W.S.G. ABACUS
TARIQ BONTEKOE & HIDDE WIERINGA

Basic course — Wolfram Mathematica

September 20, 2016

1 INTRODUCTION

This introductory course is written to give a basic idea of the possibilities of the software Wolfram Mathematica[1]. Using examples from practice, many different fields within mathematics are discussed. Please note that this is in no way a complete list of the things you can do with Mathematica. Use this course as an overview to get started.
Finally, it must be stressed that the online documentation of the Wolfram Language is very good. Use it to find out more about a function, syntax or the best way to calculate something. A basic introduction for programmers is also available online.
But perhaps the most important part of mathematics and programming: *have fun!*

2 BASICS

» Start Mathematica, and generate a new document.

A document is called a *notebook*. Save the notebook by pressing ctrl + S, or use the menu. A notebook consists of *cells*. Cells can be nested together. Cells can either be evaluated or contain static text.

» Make a new cell by clicking the horizontal text cursor, and start typing.

You can delete a cell by selecting it on the right hand side of the window, and clicking the Del. button.

2.1 MATHEMATICAL SYMBOLS

Mathematica is designed to be used for many mathematical operations. To make this easier for you, symbols can be entered visually using the *Basic Math Assistant*. It can be found in the *Palettes* menu.

» Open the Basic Math Assistant, and click a few symbols.

2.2 EVALUATION

Evaluation of a cell is done using the ⇧ + ⏎ keys, or the numpad ⏎ key.

» Type `1` in a new cell, and evaluate it.

2.3 VARIABLES

Give a variable a value by using the `=` symbol. Even though a variable might not yet have a value, you can still use it! Clear a variable using the function `Clear`.

» Evaluate `a + a`. Next, assign a value to `a` and calculate `a + a` again. Clear the variable `a`.

2.4 CALLING FUNCTIONS

Wolfram Language, the language which is used in Wolfram Mathematica, is a functional language. This means that most of the things you do is calling functions. Functions can be evaluated by writing their name, a `[`, the arguments (if any) and a `]`. For example: `Sqrt[2]` or `N[3, 4]` or `CurrentImage[]`.

» Evaluate the function `Power` with the arguments i (from the Math Assistant) and 2.

2.5 LISTS

Lists are ordered structures which can hold any kind of items, or be empty. A list is delimited by using `{` and `}`, and its items are separated using a comma (`,`).

» Create a list with the items $\sqrt{2}$, π and e^2.

2.6 FINDING DOCUMENTATION

It is impossible to know all syntax, functions and usages of the entire Wolfram Language. When using a function you do not know, you can hover with your mouse over the name. Then, you can either

1. Click the arrows to find out the basic syntax of the function;

[1] Version 11 of the software has been used.

2. Click the info button, to open the documentation of the function.

» Open the documentation of the `Power` function.

3 Calculus

Basic documentation
Calculus is a very broad subject. Below you'll find a few functions and assignments that you are most likely to use in your daily life as a mathematician.
As you probably know, calculus is all about functions. So you'll learn to define a function first. A function is defined by writing: `<function name>[<argument name>_, ...]:=<function>`. So for example `f[x_,y_]:=(x+y)^2` is the syntax for the function $f(x,y) = (x+y)^2$.

» Define the function $f(x,y,z) = \ln(x^2) \cdot \sin(y \cdot \cos(e^z))$ and evaluate it at $(4,7,2)$ and $(1,2,a)$.

As you might have noticed, the evaluation goes faster than doing it by hand. But much more is possible, so let's take a look at derivatives. For functions of one variable like `f[x_]:=x^2` just type in `f'[x]` and the output will be `2x`. But this doesn't work for our function $f(x,y,z)$. For this function you have to use `D`, this function can calculate partial derivatives over variables in the order you want it to. Total derivatives can also be calculated by using `Dt`. Lastly you can also use `Grad` to calculate the gradient of a function.

» Calculate $\partial_{x,y,z} f(x,y,z)$ and the tenth partial derivate over x of $f(x,y,z)$ (without typing $x,x,x,x,...$). Also find the total derivate of $f(x,y,z)$ over x and z.

