Chapter 2

Mathematica in 15 min

Mathematica is a glorified calculator. Here is how to use it[1].

2.1 Basic Syntax

- Symbols `+, -, /, ^, *` are all supported by *Mathematica*. Multiplication can be represented by a space between variables. `a x + b` and `a*x + b` are identical.
- **Warning:** *Mathematica* is case-sensitive. For example, the command to exit is `Quit` and not `quit` or `QUIT`.
- Brackets are used around function arguments. Write `Sin[x]`, not `Sin(x)` or `Sin{x}`.
- Parentheses `( )` group terms for math operations: `(Sin[x]+Cos[y])*(Tan[z]+z^2)`.
- If you end an expression with a `;` (semi-colon) it will be executed, but its output will not be shown. This is useful for simulations, e.g.
- Braces `{ }` are used for lists:

```
In[1]:= A = {1, 2, 3}
Out[1]= {1, 2, 3}
```

- Names can refer to variables, expressions, functions, matrices, graphs, etc. A name is assigned using `name = object`. An expression may contain undefined names:

```
In[5]:= A = (a + b)^3
Out[5]= (a + b)^3
In[6]:= A^2
Out[6]= (a + b)^6
```

[1]Actually, this is just a tip of the iceberg. It can do many many many other things.

- The percent sign % stores the value of the previous result

```
In[7]:= 5 + 3
Out[7]= 8
In[8]:= %^2
Out[8]= 64
```

2.2 Numerical Approximation

- `N[expr]` gives the approximate numerical value of expression, variable, or command:

```
In[9]:= N[Sqrt[2]]
Out[9]= 1.41421
```

- `N[%]` gives the numerical value of the previous result:

```
In[17]:= E + Pi
Out[17]= e + π
In[18]:= N[%]
Out[18]= 5.85987
```

- `N[expr,n]` gives n digits of precision for the expression `expr`:

```
In[14]:= N[Pi, 30]
Out[14]= 3.14159265358979323846264338328
```

- Expressions whose result can't be represented exactly don't give a value unless you request approximation:

```
In[11]:= Sin[3]
Out[11]= Sin[3]
In[12]:= N[Sin[3]]
Out[12]= 0.14112
```

2.3 Expression Manipulation

- `Expand[expr]` (algebraically) expands the expression `expr`:

```
In[19]:= Expand[(a + b)^2]
Out[19]= a^2 + 2 a b + b^2
```

- `Factor[expr]` factors the expression `expr`

```
In[20]:= Factor[a^2 - b^2]
Out[20]= (a - b) (a + b)
In[21]:= Factor[x^2 - 5 x + 6]
Out[21]= (-3 + x) (-2 + x)
```

- `Simplify[expr]` performs all kinds of simplifications on the expression `expr`:

```
In[35]:= A = x / (x - 1) - x / (1 + x)
```

$$\text{Out[35]=}\quad \frac{x}{-1+x} - \frac{x}{1+x}$$

```
In[36]:= Simplify[A]
```

$$\text{Out[36]=}\quad \frac{2x}{-1+x^2}$$

2.4 Lists and Functions

- If `L` is a list, its length is given by `Length[L]`. The n^{th} element of `L` can be accessed by `L[[n]]` (note the double brackets):

```
In[43]:= L = {2, 4, 6, 8, 10}
Out[43]= {2, 4, 6, 8, 10}
In[44]:= L[[3]]
Out[44]= 6
```

- Addition, subtraction, multiplication and division can be applied to lists element by element:

```
In[1]:= L = {1, 3, 4}; K = {3, 4, 2};
In[2]:= L + K
Out[2]= {4, 7, 6}
In[3]:= L / K
```

$$\text{Out[3]=}\quad \left\{\frac{1}{3}, \frac{3}{4}, 2\right\}$$

- If the expression `expr` depends on a variable (say `i`), `Table[expr,{i,m,n}]` produces a list of the values of the expression `expr` as `i` ranges from `m` to `n`

```
In[37]:= Table[i^2, {i, 0, 5}]
Out[37]= {0, 1, 4, 9, 16, 25}
```

- The same works with two indices - you will get a list of lists

```
In[40]:= Table[i^j, {i, 1, 3}, {j, 2, 3}]
Out[40]= {{1, 1}, {4, 8}, {9, 27}}
```

- It is possible to define your own functions in *Mathematica*. Just use the *underscore syntax* `f[x_]=expr`, where `expr` is some expression involving `x`:

```
In[47]:= f[x_] = x^2
Out[47]= x^2
In[48]:= f[x + y]
Out[48]= (x + y)^2
```

