} & \frac{1}{7} & \frac{1}{8} & \frac{1}{9} & \frac{1}{10} & \frac{1}{11} & \frac{1}{12} & \frac{1}{13} & \frac{1}{14} & \frac{1}{15} & \frac{1}{16} & \frac{1}{17} & \frac{1}{18} & \frac{1}{19} \\
\frac{1}{6} & \frac{1}{7} & \frac{1}{8} & \frac{1}{9} & \frac{1}{10} & \frac{1}{11} & \frac{1}{12} & \frac{1}{13} & \frac{1}{14} & \frac{1}{15} & \frac{1}{16} & \frac{1}{17} & \frac{1}{18} & \frac{1}{19} & \frac{1}{20} \\
\frac{1}{7} & \frac{1}{8} & \frac{1}{9} & \frac{1}{10} & \frac{1}{11} & \frac{1}{12} & \frac{1}{13} & \frac{1}{14} & \frac{1}{15} & \frac{1}{16} & \frac{1}{17} & \frac{1}{18} & \frac{1}{19} & \frac{1}{20} & \frac{1}{21} \\
\frac{1}{8} & \frac{1}{9} & \frac{1}{10} & \frac{1}{11} & \frac{1}{12} & \frac{1}{13} & \frac{1}{14} & \frac{1}{15} & \frac{1}{16} & \frac{1}{17} & \frac{1}{18} & \frac{1}{19} & \frac{1}{20} & \frac{1}{21} & \frac{1}{22} \\
\frac{1}{9} & \frac{1}{10} & \frac{1}{11} & \frac{1}{12} & \frac{1}{13} & \frac{1}{14} & \frac{1}{15} & \frac{1}{16} & \frac{1}{17} & \frac{1}{18} & \frac{1}{19} & \frac{1}{20} & \frac{1}{21} & \frac{1}{22} & \frac{1}{23} \\
\frac{1}{10} & \frac{1}{11} & \frac{1}{12} & \frac{1}{13} & \frac{1}{14} & \frac{1}{15} & \frac{1}{16} & \frac{1}{17} & \frac{1}{18} & \frac{1}{19} & \frac{1}{20} & \frac{1}{21} & \frac{1}{22} & \frac{1}{23} & \frac{1}{24} \\
\frac{1}{11} & \frac{1}{12} & \frac{1}{13} & \frac{1}{14} & \frac{1}{15} & \frac{1}{16} & \frac{1}{17} & \frac{1}{18} & \frac{1}{19} & \frac{1}{20} & \frac{1}{21} & \frac{1}{22} & \frac{1}{23} & \frac{1}{24} & \frac{1}{25} \\
\frac{1}{12} & \frac{1}{13} & \frac{1}{14} & \frac{1}{15} & \frac{1}{16} & \frac{1}{17} & \frac{1}{18} & \frac{1}{19} & \frac{1}{20} & \frac{1}{21} & \frac{1}{22} & \frac{1}{23} & \frac{1}{24} & \frac{1}{25} & \frac{1}{26} \\
\frac{1}{13} & \frac{1}{14} & \frac{1}{15} & \frac{1}{16} & \frac{1}{17} & \frac{1}{18} & \frac{1}{19} & \frac{1}{20} & \frac{1}{21} & \frac{1}{22} & \frac{1}{23} & \frac{1}{24} & \frac{1}{25} & \frac{1}{26} & \frac{1}{27} \\
\frac{1}{14} & \frac{1}{15} & \frac{1}{16} & \frac{1}{17} & \frac{1}{18} & \frac{1}{19} & \frac{1}{20} & \frac{1}{21} & \frac{1}{22} & \frac{1}{23} & \frac{1}{24} & \frac{1}{25} & \frac{1}{26} & \frac{1}{27} & \frac{1}{28} \\
\frac{1}{15} & \frac{1}{16} & \frac{1}{17} & \frac{1}{18} & \frac{1}{19} & \frac{1}{20} & \frac{1}{21} & \frac{1}{22} & \frac{1}{23} & \frac{1}{24} & \frac{1}{25} & \frac{1}{26} & \frac{1}{27} & \frac{1}{28} & \frac{1}{29}
\end{pmatrix}$$

```
? Subscript
```