» Calculate the gradient of $\sin(x^2+y^2)$ in both dimensions.

But Mathematica is not only capable of differentiation but also of integration. In a symbolic (`Integrate`) and a numerical way (`NIntegrate`). For example `Integrate[Cos[x],x]` gives the indefinite integral $\int \cos(x)\mathrm{d}x$, whereas `Integrate[E^x,{x,0,1}]` gives the definite integral $\int_0^1 e^x dx$. But for an integral like $\int_0^1 \sin(\sin(x))\mathrm{d}x$ you want to use `NIntegrate`, for this integral has no symbolic solution.

» Calculate the integral (using both functions) of $y * \ln x$ over an interval of your choice. Now calculate it over the area of the unit disk centered at $(2,2)$. (Hint: Use `Disk` and `Element`.)

4 Number Theory

Basic documentation
The `N` function provides a numerical representation until a given a number of digits.

» Find the $10,000^{\text{th}}$ decimal digit of $\pi/6$.

The function `PrimeQ` checks for a number if it is a prime. Using `NextPrime` you can find the next prime number. Also, using `FactorInteger` you can find the prime decomposition of a number.

» Find out which of the numbers 13, 113, 1113, 111111113 and 123456789012345678901 are prime. If a number is not a prime, find the prime decomposition and the next prime number.

5 Linear Algebra

Basic documentation
In the Wolfram Language, a matrix is simply a list of lists. A matrix can be entered in multiple ways. Using the Math Assistant, click the Matrix symbol. It is also possible to simply type `(` and `)` and add more rows and columns, using ctrl + , and ctrl + ↵.

» Enter the matrix $\begin{pmatrix} 1 & 2 & 3 \\ 2 & 4 & 6 \\ 2 & 3 & 5 \end{pmatrix}$.

Matrix/vector multiplication can be done using the function `Dot`, or simply the `.` symbol. To display the result as a matrix, use the function `MatrixForm` (be careful, you cannot use the result of `MatrixForm` to calculate with). Find the determinant of a matrix using `Det`.

» Multiply the matrices $\begin{pmatrix} 1 & e \\ \pi & i \end{pmatrix}$ and $\begin{pmatrix} \sqrt{e} & \pi^2 \\ i & -i \end{pmatrix}$ and find the determinant of the result.

A decomposition of a matrix can be handy. Use the functions `LUDecomposition`, `QRDecomposition` and `SingularValueDecomposition` to find the LU, QR and SV decompositions respectively.

» Find the LU, QR and SV decompositions of a 5×5 Hilbert matrix. (*Hint: use the* `HilbertMatrix` *function.*)

The dimensions, kernel (null space), and matrix rank can be deduced by the functions `Dimensions`, `NullSpace` and `MatrixRank` respectively.

» Find the dimensions, matrix rank and null space of the matrix $\begin{pmatrix} 1 & 1 \\ 1 & 1 \end{pmatrix}$.

To solve a system of linear equalities $Ax = b$, use the function `LinearSolve`, with as arguments the matrix A and the vector b. For solving the same linear system A for many bs, use the function `LinearSolve` with only the matrix A as an argument. It will return a reusable function which accepts vector b and solves the system.

» Solve the system of linear equations $Hx = b$ with H the 5×5 Hilbert matrix and b the vector $(1,2,3,4,5)^T$.

6 Plotting & Visualization

Basic documentation
A lot of plotting and visualization is possible in Mathematica, however the focus of this part is the plotting and visualization of functions. The basic function for this is `Plot`, for the plotting of one-variable functions.
The general syntax for plotting a function is `Plot[<fns>, {x, x_min, x_max}, <options>]`, where the options can consist of a legend, labeling, filling, etcetera. So for example the option `PlotLegends -> "Expressions"` gives a legend of the functions.

» Plot the functions $e^x + 5$ and $\sin(x)e^x$, with $0 \leq x \leq 3$ and use the option `Filling` between these two lines.

Also multiple-variable functions can be plotted using `Plot3D`. The syntax is similar to that of `Plot`. For these kind of functions also `ContourPlot` is a good way of showing its outlines.

