Introduction to Wolfram *Mathematica*

The goal of this introduction is to give a brief overwiew of the kinds of operations *Mathematica* can be used for. This tutorial is far from complete, but it covers some of the basics that should help you get started. *Mathematica* offers a lot of very powerful constructions and functionality beyond what is covered here.

To expand and view the subsections, move the mouse to the right margin of the document and double-click on the vertical bars with an arrow pointing downward. You can close the respective “cells” by double-clicking on the vertical par embracing that cell.

Notebooks

The files to which you write *Mathematica* code and over which you interact with the *Mathematica* Kernel are called Notebooks. This file here is one. Notebooks have the ending .nb. It is possible to launch *Mathematica* from the command line, too, in which case you interact more directly with the Kernel.

A Notebook is basically a series of “cells”, where cells can be of different types, e.g. input cells, output cells, text cells, or cells with display formula, items, lists, etc.

This organisation allows you to interleave text, comments and code, which can be very helpful.

Onle cells of type “Input” and “Code” can actually be evaluated by the Kernel. To evaluate a cell of one of these types, place the curser into this cell and then press the Shift+Return key combination. The result is then presented in a cell of the type “Output” just below the input cell you evaluated.

Mathematica can be used to write manuscripts (there is an export function to $\LaTeX$) or presentations, and it offers a number of interective features. We are not going to discuss these here, though.

On the right-hand side of a Notebook, there are vertical bars, which represent the hierarchical structure of cells and groups of cells such as subsections, sections, chapters, etc. By klicking on these bars, you can collapse or expand respective levels.

Amont input cells, there are two types. “Normal” input cells are evaluated only if you put your cursor there and press the Shift+Return combination. “Intitialization Cells”, however, are evaluated automatically when you choose “Evaluate Initialization Cells” from the “Evaluation” Menu. It is a good idea to place code such as function or variable definitions into initialization cells. However, calculations that need a lot of time or are more of a “scratch” type should not be assigned to initialization cells.

As a final remark, coding in *Mathematica* may seem a bit awkward if you use it the first time. I strongly recommend using the different “Assistants” available in the “Palettes” Menu. These offer, for example, easy access to greek letters or constructions like matrices, differentials, fractions, etc. There are short-cuts for these, which you will be picking up as you go. The more you use *Mathematica*, the more you will find yourself usint the short-cuts instead of the assistant palletes.

Depending on which “Stylesheet” (Menu Format) you use, the default cell type varies. If you use the “Default” Stylesheed, then a new cell is typically an input cell. You can change the type of a cell in the Menu Format/Style.

In an input cell, you can comment out code or text by surrounding it by surrounding it with brackets and asterisks. Example: (* This is my comment *).

Basics

Getting help

You can access the documentation to any *Mathematica* command by typing “?Command”; this provides a short summary with a link to the full documentation page:

```
? Sin
```

```
Sin[z] gives the sine of z. >>
```

A more detailed summary is given if you type two question marks :

```
?? Sin
```

```
Sin[z] gives the sine of z. >>

Attributes[Sin] = {Listable, NumericFunction, Protected}
```

Variables and assignments

You can use (almost) any variable name (it cannot start with a number and cannot have subscripts), and you can assign numerical expressions (e.g. a real number) or symbolic expressions (e.g. another variable or a function) to a variable name.

There are two types of assignment: direct and delayed assignment. The direct assignment assigns the value to a variable once, namely when you execute the respective cell, and then the variable keeps this value until you assign a different value to it. In contrast, if you use the delayed assignment, then *Mathematica* just keeps in mind the symbolic expression you assigned to the variable, but only evaluates it when the variable is actually used.

Direct assignment using “=”:

```
(* Directly assigning 5 to a *)
a = 5

5

(* b is now a, which is 5, so b is 5 *)
b = a

5

(* Assign a new value to a *)
a = 10

10

(* Note that b is still 5 *)

b

5
```

Delayed assignment using “:=”:

```
(* Directly assigning 5 to a *)
a = 5

5
```

```
(* Now we assign a to c, but the assignment is now delayed! *)
c := a
```

```
(* Assign a new value to a *)
a = 10
```

```
10
```

```
(* Note that c takes the current value of a,
not the one that a had when c was defined, which ws 5! *)
```

```
c
```

```
10
```

Sometimes you want to “clear” variables of their assignments. This is done as follows:

```
a
```

```
12
```

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

```
a
```

```
a
```

```
b
```

```
b
```

```
c
```

```
c
```

Basic operations

Summation, substraction, multiplication, and division:

```
a + b
```

```
a + b
```

```
8 - b
```

```
8 - b
```

```
3.8 * 2
```

```
7.6
```

Instead of a “*” you an also use a space for multiplication. If you put a space between two numbers, *Mathematica* will automatically put a “×” in there. For symbolic variables, the space is sufficient.

```
3.8 × 2
```

```
x y
```

```
x y
```

```
x * y
```

```
x y
```

```
a / c + 9 - 2 * 3
```

$3 + \frac{a}{c}$

Powers:

```
x ^ a
```

x^a

b^2

b^2

Roots:

```
Sqrt[4]
```

```
2
```

$\sqrt[4]{a^8}$

```
144
```

Exponentials:

```
Exp[x]
```

e^x

Alternatively, we can type

$e^{-a x+b}$

$e^{b-a x}$

The double-stroke e can be obtained by typing “Esc + e + e + Esc”. The escape key Esc is the gate to many short-cuts. Another example are Greek letters. To obtain the first three letters of the Greek alphabet, type “Esc + a + Esc”, “Esc + b + Esc”, and “Esc + g + Esc” to obtain α, β, and γ. To write letters in double-stroke font, such as $\mathbb{E}$, use “Esc + dsE + Esc”. Similarly, for letters in script shape, such as $\mathcal{A}$, use “Esc + scA + Esc”.

Manipulating expressions

Expanding a product:

```
Expand[a (b + c) - a (e - f)]
```

```
a b + a c - a e + a f
```

Alternatively, you can put the command after the expression, using “//”:

```
a (b + c) - a (e - f) // Expand
```

```
a b + a c - a e + a f
```

```
? 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*. >>

Factoring an expression:

```
Factor[a + c a - d a - a^2 + e a^2]
```

```
a (1 - a + c - d + a e)
```

```
? Factor
```

Factor[*poly*] factors a polynomial over the integers.
Factor[*poly*, Modulus → *p*] factors a polynomial modulo a prime *p*.
Factor[*poly*, Extension → {a_1, a_2, …}] factors a polynomial
allowing coefficients that are rational combinations of the algebraic numbers a_i. >>

Simplifying expressions:

```
expr1 = a / b + a^4 - b c + 4 / b
```

$$a^4 + \frac{4}{b} + \frac{a}{b} - b\,c$$

```
Simplify[expr1]
```

$$\frac{4 + a + a^4 b - b^2 c}{b}$$

```
FullSimplify[expr1]
```

$$a^4 + \frac{4 + a}{b} - b\,c$$

Again, we can use the alternative “post-fix” notation:

```
expr1 // Simplify
```

$$\frac{4 + a + a^4 b - b^2 c}{b}$$

Sometimes, it is useful to group subterms by a (set of) focal variable(s). This is done using “Collect”:

```
expr2 = a b + (x - z) b + (e + f - g) a
```

```
a b + a (e + f - g) + b (x - z)
```

```
Collect[expr2, b]
```

```
a (e + f - g) + b (a + x - z)
```

```
Collect[expr2, a]
```

```
a (b + e + f - g) + b (x - z)
```

```
Collect[expr2, {a, b}]
```

```
a (b + e + f - g) + b (x - z)
```

To determine the degree of a polynomical in a focal variable, use

```
expr1
```

$$a^4 + \frac{4}{b} + \frac{a}{b} - b\,c$$

```
Exponent[expr1, a]
```

```
4
```

Data structures

There is one atomic data structure, called a List. You can build more complicated structures by nesting lists. E.g., a 2x2 matrix is just a list of two sublists, where each sublist has one element.