Subscript[*x*, *y*] is an object that formats as x_y.
Subscript[*x*, y_1, y_2, …] formats as $x_{y_1,y_2,\ldots}$. ≫

```
Table[Subscript[m, i, j], {i, 3}, {j, 3}] // MatrixForm
```

$$\begin{pmatrix} m_{1,1} & m_{1,2} & m_{1,3} \\ m_{2,1} & m_{2,2} & m_{2,3} \\ m_{3,1} & m_{3,2} & m_{3,3} \end{pmatrix}$$

Coordinate Systems + Vector Calculus

Since the topic of coordinate systems is especially interesting in physical and astrophysical applications, we want to take a look at the implemented functionalities in *Mathematica*. A number of packages in *Mathematica* are not loaded into the Kernel by default (which makes the startup faster) but have to be loaded on demand. The *VectorAnalysis* package is one of those - other exaples can be found here: tutorial/PackagesForSymbolicMathematics.

■ *The VectorAnalysis Package*

Loading the package

```
Needs["VectorAnalysis`"]
```

enables a number of additional commands. E.g. with

```
CoordinateSystem
```

```
Cartesian
```

one can inquire the momentarily used coordinate system. As mentioned before, this is cartesian by default. When we change to a spherical coordinate system (remark: “American” coordinate ranges by default)

```
SetCoordinates[Spherical[r, θ, ϕ]]
```

```
Spherical[r, θ, ϕ]
```

and compute the divergence of the unit vector in r-direction, (1,0,0), we get the well known result.

```
Div[{1, 0, 0}]
```

$$\frac{2}{r}$$

```
JacobianMatrix[Spherical] // MatrixForm
```

$$\begin{pmatrix} \text{Cos}[\phi]\,\text{Sin}[\theta] & r\,\text{Cos}[\theta]\,\text{Cos}[\phi] & -r\,\text{Sin}[\theta]\,\text{Sin}[\phi] \\ \text{Sin}[\theta]\,\text{Sin}[\phi] & r\,\text{Cos}[\theta]\,\text{Sin}[\phi] & r\,\text{Cos}[\phi]\,\text{Sin}[\theta] \\ \text{Cos}[\theta] & -r\,\text{Sin}[\theta] & 0 \end{pmatrix}$$

Just as further example for some functionalities of that package, we calculate the unit volume of a cylindre with radius and height one by the differential geometric rule as the integral over the Jacobian determinant.

```
SetCoordinates[Cylindrical[r, θ, z]]
```

```
Cylindrical[r, θ, z]
```

```
Integrate[JacobianDeterminant[Cylindrical],
 {r, 0, 1}, {θ, 0, 2 * π}, {z, 0, 1}]
```

```
π
```

Just to check, we compute the divergence of the unit vector in x-direction for cartesian coordinates

```
SetCoordinates[Cartesian]
```

```
Cartesian[Xx, Yy, Zz]
```

```
Div[{1, 0, 0}]
```

```
0
```

which vanishes of course.

■ *Scalar Product, Cross Product*

Even without the *VectorAnalysis* package activated, there are vector commands loaded by default, such as the scalar (inner) product of two vectors, input via the dot (.).

```
vec = {a, b, c}
```

```
{a, b, c}
```

```
vec.vec
```

```
a^2 + b^2 + c^2
```

With the star * the entries of the list are multiplied pairwise.

```
vec * vec
```

```
{a^2, b^2, c^2}
```

The cross profuct (vector product) is set with the command Cross.

```
Cross[vec, vec]
```

```
{0, 0, 0}
```

Since a vector is parallel to itself, the cross product has to vanish.

```
Cross[{1, 0, 0}, {0, 1, 0}]
```

```
{0, 0, 1}
```

The cross product between two unit base vectors in three dimensions yields the third unit base vector.