» Plot $x^2 + y^2$ and $x^2 - y^2$ with both x and y between 0 and 5. Also make a contour plot of $\cos(x) - \sin(y)$ showing a legend as well.

The plotting functions above were the main functions, more can be found in the documentation.
But all these plots are static, you can't shift your constants and see the graph change. Fortunately, there are functions which help you do so: `Manipulate` and `Animate`.
The syntax of `Manipulate` is `Manipulate[<expr>, {u, u_min, u_max}]`, where `<expr>` is the expression you want to manipulate (i.e. function, plot) and u is the variable you want to play with.
The syntax of `Animate` is similar, the difference is that animate creates a playing movie and with `Manipulate` you can enter the value of the variable yourself (so more freedom).

» Use both `Manipulate` and `Animate` on the functions $\sin(x)$ and `Factor[a*x^2+b*x+c]` (with play variable a, b and c) and also try to `Manipulate` the plot functions of the previous exercise.

7 (Partial) Differential Equations

Basic documentation
The basic function to solve a differential equation is `DSolve`. We will look into multiple flavours of this function, suited for different purposes.
The general syntax for solving differential equations is `DSolveValue[<eqns>, <variables>, <time>]`, as an example the equation

$$x''(t) = 1 \qquad x'(0) = a, x(0) = b$$

which can be solved using `DSolveValue[{x''[t]==1, x'[0]==a, x[0]==b}, x[t], t]`.

» Solve the differential equation

$$u''(t) = -4\sin(t) + 5e^{-10t} + \cos(t)\sin(\sqrt{2}t)$$

with $u'(0) = 2$ and $u(0) = a$ using `DSolveValue` and plot the result.

Something must be said of the solutions that are found. The differential equation

$$u'(t) - t\sqrt{u(t)} = 0 \qquad u(0) = 0$$

has two solutions, namely $u(t) = 0$ and $u(t) = t^4/16$. However, only the latter is found.
Let us look at another differential equation:

$$u''(t) = \sqrt{3u(t)} \qquad u'(0) = 1, u(0) = 3.$$

This equation cannot be solved analytically by Mathematica (try it!). To solve it numerically, we use the function `NDSolveValue` instead. The equation can be solved using `NDSolveValue[{u''[t]==Sqrt[u[t]], u'[0]==1, u[0]==3}, u[t], {t, 0, 10}]`. Notice that the bounds for t have been given as $0 \leq t \leq 10$.

» Solve and plot the differential equation

$$u''(t) = -u(t)\cos(u(t)/5) - 2 \qquad u'(0) = 0, u(0) = 1$$

for $0 \leq t \leq 10$ with `NDSolveValue` and plot the result.

We can take this functionality to a different level by solving *partial* differential equations (PDEs) with the same code.
As an example we take the heat equation:

$$u_t = u_{xx} \qquad u(x,0) = u_0(x) = e^{-x^2}.$$

The solution is given by `DSolveValue[{D[u[x, t], t] == D[u[x, t], {x, 2}], u[x, 0] == Power[E,-x*x]}, u[x, t], {x, t}]`. The `D[...]` gives one or more derivatives of a function of more than one variable.

» Solve the partial differential equation

$$u_{tt} = u_{xx} \qquad u(x,0) = u_0(x) = \sin(x).$$

and plot the result using `DensityPlot` and `Plot3D` over the domain $(x,t) \in [-6,6] \times [0,4]$.

Of course, most partial differential equations found in practice are too hard to solve symbolically/analytically. For this we use the numerical variant of `DSolveValue`: `NDSolveValue`, just like with ordinary differential equations.

» Solve the partial differential equation

$$u_{tt} = \cos(t)u_x + \frac{1}{4}xtu$$

over the domain $(x,t) \in [-1,1] \times [0,1]$ and plot the result using `DensityPlot` and `ContourPlot`.

8 System & Control Theory

Basic documentation
Mathematica can represent/model a mathematical system of the form

$$x' = Ax + B, \quad y = Cx + D,$$

using two forms: the *transfer function* or the *state space*. The first can be created using the function `TransferFunctionModel`. For example the transfer function $1/s$, an integrator is represented by `TransferFunctionModel[1/s,s]`.