- To apply the function `f` (either built-in, like `Sin`, or defined by you) to each element of the list `L`, you can use the command `Map` with syntax `Map[f,L]`:

```
In[50]:= f[x_] = 3 * x
Out[50]= 3 x
In[51]:= L = {1, 2, 3, 4}
Out[51]= {1, 2, 3, 4}
In[52]:= Map[f, L]
Out[52]= {3, 6, 9, 12}
```

- If you want to add all the elements of a list L, use `Total[L]`. The list of the same length as L, but whose k^{th} element is given by the sum of the first k elements of L is given by `Accumulate[L]`:

```
In[8]:= L = {1, 2, 3, 4, 5}
Out[8]= {1, 2, 3, 4, 5}
In[9]:= Accumulate[L]
Out[9]= {1, 3, 6, 10, 15}
In[10]:= Total[L]
Out[10]= 15
```

2.5 Linear Algebra

- In *Mathematica*, matrix is a nested list, i.e., a list whose elements are lists. By convention, matrices are represented row by row (inner lists are row vectors).
- To access the element in the i^{th} row and j^{th} column of the matrix `A`, type `A[[i,j]]` or `A[[i]][[j]]`:

```
In[59]:= A = {{2, 1, 3}, {5, 6, 9}}
Out[59]= {{2, 1, 3}, {5, 6, 9}}
In[60]:= A[[2, 3]]
Out[60]= 9
In[61]:= A[[2]][[3]]
Out[61]= 9
```

- `Matrixform[expr]` displays `expr` as a matrix (provided it is a nested list)

```
In[9]:= A = Table[i * 2^j, {i, 2, 5}, {j, 1, 2}]
Out[9]= {{4, 8}, {6, 12}, {8, 16}, {10, 20}}
In[10]:= MatrixForm[A]
Out[10]//MatrixForm=
```

$$\begin{pmatrix} 4 & 8 \\ 6 & 12 \\ 8 & 16 \\ 10 & 20 \end{pmatrix}$$

- Commands `Transpose[A]`, `Inverse[A]`, `Det[A]`, `Tr[A]` and `MatrixRank[A]` return the transpose, inverse, determinant, trace and rank of the matrix `A`, respectively.
- To compute the n^{th} power of the matrix `A`, use `MatrixPower[A,n]`

```
In[21]:= A = {{1, 1}, {1, 0}}
Out[21]= {{1, 1}, {1, 0}}
In[22]:= MatrixForm[MatrixPower[A, 5]]
Out[22]//MatrixForm=
```

$$\begin{pmatrix} 8 & 5 \\ 5 & 3 \end{pmatrix}$$

- Identity matrix of order n is produced by `IdentityMatrix[n]`.
- If `A` and `B` are matrices of the same order, `A+B` and `A-B` are their sum and difference.

- If `A` and `B` are of compatible orders, `A.B` (that is a dot between them) is the matrix product of `A` and `B`.
- For a square matrix `A`, `CharacteristicPolynomial[A,x]` is the characteristic poynomial, $\det(xI - A)$ in the variable `x`:

```
In[40]:= A = {{3, 4}, {2, 1}}
Out[40]= {{3, 4}, {2, 1}}
In[42]:= CharacteristicPolynomial[A, x]
Out[42]= -5 - 4 x + x^2
```

- To get eigenvalues and eigenvectors use `Eigenvalues[A]` and `Eigenvectors[A]`. The results will be the list containing the eigenvalues in the `Eigenvalues` case, and the list of eigenvectors of `A` in the `Eigenvectors` case:

```
In[52]:= A = {{3, 4}, {2, 1}}
Out[52]= {{3, 4}, {2, 1}}
In[53]:= Eigenvalues[A]
Out[53]= {5, -1}
In[54]:= Eigenvectors[A]
Out[54]= {{2, 1}, {-1, 1}}
```

2.6 Predefined Constants

- A number of constants are predefined by *Mathematica*: `Pi`, `I` ($\sqrt{-1}$), `E` (2.71828...), `Infinity`. Don't use `I`, `E` (or `D`) for variable names - *Mathematica* will object.
- A number of standard functions are built into *Mathematica*: `Sqrt[]`, `Exp[]`, `Log[]`, `Sin[]`, `ArcSin[]`, `Cos[]`, etc.

2.7 Calculus

- `D[f,x]` gives the derivative of `f` with respect to `x`. For the first few derivatives you can use `f'[x]`, `f''[x]`, etc.

```
In[66]:= D[ x^k, x]
Out[66]= k x^(-1+k)
```

- `D[f,{x,n}]` gives the n^{th} derivative of `f` with respect to `x`
- `D[f,x,y]` gives the mixed derivative of `f` with respect to `x` and `y`.