Solving Sets of Equations

■ *optional:* *Solving Linear Sets of Equations*

One of the most importand application of computer algebra and numerical codes is solving systems of equations. Wether you want to solve a complicated switching circuit by solving the Kirchhoff rules or you look for the numerical solution of differential equations, mathematically these problems reduce to solving (huge) sets of linear equations or - even more basic - it means to invert big matrices. Although CAS and numerical schemes are very powerful tools for computationally expensive problems, there are of course limitations as well. This shall be demonstrated with the help of a little example.

The main command for solving linear systems of equations is

```
? LinearSolve
```

LinearSolve[*m*, *b*] finds an *x* which solves the matrix equation *m*.*x* == *b*.
LinearSolve[*m*] generates a LinearSolveFunction[…] which can be applied repeatedly to different *b*. ≫

which solves equations of the form m.x=b.

- **E.g.: Hilbert - Matrix**

We ragard an arbitrary real vector b (filled with random real numbers between 0 and 100)

```
b = Table[Random[Integer, 100], {n, 1, 15}]
```

```
{31, 84, 7, 64, 98, 93, 11, 84, 30, 94, 51, 62, 75, 11, 9}
```

and solve the system H.x=b, i.e. we want to determine the solution vector x. The matrix H was defined above in subsection “Automatic Filling of Lists “.

```
LinearSolve[H, b] // MatrixForm
```

$$\begin{pmatrix} -3\,734\,071\,933\,185 \\ 751\,754\,904\,677\,280 \\ -37\,714\,435\,829\,386\,680 \\ 828\,864\,504\,280\,612\,800 \\ -9\,984\,977\,804\,685\,904\,020 \\ 74\,111\,982\,969\,427\,603\,200 \\ -362\,869\,877\,239\,232\,683\,320 \\ 1\,221\,043\,396\,114\,981\,604\,880 \\ -2\,889\,102\,402\,497\,160\,559\,710 \\ 4\,847\,877\,238\,595\,981\,990\,400 \\ -5\,737\,637\,243\,526\,263\,875\,800 \\ 4\,682\,450\,439\,923\,249\,994\,000 \\ -2\,507\,202\,365\,724\,910\,608\,300 \\ 792\,653\,569\,836\,038\,568\,000 \\ -112\,131\,660\,922\,158\,101\,400 \end{pmatrix}$$

On some computers this may take quite some time since the determinant of this matrix, also known as Hilbert-Matrix, gets numerically very small with increasing problem size. From the theory of linear algebra we know that a system of linear equations is only solvable when the determinant is non-zero. Computationally, matrix inversion gets more and more expensive, the smaller the determinant.

```
? Det
```

Det[m] gives the determinant of the square matrix m. ≫

```
Det[H] // N
```

```
1.05854 × 10^-124
```

The determinant of the 15-dim. Hilbert-Matrix is "practically" zero. The higher dimensional H, the smaller gets its determinant and the more difficult gets its inversion.

```
? HilbertMatrix
```

HilbertMatrix[n] gives the $n \times n$ Hilbert matrix with elements of the form $1/(i + j - 1)$.
HilbertMatrix[{m, n}] gives the $m \times n$ Hilbert matrix. ≫

```
Det[HilbertMatrix[50]] // N
```

```
1.392615568935140 × 10^-1466
```

```
Clear["Global`*"]
```

■ *Solving Coupled Systems of Equations*

Clearly we are often also confronted with nonlinear sets of coupled equations, where we cannot express the problem as sketched before. In this case, we apply the command

```
? Solve
```

Solve[*expr*, *vars*] attempts to solve the system *expr* of equations or inequalities for the variables *vars*.
Solve[*expr*, *vars*, *dom*] solves over the domain *dom*. Common choices of *dom* are Reals, Integers, and Complexes. ≫

in which documentation entry it states already indicatively that it "attempts to solve" the problem posed. *Mathematica* has implemented a variety of algorithms that are tried out.