» Create a transfer function model for the transfer function $\frac{1}{0.5+s} + \frac{1}{3+s}$.

The other form is the state space model which can be created using the function `StateSpaceModel`. For example the system $x' = Ax + B, y = Cx + D$ can be created using `StateSpaceModel[{A, B, C, D}]`.
These models can be created in many ways: from equations, the matrix form and each others representation.

» Create a state space model from the model of the previous exercise.

Mathematica can determine a number of properties of a system. For a state space model, *observability* and *controllability* can be determined using the functions `ObservableModelQ` and `ControllableModelQ` respectively.

» Find out if the model is observable and controllable, using the state space representation.

You can simulate the output of the system for a given input using the function `OutputResponse`. It takes the model, the input function and the time range to simulate as input, for example `OutputResponse[model, UnitStep[t], {t, 0, 10}]`. You can plot the result.

» Simulate the model for the input function $\sin(5t) + \sin(t)$ for $0 \le t \le 15$.

Some properties of a system can be found by looking at special plots of the system. For example *Bode plots* and *Nyquist plots* can be generated using the functions `BodePlot` and `NyquistPlot` respectively.

» Generate Bode and Nyquist plots of the model of the previous exercises.

Finally we want to control the system to converge to the input function as quickly and robustly as possible. We find a PI and PID controller using the function `PIDTune`, which takes as arguments the model, the name of the controller we want and finally `"ReferenceControl"` to indicate we want the controlled system as output. As an example we find a PID controller for the model by calling `PIDTune[model, "PID", "ReferenceControl"]`.
To simulate the response, we can use `OutputResponse` again. (In case the plot does not show the output, set the option `Evaluated` to `True` for the `Plot` function.)

» Find a PID and PI controller of the model, and plot the controlled output response with `UnitStep[]` as input, both in the same plot.

9 Graph Theory

Basic documentation
The possibillities of Mathematica in combination with graphs are endless. But we'll only brief overview of the most useful functions here. We'll start with creating graphs. For this the functions `Graph`, `CompleteGraph`, `PlanarGraph` and `RandomGraph` are useful. Vertices in a graph can be named any way you like. Edges are constructed with arrows, so `1<->2` is an undirected edge between 1 and 2 and `2->1` is a directed edge from 2 to 1. So to construct a complete graph on three vertices, you use: `Graph[{1,2,3},{1<->2,2<->3,3<->1}]` or `CompleteGraph[3], VertexLabels->Automatic` for a labeled complete graph. There are a lot more options then `VertexLabels`, but this is the most useful one.

» Design a complete bipartite (2-partite), planar and random graph with each at least 10 vertices. Also add weight to vertices using the option `VertexWeight`.

Of course these graphs can be investigated about certain properties such as: `HamiltonianGraphQ`, `BipartiteGraphQ`, `AcyclicGraphQ`. There are also functions to calculate the automorphisms of a graph: `GraphAutomorphismGroup[G]` shows all the automorphisms of the graph G.

» Check whether or not your bipartite graph is Hamiltonian, bipartite and/or acyclic. Also use the functions `GraphAutomorphismGroup` and `GroupOrder` (look in the documentation to see how this works) to calculate the number of automorphisms of the random graph.

You can also do a depth first scan (DFS) of a graph by using `DepthFirstScan`, as well as the maximum flow and the shortest path between two edges, by using `FindMaximumFlow` respectively `FindShortestPath`.

» Perform a DFS on a gridgraph of more than 20 vertices and highlight it on the graph. (Hint: Use the documentation.) The highlighting gives a graphical view on the DFS.

» Find the maximum flow and a shortest path of a random graph, and look whether the output makes sense. Now add some weights to the graph and see what happens to the output of the shortest path algorithm.

Lastly their are some more useful functions which will save you a lot of time compared to doing it by hand: `FindClique`, `FindVertexCover` and `ChromaticPolynomial`. The first finds the largest clique of a graph, the second finds a minimal vertex cover of a graph and the last calculates the chromatic polynomial which helps you finding al the colorings of a graph.