```
myList1 = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10}
```

```
{1, 2, 3, 4, 5, 6, 7, 8, 9, 10}
```

```
myNestedList1 = {{a, b}, {c, d}}
```

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

A very efficient way of constructing lists is to use the “Table” command:

```
Table[i, {i, 1, 10, 1}]
```

```
{1, 2, 3, 4, 5, 6, 7, 8, 9, 10}
```

Table[a^x**, {x, 1, 10, 1}]**

$\{a, a^2, a^3, a^4, a^5, a^6, a^7, a^8, a^9, a^{10}\}$

```
myNestedList2 = Table[{x, y}, {y, 1, 5, 1}, {x, {a, b, c, d, e}}]
```

```
{{{a, 1}, {b, 1}, {c, 1}, {d, 1}, {e, 1}},
 {{a, 2}, {b, 2}, {c, 2}, {d, 2}, {e, 2}}, {{a, 3}, {b, 3}, {c, 3}, {d, 3}, {e, 3}},
 {{a, 4}, {b, 4}, {c, 4}, {d, 4}, {e, 4}}, {{a, 5}, {b, 5}, {c, 5}, {d, 5}, {e, 5}}}
```

Use “TableForm” or “MatrixForm” to display lists in a visually more appealing way:

```
myNestedList2 // TableForm
```

```
a    b    c    d    e
1    1    1    1    1
a    b    c    d    e
2    2    2    2    2
a    b    c    d    e
3    3    3    3    3
a    b    c    d    e
4    4    4    4    4
a    b    c    d    e
5    5    5    5    5
```

```
MatrixForm[myNestedList2]
```

$$\begin{pmatrix} \begin{pmatrix} a \\ 1 \end{pmatrix} & \begin{pmatrix} b \\ 1 \end{pmatrix} & \begin{pmatrix} c \\ 1 \end{pmatrix} & \begin{pmatrix} d \\ 1 \end{pmatrix} & \begin{pmatrix} e \\ 1 \end{pmatrix} \\ \begin{pmatrix} a \\ 2 \end{pmatrix} & \begin{pmatrix} b \\ 2 \end{pmatrix} & \begin{pmatrix} c \\ 2 \end{pmatrix} & \begin{pmatrix} d \\ 2 \end{pmatrix} & \begin{pmatrix} e \\ 2 \end{pmatrix} \\ \begin{pmatrix} a \\ 3 \end{pmatrix} & \begin{pmatrix} b \\ 3 \end{pmatrix} & \begin{pmatrix} c \\ 3 \end{pmatrix} & \begin{pmatrix} d \\ 3 \end{pmatrix} & \begin{pmatrix} e \\ 3 \end{pmatrix} \\ \begin{pmatrix} a \\ 4 \end{pmatrix} & \begin{pmatrix} b \\ 4 \end{pmatrix} & \begin{pmatrix} c \\ 4 \end{pmatrix} & \begin{pmatrix} d \\ 4 \end{pmatrix} & \begin{pmatrix} e \\ 4 \end{pmatrix} \\ \begin{pmatrix} a \\ 5 \end{pmatrix} & \begin{pmatrix} b \\ 5 \end{pmatrix} & \begin{pmatrix} c \\ 5 \end{pmatrix} & \begin{pmatrix} d \\ 5 \end{pmatrix} & \begin{pmatrix} e \\ 5 \end{pmatrix} \end{pmatrix}$$

Accessing elements of lists:

```
myList1[[3]]
```

```
3
```

```
myList1[[All]]
```

```
{1, 2, 3, 4, 5, 6, 7, 8, 9, 10}
```

```
myList1[[1 ;; 4]]
```

```
{1, 2, 3, 4}
```

```
myList1[[{1, 5, 6}]]
```

```
{1, 5, 6}
```

```
myNestedList1[[1]]
```

```
{a, b}
```

```
myNestedList1[[1, 2]]
```

```
b
```

```
myNestedList2[[1]]
```

```
{{a, 1}, {b, 1}, {c, 1}, {d, 1}, {e, 1}}
```

```
myNestedList2[[2, 4]]
```

```
{d, 2}
```

```
myNestedList2[[All, 2]]
```

```
{{b, 1}, {b, 2}, {b, 3}, {b, 4}, {b, 5}}
```

Counting from the end:

```
myList1[[-3]]
```

```
8
```

```
myList1[[-6 ;; -4]]
```

```
{5, 6, 7}
```

Asking for the position of an element in a list:

```
Position[myList1, 3]
```

```
{{3}}
```

```
Position[myNestedList2, {b, 2}]
```

```
{{2, 2}}
```

```
(* Check: *)
myNestedList2[[2, 2]]
```

```
{b, 2}
```

Some list manipulations

```
myNestedList1
```

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

```
Flatten[myNestedList1]
```

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

```
myNestedList2
```

```
{{{a, 1}, {b, 1}, {c, 1}, {d, 1}, {e, 1}},
 {{a, 2}, {b, 2}, {c, 2}, {d, 2}, {e, 2}}, {{a, 3}, {b, 3}, {c, 3}, {d, 3}, {e, 3}},
 {{a, 4}, {b, 4}, {c, 4}, {d, 4}, {e, 4}}, {{a, 5}, {b, 5}, {c, 5}, {d, 5}, {e, 5}}}
```

```
Flatten[myNestedList2]
```

```
{a, 1, b, 1, c, 1, d, 1, e, 1, a, 2, b, 2, c, 2, d, 2, e, 2, a, 3, b, 3, c,
 3, d, 3, e, 3, a, 4, b, 4, c, 4, d, 4, e, 4, a, 5, b, 5, c, 5, d, 5, e, 5}
```

```
myList1
```

```
{1, 2, 3, 4, 5, 6, 7, 8, 9, 10}
```

```
myList2 = {a, b, c, d, e, f, g, h, i, j}
```

```
{a, b, c, d, e, f, g, h, i, j}
```

```
Riffle[myList1, myList2]
```

```
{1, a, 2, b, 3, c, 4, d, 5, e, 6, f, 7, g, 8, h, 9, i, 10, j}
```

```
Join[myList1, myList2[[1 ;; 5]]]
```

```
{1, 2, 3, 4, 5, 6, 7, 8, 9, 10, a, b, c, d, e}
```

```
Partition[myList1, 2]
```

```
{{1, 2}, {3, 4}, {5, 6}, {7, 8}, {9, 10}}
```

```
Partition[Riffle[myList1, myList2], 2]
```

```
{{1, a}, {2, b}, {3, c}, {4, d}, {5, e}, {6, f}, {7, g}, {8, h}, {9, i}, {10, j}}
```

```
myList3 = {20, 4, 10, 2, 4, 5}
```

```
{20, 4, 10, 2, 4, 5}
```

```
Sort[myList3]
```

```
{2, 4, 4, 5, 10, 20}
```

```
Split[myList1]
```

```
{{1}, {2}, {3}, {4}, {5}, {6}, {7}, {8}, {9}, {10}}
```

```
Split[myList3]
```

```
{{20}, {4}, {10}, {2}, {4}, {5}}
```

```
Split[Sort[myList3]]
```

```
{{2}, {4, 4}, {5}, {10}, {20}}
```

Rules and their application

Rules can be used to replace expressions or symbols by another expression at a given point. This allows you to distinguish between “general” expressions and their specific meaning for particular variable/paramter values.

Rules are defined using an arrow “->”:

```
myRule1 := a → 5
```

```
b = a
```

```
a
```

```
b
a
```

Rules are applied using “/.”:

```
b /. myRule1
5
```

```
Sin[x]
Sin[x]
```

```
Sin[x] /. {x → 2 π}
0
```

Alternatively, and equivalently, we can use “Replace” to apply the rule:

```
Replace[b, myRule1]
5
```

Textual versus short-cut commands

For many commands, there are short-cuts. We have seen an example above: The command “Replace” has the short-cut “./”. Rather than listing all combinations of commands and short-cuts here, we just mention this. You will encounter these pairs in practice early enough.

Functions

Function definitions

To define a function that takes arguments, we use the following syntax:

```
In[1]:= myFunc1[a_, b_, c_] := a + b / c
```

Note that we use the delayed assignment “:=” here. Moreover, arguments are declared using the so-called pattern “x_”, which stands for “any expression, to be referred to as *x*”. It often makes sense to place functions in an “initialization cell”, so that they are defined whenever the Notebook is initialised.

We can then use the function as follows:

```
myFunc1[1, 4, 3]
```

$\frac{7}{3}$

Another example:

```
In[2]:= myFunc2[x_] := α x^2 + β x + γ
```

In this case, we define the function as a function of *x*, with parameters *α*, *β*, and *γ* to be defined when we use the function:

```
myFunc2[x]
```

$x^2\,\alpha + x\,\beta + \gamma$

```
myFunc2[x] /. {α → 1, β → 2, γ → 4}
```

$4 + 2\,x + x^2$

Recursions

Recursion equations are, in most cases, best defined using the construction "*f*[x_] := *f*[*x*] = expression including (*x* – 1)", in combination with the definition of *f*[0]. The advantage of this construction is that the function is computed once for every *x* and these values are then remembered. However, this only pays off if we initially assign specific numeric values to any of the parameters that are part of "expression including (*x* – 1)".

For instance, the recursion equation for the logistic model of growth in discrete time can be implemented as follows:

```
In[3]:= Clear[n]
n0 = 10;
myr = 0.5;
myK = 1000;
```

$$n[t\_] := n[t] = n[t-1] + myr\, n[t-1] \left(1 - \frac{n[t-1]}{myK}\right)$$

```
n[0] = n0;

n[0]

10

n[1]

14.95

n[100]

1000.

n[2000]
```

$$\text{Hold}\left[n[978-1] + myr\, n[978-1] \left(1 - \frac{n[978-1]}{myK}\right)\right]$$

Mathematica by default sets the recursion limit to 1024 to prevent unintended infinite loops. To navigate around this, you can set this recursion limit to infinity. It is a good idea to do this locally in a so-called "Block":

```
Block[{$RecursionLimit = Infinity}, n[2000]]

1000.
```

As another example, consider the diploid model of natural selection in discrete time:

```
Clear[p]
p0 = 0.01;
myW11 = 1;
myW12 = 1 - h s /. {h → 0.5, s → 0.05};
myW22 = 1 - s /. {s → 0.05};
```

$$p[t\_] := p[t] = p[t-1] \left((p[t-1]\, myW11 + (1 - p[t-1])\, myW12) \middle/ \left(p[t-1]^2\, myW11 + 2\, p[t-1]\, (1 - p[t-1])\, myW12 + (1 - p[t-1])^2\, myW22\right)\right)$$

```
p[
   0] =
  p0;
```

```
p[3]
0.0108015
```

```
p[10]
0.0129268
```

```
p[100]
0.119143
```

```
p[1000]
1.
```

Approximating a function using a Taylor series

A Taylor series of a function can be obtained as follows:

```
myFunc2[x]
```

$x^2 \alpha + x \beta + \gamma$

A 1st-order Taylor series of “myFunc2” around $x = 0$:

```
Series[myFunc2[x], {x, 0, 1}]
```

$\gamma + \beta x + O[x]^2$

```
Series[myFunc2[x], {x, 0, 1}] // Normal
```

$x \beta + \gamma$

A 2nd-order Taylor series of a 3rd-order polynomial in x around $x = 0$:

$\text{Series}\left[a x^3 - b x^2 + c x - d, \{x, 0, 1\}\right]$

$-d + c x + O[x]^2$

The same, but around $x = 1$:

$\text{Series}\left[a x^3 - b x^2 + c x - d, \{x, 1, 1\}\right]$

$(c - d) + (a + c) (x - 1) + O[x - 1]^2$

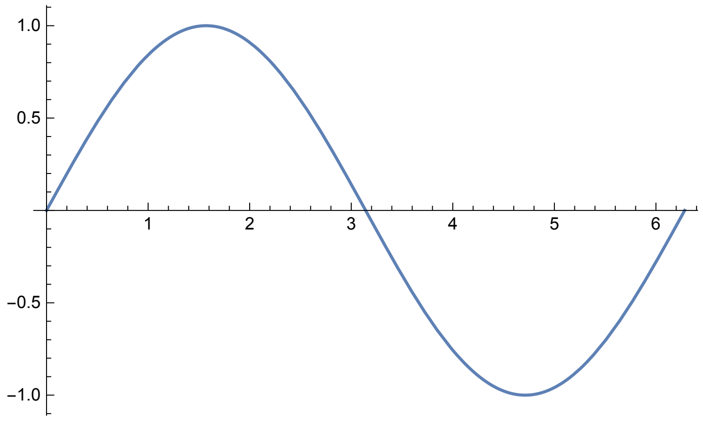
$\text{Series}\left[a x^3 - b x^2 + c x - d, \{x, 0, 2\}\right]$

$-d + c x - a x^2 + O[x]^3$

Plotting

Plotting a continuous function

```
Plot[Sin[ϕ], {ϕ, 0, 2 π}]
```

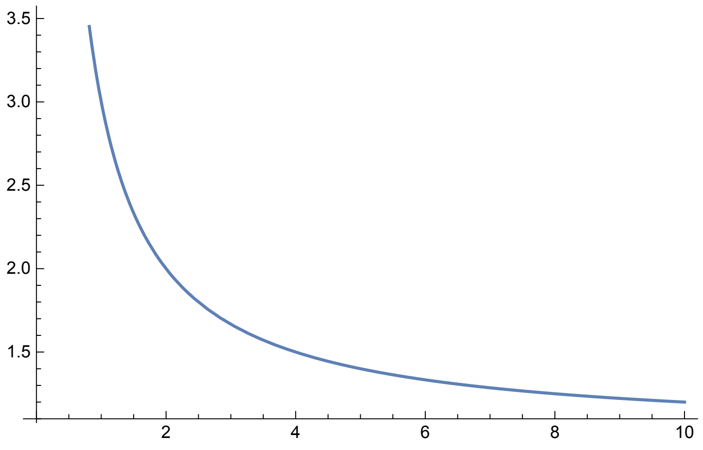


```
? myFunc1
```

```
Global`myFunc1
```

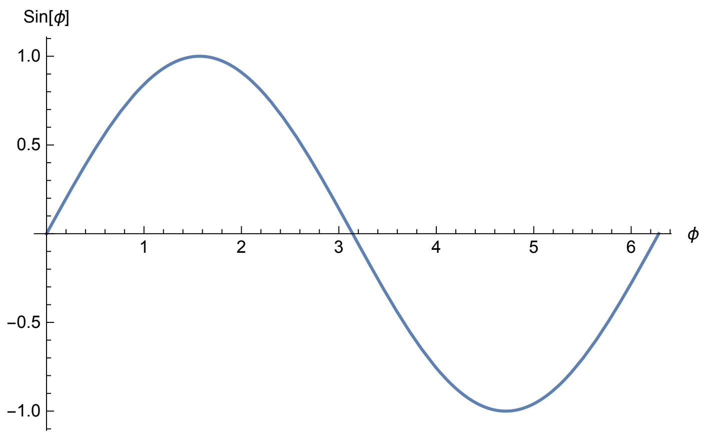
myFunc1[a_, b_, c_] := $a + \frac{b}{c}$

```
Plot[myFunc1[1, 2, c], {c, 0, 10}]
```



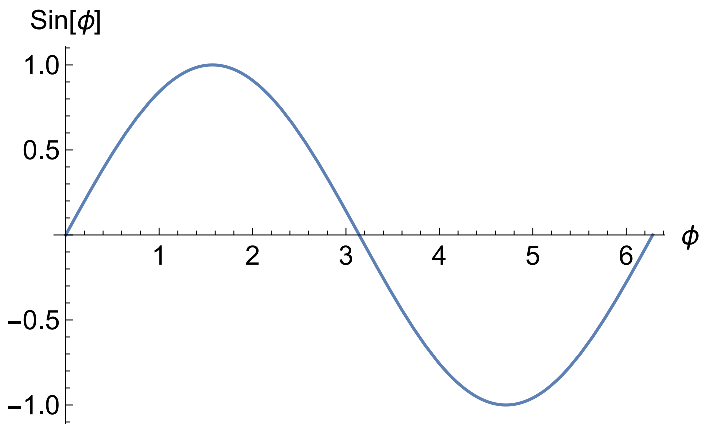
Adding axis labels:

```
Plot[Sin[ϕ], {ϕ, 0, 2 π},
 AxesLabel → {"ϕ", "Sin[ϕ]"}
]
```



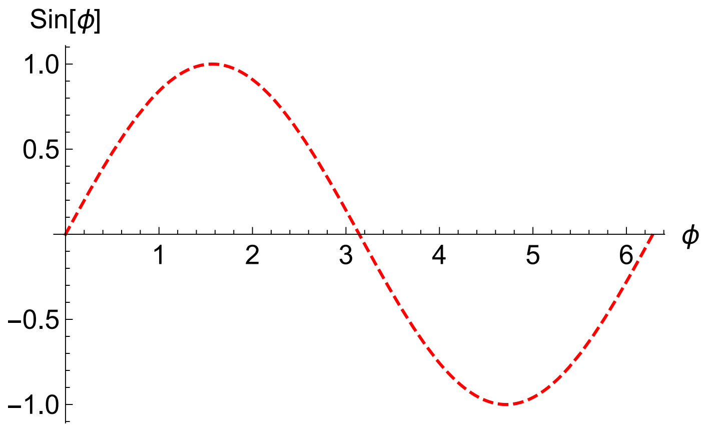
Increasing the font size for the labels:

```
Plot[Sin[ϕ], {ϕ, 0, 2 π},
 AxesLabel → {"ϕ", "Sin[ϕ]"},
 LabelStyle → Directive[FontSize → 14]
]
```



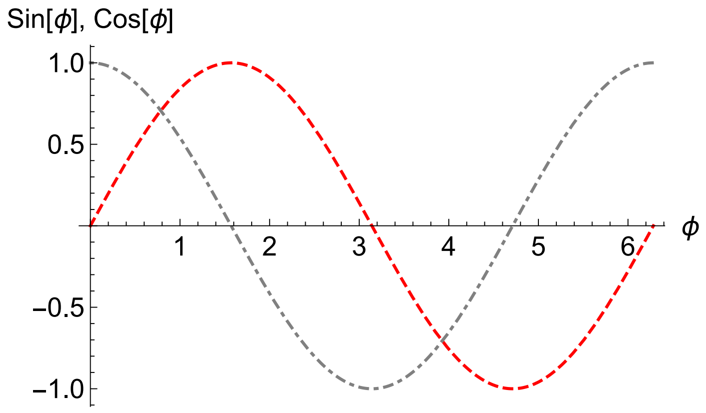
Changing the style of the line:

```
Plot[Sin[ϕ], {ϕ, 0, 2 π},
 AxesLabel → {"ϕ", "Sin[ϕ]"},
 LabelStyle → Directive[FontSize → 14],
 PlotStyle → {{Red, Dashed}}
]
```



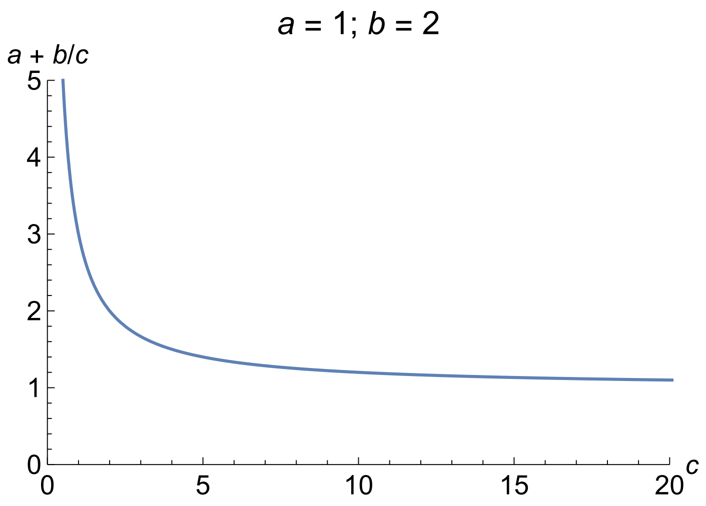
Plotting multiple functions:

```
Plot[{Sin[ϕ], Cos[ϕ]}, {ϕ, 0, 2 π},
 AxesLabel → {"ϕ", "Sin[ϕ], Cos[ϕ]"},
 LabelStyle → Directive[FontSize → 14],
 PlotStyle → {{Red, Dashed}, {Gray, DotDashed}}
]
```



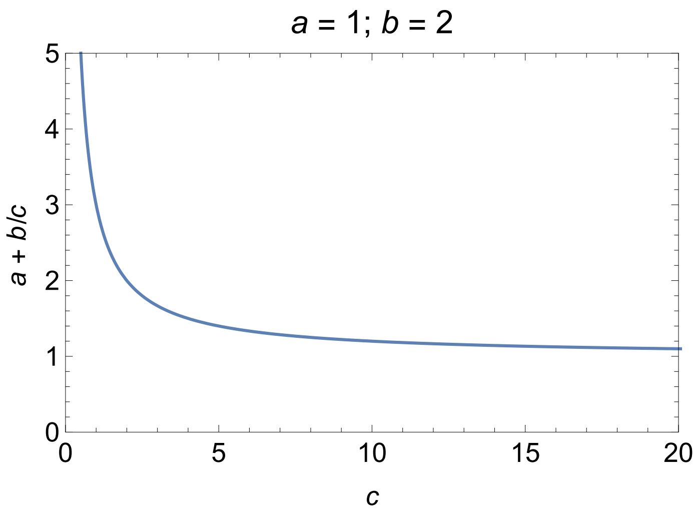
Changing the plot range and adding a title:

```
Plot[myFunc1[1, 2, c], {c, 0, 100},
 PlotRange → {{0, 20}, {0, 5}},
 AxesLabel → {"c", "a + b/c"},
 PlotLabel → "a = 1; b = 2",
 LabelStyle → Directive[FontSize → 14]
]
```



Using a frame and annotating the frame edges:

```
Plot[myFunc1[1, 2, c], {c, 0, 100},
 PlotRange → {{0, 20}, {0, 5}},
 Frame → True,
 FrameLabel → {"c", "a + b/c"},
 PlotLabel → "a = 1; b = 2",
 LabelStyle → Directive[FontSize → 14]
]
```



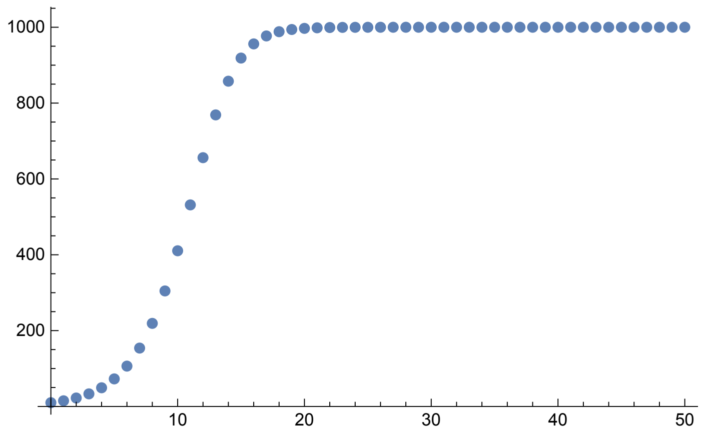
Plotting discrete points

Let us plot the recursion equation for the discrete-time logistic growth model:

```
plotData1 = Table[{t, n[t]}, {t, 0, 50}]
```

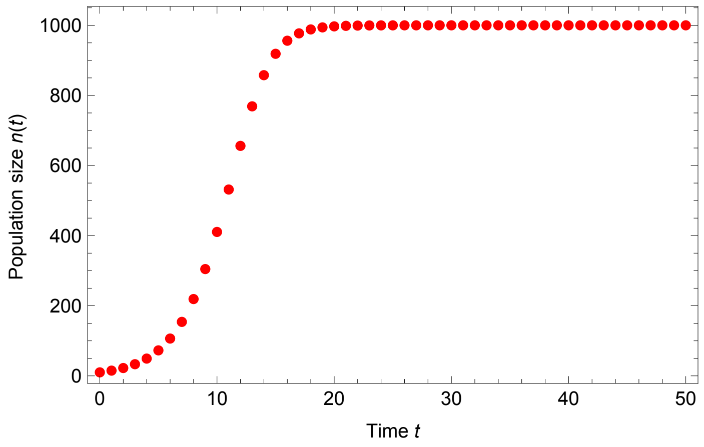
```
{{0, 10}, {1, 14.95}, {2, 22.3132}, {3, 33.2209}, {4, 49.2796}, {5, 72.7051},
 {6, 106.415}, {7, 153.96}, {8, 219.088}, {9, 304.632}, {10, 410.548},
 {11, 531.547}, {12, 656.05}, {13, 768.874}, {14, 857.727}, {15, 918.743},
 {16, 956.07}, {17, 977.07}, {18, 988.272}, {19, 994.067}, {20, 997.016},
 {21, 998.504}, {22, 999.251}, {23, 999.625}, {24, 999.812}, {25, 999.906},
 {26, 999.953}, {27, 999.977}, {28, 999.988}, {29, 999.994}, {30, 999.997},
 {31, 999.999}, {32, 999.999}, {33, 1000.}, {34, 1000.}, {35, 1000.},
 {36, 1000.}, {37, 1000.}, {38, 1000.}, {39, 1000.}, {40, 1000.},
 {41, 1000.}, {42, 1000.}, {43, 1000.}, {44, 1000.}, {45, 1000.},
 {46, 1000.}, {47, 1000.}, {48, 1000.}, {49, 1000.}, {50, 1000.}}
```