- **E.g. : System of three linear equations**

```
Solve[{3 * a == b, 2 * a == c, c == 1}, {a, b, c}]
```

$$\left\{\left\{a \to \frac{1}{2},\ b \to \frac{3}{2},\ c \to 1\right\}\right\}$$

■ **E.g.: System of three nonlinear equations**

```
Solve[{3 * a^3 == b^2, 2 * a^2 == c^2, c^3 == 1}, {a, b, c}]
```

$$\left\{\left\{a \to -\frac{1}{\sqrt{2}},\ b \to -\frac{i\sqrt{3}}{2^{3/4}},\ c \to 1\right\},\ \left\{a \to -\frac{1}{\sqrt{2}},\ b \to \frac{i\sqrt{3}}{2^{3/4}},\ c \to 1\right\},\right.$$

$$\left\{a \to \frac{1}{\sqrt{2}},\ b \to -\frac{\sqrt{3}}{2^{3/4}},\ c \to 1\right\},\ \left\{a \to \frac{1}{\sqrt{2}},\ b \to \frac{\sqrt{3}}{2^{3/4}},\ c \to 1\right\},$$

$$\left\{a \to -\sqrt{-\frac{1}{4} - \frac{i\sqrt{3}}{4}},\ b \to -\frac{1}{2} i \sqrt{\frac{3}{2}}\ \left(-1 - i\sqrt{3}\right)^{3/4},\ c \to \frac{1}{2}\left(-1 + i\sqrt{3}\right)\right\},$$

$$\left\{a \to -\sqrt{-\frac{1}{4} - \frac{i\sqrt{3}}{4}},\ b \to \frac{1}{2} i \sqrt{\frac{3}{2}}\ \left(-1 - i\sqrt{3}\right)^{3/4},\ c \to \frac{1}{2}\left(-1 + i\sqrt{3}\right)\right\},$$

$$\left\{a \to \sqrt{-\frac{1}{4} - \frac{i\sqrt{3}}{4}},\ b \to -\frac{1}{2}\sqrt{\frac{3}{2}}\ \left(-1 - i\sqrt{3}\right)^{3/4},\ c \to \frac{1}{2}\left(-1 + i\sqrt{3}\right)\right\},$$

$$\left\{a \to \sqrt{-\frac{1}{4} - \frac{i\sqrt{3}}{4}},\ b \to \frac{1}{2}\sqrt{\frac{3}{2}}\ \left(-1 - i\sqrt{3}\right)^{3/4},\ c \to \frac{1}{2}\left(-1 + i\sqrt{3}\right)\right\},$$

$$\left\{a \to -\sqrt{-\frac{1}{4} + \frac{i\sqrt{3}}{4}},\ b \to -\frac{1}{2} i \sqrt{\frac{3}{2}}\ \left(-1 + i\sqrt{3}\right)^{3/4},\ c \to \frac{1}{2}\left(-1 - i\sqrt{3}\right)\right\},$$

$$\left\{a \to -\sqrt{-\frac{1}{4} + \frac{i\sqrt{3}}{4}},\ b \to \frac{1}{2} i \sqrt{\frac{3}{2}}\ \left(-1 + i\sqrt{3}\right)^{3/4},\ c \to \frac{1}{2}\left(-1 - i\sqrt{3}\right)\right\},$$

$$\left\{a \to \sqrt{-\frac{1}{4} + \frac{i\sqrt{3}}{4}},\ b \to -\frac{1}{2}\sqrt{\frac{3}{2}}\ \left(-1 + i\sqrt{3}\right)^{3/4},\ c \to \frac{1}{2}\left(-1 - i\sqrt{3}\right)\right\},$$

$$\left.\left\{a \to \sqrt{-\frac{1}{4} + \frac{i\sqrt{3}}{4}},\ b \to \frac{1}{2}\sqrt{\frac{3}{2}}\ \left(-1 + i\sqrt{3}\right)^{3/4},\ c \to \frac{1}{2}\left(-1 - i\sqrt{3}\right)\right\}\right\}$$

Mathematica will list all found solutions also if they sometimes not usable for our purposes because they are unphysical or violate some other conditions. In this case we frequently have to come back to

pen and paper in order to pick the “right” results.