- `Integrate[f,x]` gives the *indefinite* integral of `f` with respect to `x`:

```
In[67]:= Integrate[Log[x], x]
Out[67]= -x + x Log[x]
```

- `Integrate[f,{x,a,b}]` gives the *definite* integral of `f` on the interval $[a, b]$ (`a` or `b` can be `Infinity` (∞) or `-Infinity` ($-\infty$)) :

```
In[72]:= Integrate[Exp[-2*x], {x, 0, Infinity}]
```

Out[72]= $\frac{1}{2}$

- `NIntegrate[f,{x,a,b}]` gives the numerical approximation of the definite integral. This usually returns an answer when `Integrate[]` doesn't work:

```
In[76]:= Integrate[1/(x+Sin[x]), {x, 1, 2}]
```

Out[76]= $\int_1^2 \frac{1}{x + \text{Sin}[x]} \, dx$

```
In[77]:= NIntegrate[1/(x+Sin[x]), {x, 1, 2}]
Out[77]= 0.414085
```

- `Sum[expr,{n,a,b}]` evaluates the (finite or infinite) sum. Use `NSum` for a numerical approximation.

```
In[80]:= Sum[1/k^4, {k, 1, Infinity}]
```

Out[80]= $\frac{\pi^4}{90}$

- `DSolve[eqn,y,x]` solves (given the general solution to) an ordinary differential equation for function y in the variable x:

```
In[88]:= DSolve[y''[x] + y[x] == x, y[x], x]
Out[88]= {{y[x] → x + C[1] Cos[x] + C[2] Sin[x]}}
```

- To calculate using initial or boundary conditions use `DSolve[{eqn,conds},y,x]`:

```
In[93]:= DSolve[{y'[x] == y[x]^2, y[0] == 1}, y[x], x]
```

Out[93]= $\left\{\left\{y[x] \to \frac{1}{1-x}\right\}\right\}$

2.8 Solving Equations

- Algebraic equations are solved with `Solve[lhs==rhs,x]`, where `x` is the variable with respect to which you want to solve the equation. Be sure to use `==` and not `=` in equations. *Mathematica* returns the list with all solutions:

```
In[81]:= Solve[x^3 == x, x]
Out[81]= {{x → -1}, {x → 0}, {x → 1}}
```

- `FindRoot[f,{x,x0}]` is used to find a root when `Solve[]` does not work. It solves for x numerically, using an initial value of x0:

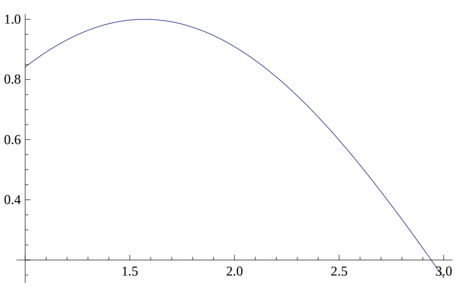
```
In[82]:= FindRoot[Cos[x] == x, {x, 1}]
Out[82]= {x → 0.739085}
```

2.9 Graphics

- `Plot[expr,{x,a,b}]` plots the expression `expr`, in the variable `x`, from a to b:

```
In[83]:= Plot[Sin[x], {x, 1, 3}]
```

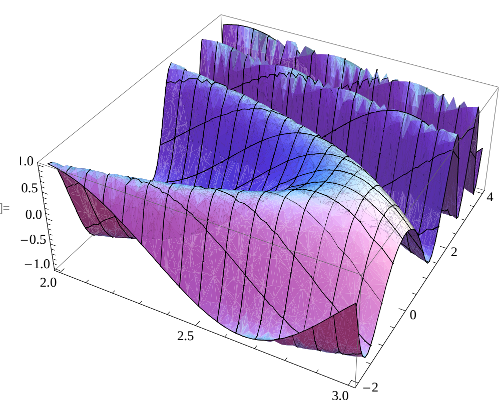
Out[83]=



- `Plot3D[expr,{x,a,b},{y,c,d}]` produces a 3D plot in 2 variables:

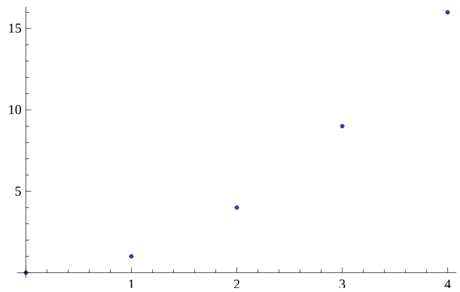
```
In[84]:= Plot3D[Sin[x^2 + y^2], {x, 2, 3}, {y, -2, 4}]
```