```
ListPlot[plotData1]
```



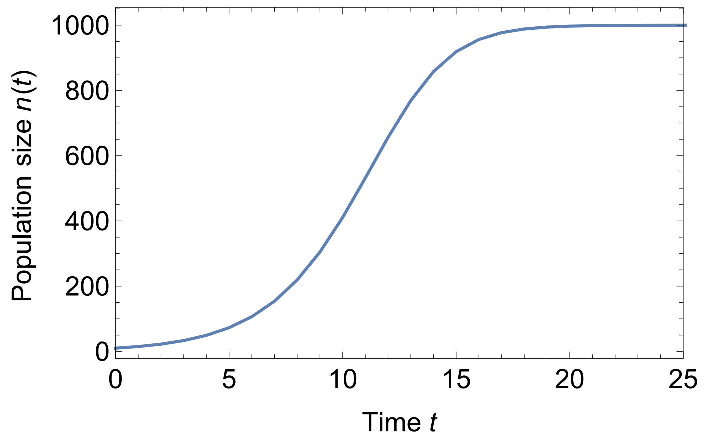
Improving the look and annotating the plot:

```
listPlot1 = ListPlot[plotData1,
  Frame → True,
  FrameLabel → {"Time t", "Population size n(t)"},
  LabelStyle → Directive[FontSize → 10],
  PlotStyle → Red
 ]
```



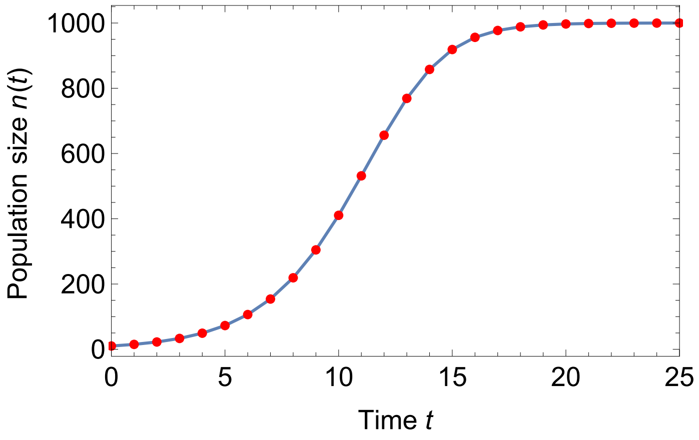
Joining the dots and restricting the plot range along the *x* axis:

```
listPlot2 = ListPlot[plotData1,
  Frame → True,
  FrameLabel → {"Time t", "Population size n(t)"},
  LabelStyle → Directive[FontSize → 14],
  Joined → True,
  PlotRange → {{0, 25}, Full}
 ]
```



Combining two plots:

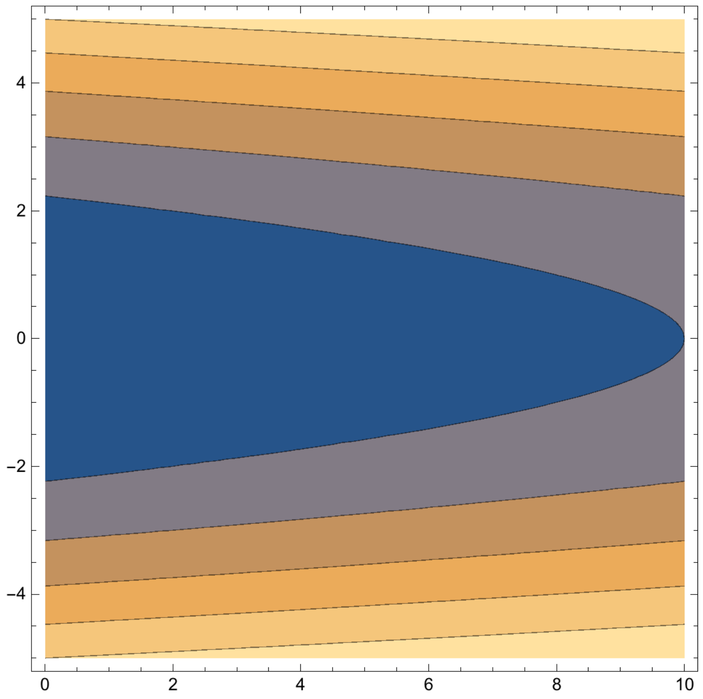
```
Show[{listPlot2, listPlot1}
]
```



Note that the plot range is adopted from the first plot element in the list given to “Show”, as are other plotting arguments (e.g. the size of the axes labels).

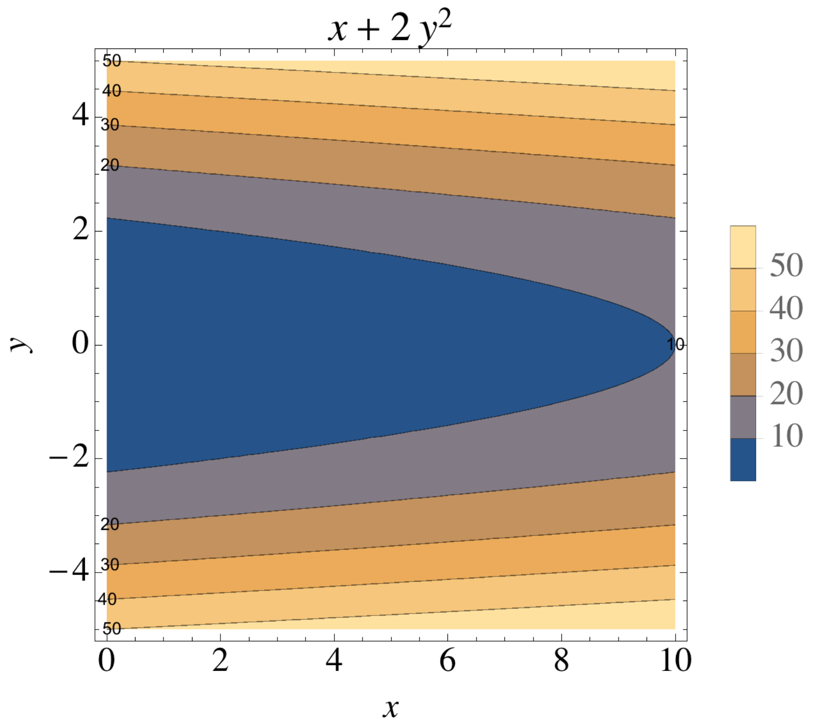
Contour plots

```
ContourPlot[
 x + 2 y^2, {x, 0, 10}, {y, -5, 5}
]
```



Again some styling:

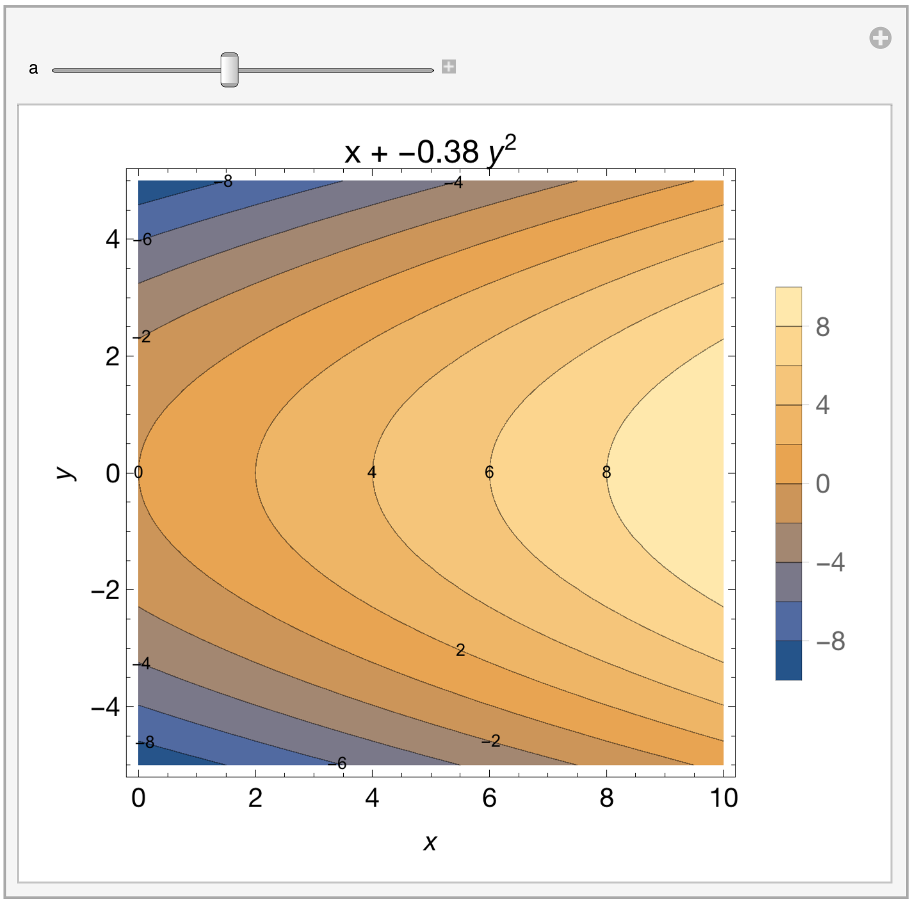
```
ContourPlot[
 x + 2 y^2, {x, 0, 10}, {y, -5, 5},
 FrameLabel → {"x", "y"},
 PlotLabel → x + 2 y^2,
 LabelStyle → Directive[FontSize → 18, FontFamily → "Times"],
 PlotLegends → Automatic,
 ContourLabels → True
]
```



Interactive plots

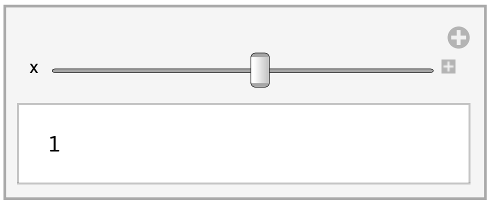
Interactive plots can be drawn using “Manipulate”. Although this command can also be used in other contexts, I use it most often when plotting.