■ **Some more useful commands concerning linear algebra:**

```
? RandomReal
```

RandomReal[] gives a pseudorandom real number in the range 0 to 1.
RandomReal[{x_{min}, x_{max}}] gives a pseudorandom real number in the range x_{min} to x_{max}.
RandomReal[x_{max}] gives a pseudorandom real number in the range 0 to x_{max}.
RandomReal[*range*, n] gives a list of n pseudorandom reals.
RandomReal[*range*, {n_1, n_2, …}] gives an $n_1 \times n_2 \times \ldots$ array of pseudorandom reals. ≫

```
A = RandomReal[{-1, 1}, {5, 5}]
```

```
{{-0.474727, 0.388645, -0.461371, 0.938493, 0.950533},
 {0.830478, 0.139342, -0.801181, 0.0834233, 0.259048},
 {0.606089, 0.750426, -0.0652498, 0.50112, -0.928461},
 {-0.903335, 0.207855, 0.368359, -0.051829, 0.0604615},
 {0.0480989, -0.359201, 0.128467, 0.236191, -0.0770996}}
```

```
? MatrixRank
```

MatrixRank[m] gives the rank of the matrix m. ≫

```
MatrixRank[A]
```

```
5
```

```
? Eigenvalues
```

Eigenvalues[m] gives a list of the eigenvalues of the square matrix m.
Eigenvalues[{m, a}] gives the generalized eigenvalues of m with respect to a.
Eigenvalues[m, k] gives the first k eigenvalues of m.
Eigenvalues[{m, a}, k] gives the first k generalized eigenvalues. ≫

```
Eigenvalues[A]
```

```
{0.195529 + 1.07151 ⅈ, 0.195529 - 1.07151 ⅈ,
 -0.325532 + 0.509743 ⅈ, -0.325532 - 0.509743 ⅈ, -0.269558}
```

```
? Eigenvectors
```

Eigenvectors[m] gives a list of the eigenvectors of the square matrix m.
Eigenvectors[{m, a}] gives the generalized eigenvectors of m with respect to a.
Eigenvectors[m, k] gives the first k eigenvectors of m.
Eigenvectors[{m, a}, k] gives the first k generalized eigenvectors. ≫

```
Eigenvectors[A]
```

```
{{0.144597 + 0.32199 i, 0.647119 + 0. i, 0.0223833 - 0.456948 i,
   -0.391941 - 0.0945935 i, -0.127755 + 0.261649 i},
 {0.144597 - 0.32199 i, 0.647119 + 0. i, 0.0223833 + 0.456948 i,
   -0.391941 + 0.0945935 i, -0.127755 - 0.261649 i},
 {-0.0352433 - 0.480426 i, -0.327103 - 0.0011468 i,
   -0.259849 - 0.380617 i, 0.542821 + 0. i, -0.276221 - 0.278583 i},
 {-0.0352433 + 0.480426 i, -0.327103 + 0.0011468 i,
   -0.259849 + 0.380617 i, 0.542821 + 0. i, -0.276221 + 0.278583 i},
 {0.320862, 0.369529, 0.656308, -0.272299, 0.505578}}
```

```
? CharacteristicPolynomial
```

CharacteristicPolynomial[m, x] gives the characteristic polynomial for the matrix m.
CharacteristicPolynomial[{m, a}, x] gives the generalized characteristic polynomial with respect to a. ≫

```
CP[x_] = CharacteristicPolynomial[A, x]
```

$$-0.116983 - 0.603626\, x - 0.979111\, x^2 - 1.36765\, x^3 - 0.529564\, x^4 - x^5$$

The eigenvalues of a matrix (resp. a linear map) are the zeros of the characteristical polynomial which we can probe:

```
CP[Eigenvalues[A]]
```

$$\{2.22045 \times 10^{-16} + 0.\, i,\ 2.22045 \times 10^{-16} + 0.\, i,\ -2.77556 \times 10^{-17} + 5.55112 \times 10^{-17}\, i,\ -2.77556 \times 10^{-17} - 5.55112 \times 10^{-17}\, i,\ -1.73472 \times 10^{-17}\}$$

... is approximately zweo.