Out[84]=



- If `L` is a list of the form `L=` $\{\{x_1, y_1\}, \{x_2, y_2\}, \ldots, \{x_n, y_n\}\}$, you can use the command `ListPlot[L]` to display a graph consisting of points (x_1, y_1), ..., (x_n, y_n):

```
In[11]:= L = Table[{i, i^2}, {i, 0, 4}]
Out[11]= {{0, 0}, {1, 1}, {2, 4}, {3, 9}, {4, 16}}
In[12]:= ListPlot[L]
Out[12]=
```



2.10 Probability Distributions and Simulation

- `PDF[distr,x]` and `CDF[distr,x]` return the pdf (pmf in the discrete case) and the cdf of the distribution `distr` in the variable x. `distr` can be one of:
  - `NormalDistribution[m,s]`,
  - `ExponentialDistribution[l]`,
  - `UniformDistribution[{a,b}]`,
  - `BinomialDistribusion[n,p]`,

  and many many others (see `?PDF` and follow various links from there).

- Use `ExpectedValue[expr,distr,x]` to compute the expectation $\mathbb{E}[f(X)]$, where `expr` is the expression for the function f in the variable x:

```
In[23]:= distr = PoissonDistribution[λ]
Out[23]= PoissonDistribution[λ]
In[25]:= PDF[distr, x]
```

Out[25]= $\dfrac{e^{-\lambda}\lambda^x}{x!}$

```
In[27]:= ExpectedValue[x^3, distr, x]
```

Out[27]= $\lambda + 3\lambda^2 + \lambda^3$

- There is no command for the generating function, but you can get it by computing the characteristic function and changing the variable a bit `CharacteristicFunction[distr, - I Log[s]]`:

```
In[22]:= distr = PoissonDistribution[λ]
Out[22]= PoissonDistribution[λ]
In[23]:= CharacteristicFunction[distr, -I Log[s]]
Out[23]= e^((-1+s) λ)
```

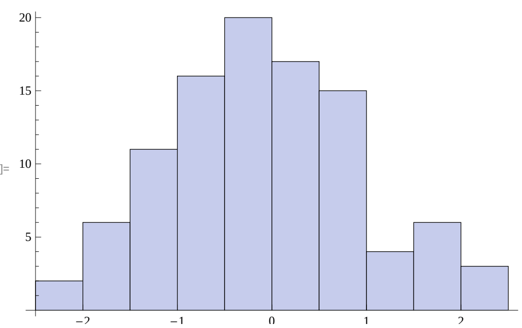
- To get a random number (unifomly distributed between 0 and 1) use `RandomReal[]`. A uniformly distributed random number on the interval $[a, b]$ can be obtained by `RandomReal[{a,b}]`. For a list of n uniform random numbers on $[a, b]$ write `RandomReal[{a,b},n]`.

```
In[2]:= RandomReal[]
Out[2]= 0.168904
In[3]:= RandomReal[{7, 9}]
Out[3]= 7.83027
In[5]:= RandomReal[{0, 1}, 3]
Out[5]= {0.368422, 0.961658, 0.692345}
```

- If you need a random number from a particular *continuous* distribution (normal, say), use `RandomReal[distr]` or `RandomReal[distr,n]` if you need n draws.
- When drawing from a *discrete* distribution use `RandomInteger` instead.
- If `L` is a list of numbers, `Histogram[L]` displays a histogram of `L` (you need to load the package *Histograms* by issuing the command `<<Histograms‘` before you can use it):

```
In[7]:= L = RandomReal[NormalDistribution[0, 1], 100];
In[10]:= << Histograms`
In[12]:= Histogram[L]
```

Out[12]=



2.11 Help Commands

- `?name` returns information about `name`
- `??name` adds extra information about `name`
- `Options[command]` returns all options that may be set for a given command

- `?pattern` returns the list of matching names (used when you forget a command). pattern contains one or more asterisks `*` which match any string. Try `?*Plot*`

2.12 Common Mistakes

- *Mathematica* is case sensitive: `Sin` is not `sin`
- Don't confuse braces, brackets, and parentheses `{}`, `[]`, `()`
- Leave spaces between variables: write `a x^2` instead of `ax^2`, if you want to get ax^2.
- Matrix multiplication uses `.` instead of `*` or a space.
- Don't use `=` instead of `==` in `Solve` or `DSolve`
- If you are using an older version of *Mathematica*, a function might be defined in an external module which has to be loaded before the function can be used. For example, in some versions, the command `<<Graphics'` needs to be given before any plots can be made. The symbol at the end is *not* an apostrophe - it is the dash above the TAB key.
- Using `Integrate[]` around a singular point can yield wrong answers. (Use `NIntegrate[]` to check.)
- Don't forget the underscore `_` when you define a function.