```
Manipulate[
 ContourPlot[
  x + a y^2, {x, 0, 10}, {y, -5, 5},
  FrameLabel → {"x", "y"},
  PlotLabel → "x + " <> ToString[a] <> " y^2",
  LabelStyle → Directive[FontSize → 14, FontFamily → "Helvetica Neue"],
  PlotLegends → Automatic,
  ContourLabels → True
 ],
 {{a, -0.5}, -5, 5}
]
```



Using the “Manipulate” command in another context:

```
Manipulate[
 x,
 {{x, 1}, -10, 10, 2}
]
```



```
1
```

Essentially, you can “manipulate” any *Mathematica* expression. Just be aware of the fact that *Mathematica* internally pre-computes that expression for the range of variables you indicate. This may be overly costly for complicated expressions!

Equations

Solving equations (symbolically and numerically)

First of all, it is time to introduce the “equals” operator “==”, and higlight the difference to the “assignment” operator “=”. Equations are expressed as follows:

```
a x + c == 5
```

```
c + a x == 5
```

We can also assign an equation to a variable for easier access later on. You can use both the immediate and delayed assignment for this:

```
myEq1 = a x + c == 5
```

```
c + a x == 5
```

```
myEq1Alt := a x + c == 9
```

```
myEq1
```

```
c + a x == 5
```

```
myEq1Alt
```

```
c + a x == 9
```

To solve an equation symbolically for a variable of interest (e.g. *x* here), we use

```
Solve[a x + c == 5, x]
```

$\{\{x \to \frac{5 - c}{a}\}\}$

Mathematica returns a (potentially nested) list representing the set of all solutions it can find.

We can also solve an equation that is accessed as a variable:

```
Solve[myEq1Alt, x]
```

$\{\{x \to \frac{9 - c}{a}\}\}$

If we try to solve a transcendental equation, *Mathematica* complains because in general such equations cannot be solved analytically:

$\text{Solve}[e^{-a x} + x == b x^2, x]$

$\text{Solve}[e^{-a x} + x == a x^2, x]$

$\text{Solve}[x == e^{-a x}, x]$

$\{\{x \to \frac{\text{ProductLog}[a]}{a}\}\}$

However, you can sometimes find (a) numerical solution(s) for a specific set of parameter values using “NSolve”. Let us try this for the transcendental equation we just encountered:

$\text{NSolve}[x == e^{-0.5 x}, x]$

```
{{x → 0.703467}}
```

```
? NSolve
```

NSolve[*expr*, *vars*] attempts to find numerical approximations
to the solutions of the system *expr* of equations or inequalities for the variables *vars*.
NSolve[*expr*, *vars*, Reals] finds solutions over the domain of real numbers. ≫

Solving systems of equations

To solve a system of equations, we proceed as follows:

```
myEq2 := a x + y == c
myEq3 := b x - y == d
```

```
Solve[{myEq2, myEq3}, {x, y}]
```

$$\left\{\left\{x \to -\frac{-c-d}{2\,a},\ y \to \frac{c-d}{2}\right\}\right\}$$

Using assumptions

Sometimes, it is crucial to tell *Mathematica* about assumptions you are happy to make. This often allows for much simpler solutions, sometimes it even makes the difference to *Mathematica* finding a solution at all.

```
Sqrt[x^2]
```

$$\sqrt{x^2}$$

```
Simplify[Sqrt[x^2], Assumptions → {x > 0}]
```

```
x
```

Dealing with inequalities

A useful command in the context of inequalities is “Reduce”.

```
Reduce[{0 < x < 2, 1 < x < 4}, x]
```

```
1 < x < 2
```

```
Reduce[{x < 1, x > 3}, x]
```

```
False
```

Again, telling *Mathematica* about assumptions you are happy to make can simplify things a lot:

```
Simplify[Reduce[a x + b x^2 < c, x]]
```

$$x \in \text{Reals}\ \&\&$$

$$\Bigg(\Bigg(c < 0\ \&\&\ \Bigg(\Bigg(\Bigg(a \neq 0\ ||\ \sqrt{\frac{c}{b}} < x\ ||\ \sqrt{\frac{c}{b}} + x < 0\Bigg)\ \&\&\ \Bigg(a == 0\ ||\ b^2\sqrt{\frac{a^2+4\,b\,c}{b^2}} < b\,(a+2\,b\,x)\ ||$$

$$b\left(a+b\left(\sqrt{\frac{a^2+4\,b\,c}{b^2}}+2\,x\right)\right) < 0\Bigg)\ \&\&\ b < 0\Bigg)\ ||$$

$$(b == 0\ \&\&\ a \neq 0\ \&\&\ a\,x < c)\ ||\ \Big(a \neq 0\ \&\&\ b > 0\ \&\&\ c\left(a^2+4\,b\,c\right) < 0\ \&\&$$

$$\sqrt{\frac{a^2+4\,b\,c}{b^2}} > \frac{a}{b} + 2\,x\ \&\&\ b\left(a+b\left(\sqrt{\frac{a^2+4\,b\,c}{b^2}}+2\,x\right)\right) > 0\Big)\Bigg)\Bigg)\ ||$$

$$\Bigg(c == 0\ \&\&\ \Bigg(\Bigg((a \neq 0\ ||\ x \neq 0)\ \&\&\ \Bigg(a == 0\ ||\ \sqrt{\frac{a^2}{b^2}} < \frac{a}{b} + 2\,x\ ||\ b\left(a+b\left(\sqrt{\frac{a^2}{b^2}}+2\,x\right)\right) < 0\Bigg)\ \&\&$$

$$b < 0\Bigg)\ ||\ (b == 0\ \&\&\ ((a < 0\ \&\&\ x > 0)\ ||\ (a > 0\ \&\&\ x < 0)))\ ||$$

$$\Bigg(a \neq 0\ \&\&\ b > 0\ \&\&\ \sqrt{\frac{a^2}{b^2}} > \frac{a}{b} + 2\,x\ \&\&\ b\left(a+b\left(\sqrt{\frac{a^2}{b^2}}+2\,x\right)\right) > 0\Bigg)\Bigg)\Bigg)\ ||$$

$$\Bigg(c > 0\ \&\&\ \Bigg(\Bigg(a == 0\ \&\&\ \Bigg(\Bigg(\sqrt{\frac{c}{b}} + x > 0\ \&\&\ x < \sqrt{\frac{c}{b}}\Bigg)\ ||\ b \leq 0\Bigg)\Bigg)\ ||\ (b == 0\ \&\&\ a \neq 0\ \&\&\ a\,x < c)\ ||$$

$$\Bigg(a \neq 0\ \&\&\ \Bigg(\Bigg(4\,b + \frac{a^2}{c} == 0\ \&\&\ b\left(a+b\left(\sqrt{\frac{a^2+4\,b\,c}{b^2}}+2\,x\right)\right) \neq 0\Bigg)\ ||$$

$$\Bigg(b < 0\ \&\&\ c\left(a^2+4\,b\,c\right) > 0\ \&\&\ \Bigg(b\left(a+b\left(\sqrt{\frac{a^2+4\,b\,c}{b^2}}+2\,x\right)\right) <$$

$$0\ ||\ b\left(a-b\sqrt{\frac{a^2+4\,b\,c}{b^2}}+2\,b\,x\right) > 0\Bigg)\Bigg)\ ||\ c\left(a^2+4\,b\,c\right) < 0\ ||$$

$$\Bigg(b > 0\ \&\&\ \sqrt{\frac{a^2+4\,b\,c}{b^2}} > \frac{a}{b} + 2\,x\ \&\&\ b\left(a+b\left(\sqrt{\frac{a^2+4\,b\,c}{b^2}}+2\,x\right)\right) > 0\Bigg)\Bigg)\Bigg)\Bigg)\Bigg)\Bigg)$$

Let us tell *Mathematica* what we are willing to assume about the parameters and variables:

```
Simplify[Reduce[a x + b x^2 < c, x], Assumptions → {c < 0, a > 0, b > 0, x ∈ Reals}]
```

$a^2 + 4\,b\,c > 0\ \&\&\ -\frac{a + \sqrt{a^2 + 4\,b\,c}}{2\,b} < x < \frac{-a + \sqrt{a^2 + 4\,b\,c}}{2\,b}$

Calculus

Derivatives

To take the derivative of a function, you either use the prime notation or the command “D”:

```
myf[x_] := Sin[x] + x^2
```

```
myf'[x]
```

```
2 x + Cos[x]
```

```
D[myf[x], x]
```

```
2 x + Cos[x]
```

Taking the second derivative:

```
myf''[x]
```

```
2 - Sin[x]
```

Or:

```
D[myf[x], {x, 2}]
```

```
2 - Sin[x]
```

Integrals

To obtain an indefinite integral, we use

```
Integrate[2 x + Cos[x], x]
```

$x^2 + \mathrm{Sin}[x]$

```
Integrate[myFunc2[x], x]
```

$\frac{x^3\,\alpha}{3} + \frac{x^2\,\beta}{2} + x\,\gamma$

For a definite integral, we have to give the boundaries of integration:

```
Integrate[2 x + Cos[x], {x, 0, π}]
```

π^2

```
Integrate[myFunc2[x], {x, -10, 5}]
```

$375\,\alpha - \frac{75\,\beta}{2} + 15\,\gamma$

Numerical integration is done using “NIntegrate”:

```
NIntegrate[Sin[Sin[x]], {x, 0, 2}]
```

```
1.24706
```

Linear algebra

Writing vectors and matrices

As mentioned above, the universal data structure used by *Mathematica* are (nested) lists. A vector is therefore represented as a one-dimensional list; a matrix as a nested list of depth 2, i.e. with two levels.

```
Clear[p]
```

```
myVec1 := {v1, v2, v3}
```

```
myMat1 := {{a, b, c}, {d, e, f}, {g, h, i}}
myMat2 := {{p, q}, {r, s}, {t, u}}
```

```
myVec1
```

```
{v1, v2, v3}
```

Again, we can use “MatrixForm” or “TableForm” to obtain a more userfriendly display of nested lists:

```
myVec1 // MatrixForm
```

$$\begin{pmatrix} v1 \\ v2 \\ v3 \end{pmatrix}$$

```
myMat1 // MatrixForm
```

$$\begin{pmatrix} a & b & c \\ d & e & f \\ g & h & i \end{pmatrix}$$

```
TableForm[myMat1]
```

```
a    b    c
d    e    f
g    h    i
```

Matrix operations

We multiply matrices or vectors using “.”:

```
myMat1.myMat2
```

```
{{a p + b r + c t, a q + b s + c u},
 {d p + e r + f t, d q + e s + f u}, {g p + h r + i t, g q + h s + i u}}
```

```
% // MatrixForm
```

$$\begin{pmatrix} a p + b r + c t & a q + b s + c u \\ d p + e r + f t & d q + e s + f u \\ g p + h r + i t & g q + h s + i u \end{pmatrix}$$

```
myVec1.myMat1
```

```
{a v1 + d v2 + g v3, b v1 + e v2 + h v3, c v1 + f v2 + i v3}
```

The operators “+“, “-“, and “^” work element-wise:

```
myMat1 + 3
```

```
{{3 + a, 3 + b, 3 + c}, {3 + d, 3 + e, 3 + f}, {3 + g, 3 + h, 3 + i}}
```

```
myMat2 + myVec1
```

```
{{p + v1, q + v1}, {r + v2, s + v2}, {t + v3, u + v3}}
```

Determinant, inverse, transpose, trace

```
Det[myMat1]
```

```
-c e g + b f g + c d h - a f h - b d i + a e i
```

```
Inverse[myMat1] // Simplify
```

$$\left\{\left\{\frac{f h - e i}{c e g - b f g - c d h + a f h + b d i - a e i}, \frac{c h - b i}{-c e g + b f g + c d h - a f h - b d i + a e i}, \frac{c e - b f}{c e g - b f g - c d h + a f h + b d i - a e i}\right\}, \left\{\frac{f g - d i}{-c e g + b f g + c d h - a f h - b d i + a e i}, \frac{c g - a i}{c e g - b f g - c d h + a f h + b d i - a e i}, \frac{c d - a f}{-c e g + b f g + c d h - a f h - b d i + a e i}\right\}, \left\{\frac{e g - d h}{c e g - b f g - c d h + a f h + b d i - a e i}, \frac{b g - a h}{-c e g + b f g + c d h - a f h - b d i + a e i}, \frac{b d - a e}{c e g - b f g - c d h + a f h + b d i - a e i}\right\}\right\}$$

```
Transpose[myMat1]
```

```
{{a, d, g}, {b, e, h}, {c, f, i}}
```

```
Tr[myMat1]
```

```
a + e + i
```

Eigenvalues and eigenvectors

```
Eigenvalues[{{a, b}, {c, d}}]
```

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

```
Eigenvectors[{{a, b}, {c, d}}]
```

$$\left\{\left\{-\frac{-a + d + \sqrt{a^2 + 4 b c - 2 a d + d^2}}{2 c}, 1\right\}, \left\{-\frac{-a + d - \sqrt{a^2 + 4 b c - 2 a d + d^2}}{2 c}, 1\right\}\right\}$$

Example: Analysis of a simple non-linear SI model

Let us consider a simple model of disease with two categories of host individuals: infected ones and those susceptible to the disease. Let the number of infected and susceptible individuals at time *t* be *Y*(*t*) and *S*(*t*), respectively. We assume that the population of susceptible hosts is replenished by immigration at a total rate θ, and that recovered individuals immediately become susceptible again, i.e. there is no class of immune or resistant hosts. Further, denote the transition rate of the disease by β. Recall that under the mass-action principle the transition rate is given by the product of the rate of contact, *c*, between susceptible and infected hosts times the probability of transmission conditional on a contact, *a*, i.e. $\beta = c\,a$. Last, let *d* be the per capita background mortality rate of the host, *v* the additional mortality caused by infection, and γ the rate of clearance of the disease through defense mechanisms of the host.

Writing the differential equations

The model can then be described in continuous time by the following pair of differential equations :

$$\begin{aligned} \frac{dS}{dt} &= \theta - d\,S - \beta\,S\,Y + \gamma\,Y, \\ \frac{dY}{dt} &= \beta\,S\,Y - (d + \nu + \gamma)\,Y. \end{aligned} \tag{1}$$

Implementing these equations in *Mathematica* (choosing to call the differential of variable *X* “XDot”):

```
In[25]:= SDot := θ - d S - β S Y + γ Y
         YDot := β S Y - (d + ν + γ) Y
```

Finding the equilibria

To identify the equilibria $\left(\hat{S},\ \hat{Y}\right)$ of the model, we set $dS/dt = 0$ and $dY/dt = 0$. This yields the two equilibrium conditions that must be jointly satisfied:

$$\begin{aligned} 0 &= \theta - d\,\hat{S} - \beta\,\hat{S}\,\hat{Y} + \gamma\,\hat{Y}, \\ 0 &= \beta\,\hat{S}\,\hat{Y} - (d + \nu + \gamma)\,\hat{Y}. \end{aligned} \tag{2}$$

In contrast to linear multivariable models, there can now be multiple equilibria for non-linear models. We wish to identify all of them. In practice, this can be a challenging, sometimes impossible talk, as not all solutions to Eq. (2) might be found explicitly. In the case of our example, the solutions can be found by hand by factoring the equations in (2) and looking for values of the variables that make any of the factors equal to 0.

Here, we will use *Mathematica* to find the solutions and simplify these:

```
In[27]:= solEq1 = Solve[{SDot == 0, YDot == 0}, {S, Y}] // FullSimplify
```

Out[27]= $\left\{\left\{S \to \frac{d+\gamma+\nu}{\beta},\ Y \to \frac{\beta\,\theta - d\,(d+\gamma+\nu)}{\beta\,(d+\nu)}\right\},\ \left\{S \to \frac{\theta}{d},\ Y \to 0\right\}\right\}$

Mathematica returns the solutions in the form of “rules”. The second solution $\left(\hat{S} = \theta/d,\ \hat{Y} = 0\right)$ corresponds to the case where the disease is absent. The first solution represents the so-called “endemic equilibrium” where the disease is present.

Assessing stability

The stability of an equilibrium is determined by a local stability analysis, which is a *linearisation* of the nonlinear model near the equilibrium of interest. If there are several equilibria of interest, the stability analysis must be repeated separately for each of them.

The linearisation is achieved by constructing a matrix whose elemenents are the partial derivatives of the original functions with respect to each dynamical variable in turn. This matrix of partial derivatives is called the **Jacobian matrix**. To assess if an equilibrium of interest is stable, we need to evaluate the Jacobian matrix at that equilibrium. This matrix is sometimes referred to as the **local stability matrix** of that equilibrium. We then need to compute the eigenvalues of the local stability matrix. In continuous time, the equilibrium is stable if the real part of the eigenvalue with the largest real part (i.e. of the leading eigenvalue) is negative.

We start by constructing the Jacobian matrix:

```
In[30]:= Jac = Table[Table[D[f, v], {v, {S, Y}}], {f, {SDot, YDot}}]

Out[30]= {{-d - Y β, -S β + γ}, {Y β, -d + S β - γ - ν}}
```

In[31]:=
```
Jac // MatrixForm
```

Out[31]//MatrixForm=

$$\begin{pmatrix} -d - Y\beta & -S\beta + \gamma \\ Y\beta & -d + S\beta - \gamma - \nu \end{pmatrix}$$

We then evaluate the Jacobian matrix at each equilibrium in turn, find the leading eigenvalue, and determine if the equilibrium is stable.

Let us start with the equilibrium where the disease is absent:

In[38]:=
```
locStabMat1 = Jac /. solEq1⟦2⟧ // Simplify
```

Out[38]= $\left\{\left\{-d, \gamma - \frac{\beta\theta}{d}\right\}, \left\{0, -d - \gamma + \frac{\beta\theta}{d} - \nu\right\}\right\}$

In[39]:=
```
locStabMat1 // MatrixForm
```

Out[39]//MatrixForm=

$$\begin{pmatrix} -d & \gamma - \frac{\beta\theta}{d} \\ 0 & -d - \gamma + \frac{\beta\theta}{d} - \nu \end{pmatrix}$$

We note that this is an upper triangular matrix, which means we can read off the eigenvalues from the diagonal. They are

$$r_1 = \frac{\beta\theta}{d} - (d + v + \gamma) \text{ and} \qquad (3)$$
$$r_2 = -d.$$

Because the background mortality d is a positive constant, r_2 is always negative, and the stability only depends on the sign of r_1. To assess when r_1 is negative, we start by setting

$$r_1 = \frac{\beta\theta}{d} - (d + v + \gamma) < 0. \qquad (4)$$

We note that θ/d is equal to $\hat{S}$ when the disease is absent. Making this substitution, dividing both sides by $(d + v + \gamma)$, and rearranging, we obtain

$$\beta\hat{S}/(d + v + \gamma) < 1. \qquad (5)$$

The quantity on the left-hand side is sometimes called R_0, the reproductive number. It represents *the expected number of new infections produced per infected host when a disease is introduced into a susceptible population*. If $R_0 < 1$, the disease will not spread, but if $R_0 > 1$ it will.

Let us turn to the second equilibrium, the so-called endemic equilibrium.

It is a good idea to first check when this equilibrium is biologically feasible. In our case, this means determining when both $\hat{S}$ and $\hat{I}$are positive. We note that $\hat{S} = (d + \gamma + v)/\beta$ is always positive, as all the parameters are positive. The condition under which $\hat{I}$ is positive turns out to be the opposite of the condition in Eq. (4), i.e. to the condition that the disease will not spread. In other words, the second equilibrium is biologically feasible whenever the disease can spread when rare.

Next, we evaluate the Jacobian matrix at the endemic equilibrium:

In[41]:=
```
locStabMat2 = Jac /. solEq1⟦1⟧ // Simplify
```

Out[41]= $\left\{\left\{\frac{d\gamma - \beta\theta}{d + \nu}, -d - \nu\right\}, \left\{\frac{\beta\theta - d(d + \gamma + \nu)}{d + \nu}, 0\right\}\right\}$

In[42]:=
```
locStabMat2 // MatrixForm
```

Out[42]//MatrixForm=

$$\begin{pmatrix} \frac{d\gamma - \beta\theta}{d+\nu} & -d - \nu \\ \frac{\beta\theta - d(d+\gamma+\nu)}{d+\nu} & 0 \end{pmatrix}$$

Although *Mathematica* finds the eigenvalues without problems in this case, they turn out to be relatively complicated:

```
In[44]:= Eigenvalues[locStabMat2] // Simplify
```

Out[44]=
$$\left\{-\frac{1}{2\,(d+\nu)}\left(-d\,\gamma+\beta\,\theta+\sqrt{\left(4\,d^4+4\,d^3\,(\gamma+3\,\nu)+\beta\,\theta\left(\beta\,\theta-4\,\nu^2\right)+d^2\left(\gamma^2-4\,\beta\,\theta+8\,\gamma\,\nu+12\,\nu^2\right)+d\left(4\,\nu^2\,(\gamma+\nu)-2\,\beta\,\theta\,(\gamma+4\,\nu)\right)\right)}\right),\right.$$
$$\left.\frac{1}{2\,(d+\nu)}\left(d\,\gamma-\beta\,\theta+\sqrt{\left(4\,d^4+4\,d^3\,(\gamma+3\,\nu)+\beta\,\theta\left(\beta\,\theta-4\,\nu^2\right)+d^2\left(\gamma^2-4\,\beta\,\theta+8\,\gamma\,\nu+12\,\nu^2\right)+d\left(4\,\nu^2\,(\gamma+\nu)-2\,\beta\,\theta\,(\gamma+4\,\nu)\right)\right)}\right)\right\}$$

Instead, we use a helpful rule applicable to 2 x 2 matrices. This rule says that the two eigenvalues of a 2 x 2 matrix **M** must have the same sign if their product – which is the determinant Det[**M**] – is positive. Further, when Det[**M**] > 0, the real part of both eigenvalues will be positive if their sum – which is the trace of the matrix, Tr[**M**] – is positive. The real parts of both eigenvalues will be negative if Tr[**M**] < 0. When Det[**M**] < 0, one eigenvalue will be positive and one negative.

Therefore, the endemic equilibrium is stable if the determinant of the respective local stability matrix is positive, and its trace is negative. The determinant of this matrix is

```
In[46]:= Det[locStabMat2] // Simplify
```

Out[46]= $-d^2+\beta\,\theta-d\,(\gamma+\nu)$

Again, after some rearrangement, the condition for the determinant to be positive turns out to be the opposite of the condition in Eq. (4). In other words, the determinant is positive whenever the endemic equilibrium is biologically feasible.

The trace of the local stability marix is

```
In[48]:= Tr[locStabMat2]
```

Out[48]= $\dfrac{d\,\gamma-\beta\,\theta}{d+\nu}$

Because all parameters are positive, the sign of the trace depends on the numerator $d\,\gamma-\beta\,\theta$.

Going back to the condition under which $\hat{I}$ is biologically feasible, we recall that

$$\beta\,\theta-d\,(\gamma+d+v)=\beta\,\theta-d^2+\gamma\,d+d\,v>0$$

must hold. In other words, we require $\beta\,\theta>d^2+\gamma\,d+d\,v$ for $\hat{I}$ to be biologically feasible. Because all parameters are positive, we immediately see that $\beta\,\theta>d\,\gamma$ whenever $\hat{I}$ is biologically feasible, and hence the trace of the local stability matrix is negative. This implies that the real parts of both eigenvalues are negative, and that the endemic equilibrium is locally stable whenever it is biologically feasible.