Analysis - Working with Functions, Derivatives, Integrals

Introductory Remarks

■ *Mathematical Entities in Mathematica*

As mentioned before, *Mathematica* has some syntactic idiosyncrasies. As an example we regard one of the most beautiful formulas in mathematics:

DONT :

```
e^(i*pi) == -1
```

$$e^{i\,pi} == -1$$

DO :

```
E^(I*Pi) == -1
```

```
True
```

```
?E
```

E is the exponential constant e (base of natural logarithms), with numerical value $\simeq 2.71828$. ≫

```
?I
```

I represents the imaginary unit $\sqrt{-1}$. ≫

```
?Pi
```

Pi is π, with numerical value $\simeq 3.14159$. ≫

```
? Infinity
```

Infinity or ∞ is a symbol that represents a positive infinite quantity. ≫

■ *optional:* Natural Constants etc.

```
<< PhysicalConstants`
```

```
SpeedOfLight
```

$$\frac{299\,792\,458\ \text{Meter}}{\text{Second}}$$

```
FineStructureConstant
```

```
0.00729735
```

```
AgeOfUniverse
```

$$4.7 \times 10^{17}\ \text{Second}$$

```
MagneticFluxQuantum
```

$$2.06783 \times 10^{-15}\ \text{Weber}$$

■ *Important Embedded Functions*

Clearly the most important functions are implemented in *Mathematica;* as mentioned before all these commands begin with a capital letter. Some Examples:

```
? Sin
```

Sin[*z*] gives the sine of *z*. ≫

```
? Log
```

Log[z] gives the natural logarithm of z (logarithm to base e).
Log[b, z] gives the logarithm to base b. ≫

```
? Gamma
```

Gamma[z] is the Euler gamma function $\Gamma(z)$.
Gamma[a, z] is the incomplete gamma function $\Gamma(a, z)$.
Gamma[a, z_0, z_1] is the generalized incomplete
gamma function $\Gamma(a, z_0) - \Gamma(a, z_1)$. It is also a unit of magnetic flux density. ≫

```
? DiracDelta
```

DiracDelta[x] represents the Dirac delta function $\delta(x)$.
DiracDelta[x_1, x_2, …] represents the multidimensional Dirac delta function $\delta(x_1, x_2, ...)$. ≫

■ *Series*

In order to get used to handling series, we examine a frequent application, namely determining the limit of a series.

```
Clear["Global`*"]
(* please enter this input to delete all previous definitions
 which would partially conflict with the upcoming section *)
```

We can define series analogously to functions

```
a[n_] = (1 + 1 / n) ^ (n)
```

$$\left(1 + \frac{1}{n}\right)^n$$

or recursively

```
b[n_] := b[n - 1] * b[n - 2]
b[0] := 1
b[1] := 2
b[2] := 2
```

and determine the limit or generate a list of entries.

```
? Limit
```

Limit[*expr*, $x \to x_0$] finds the limiting value of *expr* when x approaches x_0. ≫

```
Limit[a[n], n → Infinity]
```

```
e
```

```
Table[b[n], {n, 0, 10}]
```

```
{1, 2, 2, 4, 8, 32, 256, 8192,
 2 097 152, 17 179 869 184, 36 028 797 018 963 968}
```

Of course also sums are implemented im *Mathematica*

```
? Sum
```

Sum[f, {i, i_{max}}] evaluates the sum $\sum_{i=1}^{i_{max}} f$.

Sum[f, {i, i_{min}, i_{max}}] starts with $i = i_{min}$.

Sum[f, {i, i_{min}, i_{max}, di}] uses steps di.

Sum[f, {i, {i_1, i_2, ...}}] uses successive values i_1, i_2, ...