» Make another random graph and find the largest clique and a minimal vertex cover. Also try to highlight it in the graph. (Hint: Use `HighlightGraph` and `Subgraph`). Lastly, calculate the chromatic polynomial of the graph.

10 Optimization

Basic documentation
One of the most well-known optimization problems probably is the *traveling salesman problem*. The actual problem is about finding the tour in a graph that visits each vertex exactly once and minimizes the total distance. For this you use the function `FindShortestTour`.

» Calculate the shortest past on a connected `RandomGraph` on 10 vertices and at least 30 edges. Next highlight the path on the graph. (Hint: Use `HighlightGraph` and `PathGraph`.)

Totally different situations can also be solved with this (and similar) functions. They can be found in the documentation, as well as some clear examples.
In the area of optimization more is possible, namely the minimization and maximization of a function. This can be done symbolically by using `Minimize` or `Maximize` and numerically by `NMinimize` or `NMaximize`.

» Minimize the following formula: $ax^2 + bx + c$ as a function. And after that calculate it for $a = 1$, $b = 3$, $b = 4$.

» Maximize $-x - y$ subject to $x + 2y \geq 3$ and $x, y \geq 0$ in the numerical and symbolical way. For the numerical way add the constraint $y \in \mathbb{N}$. (Hint: use `Integers`).

As you already noticed Mathematica can optimize functions with integer valued variables (at least numerically). But Mathematica can also be used for more difficult cases with the command `LinearProgramming`. This function is generally defined as `LinearProgramming[<c>, <A>, <b>]` for a problem where $c^T \cdot x$ is minimized subject to $A \cdot x \geq b$ and $x \geq 0$.

» Minimize $-x + y - 3z$ subject to $x + 2y + 10z = 0$, $x - z \leq 0$ and $y + z \geq 1$. (Hint: Use the documentation.)

11 Bonus

The following sections are bonus material. It will not be discussed during the course. However, it shows some fine applications in which Mathematica can help or entertain you besides being used for mathematics.

11.1 Image manipulation

Basic documentation
You can import images (actually any file), either from your hard drive or from the web using `Import`. The result will be an `Image` object, which can be manipulated.

» Import the image `https://www.abacus.utwente.nl/bundles/abacuswebsite/image/layout/logo.png` and save it in a variable for later use.

You can rescale, rotate and transform the image using the functions `ImageResize`, `ImageRotate` and `ImageTransform` respectively.

» Resize the image to 200 × 50 pixels, rotate it 135° and finally transform it using the `Sqrt` function. The result should be  .

Using machine learning, the content of an image can be found. The function `ImageIdentify` can do this for you. The initial initialization can take a while because image data has to be downloaded from the Wolfram Research server.

» Find the content of the image `http://www.spyderonlines.com/images/wallpapers/random-picture/random-picture-13.jpg`

Finally, the result can be saved using the function `Export`, for example `Export[<path>, image]` where `<path>` is the location on your computer.

11.2 Data (cities and Pokémon)

Basic demonstration
Wolfram has acquired a lot of public data over time, which can be used, real time, in your Mathematica notebooks. To show some examples how you can use this data, we demonstrate the interaction of Mathematica with data of cities around the world and of Pokémon.
A piece of data (a datum) is called an *Entity*. An entity has Properties, which can be queried.
We first focus on cities. Find an entity by pressing ctrl + =, entering a name of a city, and pressing ↵. The found entity should have a ✓ next to it and be highlighted in yellow.

» Find the entity *Enschede*.

You can find all properties of an entity by using the function `EntityProperties`. You can find the value of a specific property by using the function `EntityValue`, and giving the entity and the name of the property as arguments.

» Find the properties of the entity *Enschede*. Also find the values of the population, coordinates and the elevation of the city.

Finally, an entire class of values can be queried by giving an *Entity class* as an argument to `EntityValue`. An entity class can be found the same way as for a single entity.

» Find the entity class *Pokémon*.

Using `EntityValue`, one or more properties of an entire class can be queried.

» Find the names of all the Pokémon in the class *Pokémon*.

» Make a histogram using the function `Histogram` of the *attack* and *defense* power of all Pokémon in the class *Pokémon*, in the same plot.