Sum[f, {i, i_{min}, i_{max}}, {j, j_{min}, j_{max}}, ...] evaluates the multiple sum $\sum_{i=i_{min}}^{i_{max}} \sum_{j=j_{min}}^{j_{max}} \ldots f$.

Sum[f, i] gives the indefinite sum $\sum_i f$. ≫

```
Sum[x^n / (n!^2), {n, 0, Infinity}]
```

```
BesselI[0, 2 √x]
```

```
? BesselI
```

BesselI[n, z] gives the modified Bessel function of the first kind $I_n(z)$. ≫

```
Sum[x^k / k!, {k, 0, n}]
```

$$\frac{e^x\ \text{Gamma}[1+n, x]}{n!}$$

and especially useful is also the Series command which expands a function into its Taylor series.

```
?Series
```

Series[f, {x, x_0, n}] generates a power series expansion for f about the point $x = x_0$ to order $(x - x_0)^n$.
Series[f, {x, x_0, n_x}, {y, y_0, n_y}, ...] successively finds series expansions with respect to x, then y, etc. ≫

```
Series[Sin[x], {x, 0, 15}]
```

$$x - \frac{x^3}{6} + \frac{x^5}{120} - \frac{x^7}{5040} + \frac{x^9}{362\,880} - \frac{x^{11}}{39\,916\,800} + \frac{x^{13}}{6\,227\,020\,800} - \frac{x^{15}}{1\,307\,674\,368\,000} + O[x]^{16}$$

Differentiation and Integration

■ *Integration*

Basic differentiation syntax was presented already in section I/O III. Integration commands are analogous

```
Clear["Global`*"]
(* please enter this input to delete all previous definitions
 which would partially conflict with the upcoming section *)
```

```
?Integrate
```

Integrate[f, x] gives the indefinite integral $\int f\, dx$.

Integrate[f, {x, x_{min}, x_{max}}] gives the definite integral $\int_{x_{min}}^{x_{max}} f\, dx$.

Integrate[f, {x, x_{min}, x_{max}}, {y, y_{min}, y_{max}}, ...] gives the multiple integral $\int_{x_{min}}^{x_{max}} dx \int_{y_{min}}^{y_{max}} dy \ldots f$. ≫

- **E.g.: Some Integrals**

```
Integrate[Sin[x], x]
```

```
-Cos[x]
```

```
Integrate[E^(x^2), x]
```

$$\frac{1}{2}\sqrt{\pi}\ \text{Erfi}[x]$$

```
?Erfi
```

Erfi[z] gives the imaginary error function erf(iz)/i. ≫

```
Integrate[Sin[x]*Cos[x], {x, -Pi, Pi}]
```

```
0
```

```
Integrate[E^(-x), {x, 0, Infinity}]
```

```
1
```

Differential Equations

Solving Ordinary Differential Equations (ODEs)

Solving differential equations is another major strenght of CAS. While solving an ODE can require a sophisticated ansatz and several analytic techniques, *Mathematica* is a lot faster in giving results with help of the command DSolve.

```
Clear["Global`*"]
(* please enter this input to delete all previous definitions
 which would partially conflict with the upcoming section *)
```

- **E.g.: Solving ODEs without boundary condition**

```
? DSolve
```

DSolve[*eqn*, *y*, *x*] solves a differential equation for the function *y*, with independent variable *x*.
DSolve[{eqn_1, eqn_2, ...}, {y_1, y_2, ...}, *x*] solves a list of differential equations.
DSolve[*eqn*, *y*, {x_1, x_2, ...}] solves a partial differential equation. ≫

```
DSolve[y'[x] + y[x] == 0, y[x], x]
```

```
{{y[x] → e^-x C[1]}}
```

- **E.g.: Solving ODEs with boundary condition**

```
DSolve[{z'[x] + z[x] == 0, z[0] == 1}, z[x], x]
```

```
{{z[x] → e^-x}}
```

- **E.g.: Solving systems of ODEs**

```
eqn1 := s'[x] + t[x] == 0
eqn2 := t'[x] + s[x] == 0
```

```
DSolve[{eqn1, eqn2}, {s[x], t[x]}, x]
```

$$\left\{\left\{s[x] \to \frac{1}{2} e^{-x}\left(1+e^{2x}\right) C[1] - \frac{1}{2} e^{-x}\left(-1+e^{2x}\right) C[2],\right.\right.$$
$$\left.\left. t[x] \to -\frac{1}{2} e^{-x}\left(-1+e^{2x}\right) C[1] + \frac{1}{2} e^{-x}\left(1+e^{2x}\right) C[2]\right\}\right\}$$

- **E.g.: Numerical solutions of ODEs**

Whenever *Mathematica* fails to find a closed analytic solution of an ODE, such as the following example shows

```
DSolve[y''[x] + y'[x] + y[x] * (1 + y[x]) == 0, y[x], x]
```

```
DSolve[y[x] (1 + y[x]) + y'[x] + y''[x] == 0, y[x], x]
```

there is the possibility to look for solutions numerically. Iterative algorithms implemented in *Mathe-*

matica are capable of finding numerical solutions of ODEs.

```
? NDSolve
```

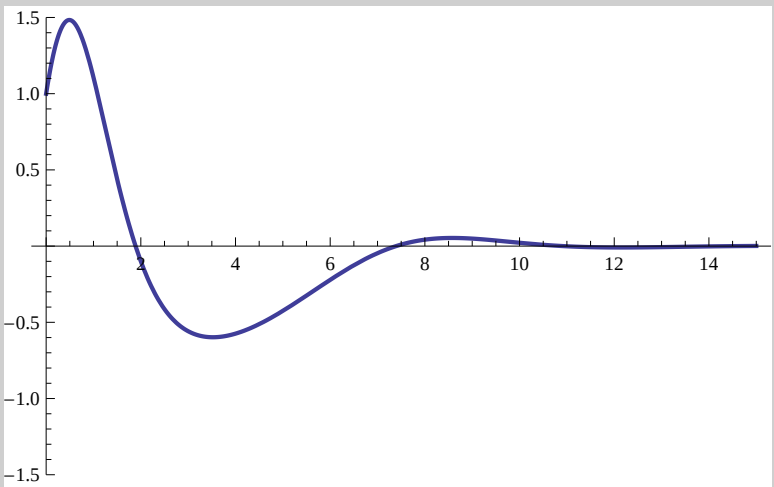
NDSolve[*eqns*, *y*, {*x*, x_{min}, x_{max}}] finds a numerical solution to the ordinary differential equations *eqns* for the function *y* with the independent variable *x* in the range x_{min} to x_{max}.
NDSolve[*eqns*, *y*, {*x*, x_{min}, x_{max}}, {*t*, t_{min}, t_{max}}] finds a numerical solution to the partial differential equations *eqns*.
NDSolve[*eqns*, {y_1, y_2, ...}, {*x*, x_{min}, x_{max}}] finds numerical solutions for the functions y_i. ≫

```
sol = NDSolve[{y[x] (1 + y[x]) + y'[x] + y''[x] == 0, y[0] == 1, y'[0] == 2},
  y[x], {x, 0, 15}]
```

```
{{y[x] → InterpolatingFunction[{{0., 15.}}, <>][x]}}
```

The result is output as “InterpolatingFunction” meaning that the function is only known at those points where the algorithm has provided a numerical result.

```
Plot[Evaluate[y[x] /. sol], {x, 0, 15},
 PlotRange → 1.5, PlotStyle → {Thick}, Background → White]
```



CAS: Alternatives

Mathematica is a comprehensive computer algebra system, however it is a proprietary product. There are several free open source alternatives:

Axiom: http://axiom-wiki.newsynthesis.org/FrontPage
Scilab: http://www.scilab.org/

Octave: http://www.gnu.org/software/octave/
Maxima: http://maxima.sourceforge.net/
Wiris: http://www.wiris.com/
Geogebra: http://www.geogebra.org/cms/