# Part II. Calculations and Calculus with *Mathematica*

## Section 1. Getting Started: *Mathematica* as a Calculator

*Mathematica* is one of a handful of computer software applications that "does mathematics". It works following an input/output paradigm. The user types the input in an Input cell, *Mathematica* processes the input and returns the result, referred to as the output. An input cell appears automatically at the end of every worksheet and can be obtained anywhere by clicking between cells.

To add 1 and 2 below, we clicked below this text cell, typed "1+2" and then pressed the [shift]-[return] key. PC users, press the [shift]-[Enter] key.

The input is displayed on the computer screen in a bold monospaced font. The output appears directly below the input.

```
1+2
```

```
3
```

**About the input**. When an input entry is made with no trailing punctuation *Mathematica* processes it and places the output in an output cell. Several entries can be made in one input cell. Simply press the [return] key and type in another entry. If a particular entry is terminated with a semicolon ( ; ) then *Mathematica* will process the entry but will not show the output. The output is said to be suppressed.

The next input cell contains three entries asking for three square roots. Multiplication is specified with an asterisk * (shift-8) and exponentiation with the carat symbol ^ (shift-6). The multiplication sign can be omitted and replaced by a space if so desired. The square root function is denoted Sqrt (remember, square brackets must be used for all *Mathematica* functions)j.

```
Sqrt[2*3]
Sqrt[5];
Sqrt[3^4]
```

```
√6
```

```
9
```

All three square roots were processed, but only the first and third outputs are shown because the second entry terminated with a semicolon suppressing the output.

The entry Sqrt[3*2] outputs as $\sqrt{6}$ , the exact value. *Mathematica* always attempts to return exact values, if exact values are entered. Compare the first output to the output for the entry

Sqrt[2*3.0]

```
Sqrt[2*3.0]
```

```
2.44949
```

*Mathematica* interprets Sqrt[2*3.0] as an approximate input and returns an approximate output in what is called "floating point form" (6 digit accuracy is the default). It is often desirable to have floating point output even though the input is exact. In that case use the N function ( evaluate as a floating point number) as follows.

```
(Sqrt[20]+Sqrt[30])/(Sqrt[5]+Sqrt[7])
```

$$\frac{2\sqrt{5}+\sqrt{30}}{\sqrt{5}+\sqrt{7}}$$

This is an exact value. The entry

N[%]

will output the number in 6 digit floating point form. The percent sign refers to the most recent output.

```
N[%]
```

```
2.03804
```

If a 4 digit approximation is adequate, then enter the following. Two percent signs refer to the second to most recent output.

```
N[%%,4]
```

```
2.038
```

The next entry generates a 60 digit approximation to $\pi$. Note that *Mathematica* uses

Pi

to denote the mathematical constant $\pi$. Be sure to capitalize the p.

```
Pi
N[Pi,60]
```

```
π

3.14159265358979323846264338327950288419716939937510582097494
```

Note that the output for Pi is $\pi$.

A 10 digit approximation to the famous number *e* appears next. There are two ways to get *Mathematica* to output *e*. One way is to enter E. The other way is to apply the exponential function Exp to the number 1. These ideas are illustrated below.

```
E
N[E,10]
```

```
ⅇ

2.718281828
```

Note that *Mathematica* prints e in its output as ⅇ . The same output can be obtained as follows

```
Exp[1]
N[Exp[1],10]
```

```
ⅇ

2.718281828
```

### Parentheses for grouping must be the round kind: ( )

Parentheses are often essential when entering complicated expressions. For example, the calculation of

$$\frac{2^{50}-1}{2^{49}+1}$$

must be made as follows.

```
(2^50-1)/(2^49+1)
```

$$\frac{375299968947541}{187649984473771}$$

As usual, *Mathematica* outputs the exact value. Here is the same number in floating point form (10 digits).

```
N[%,10]
```

```
2.000000000
```

And here is the 20 digit floating point form.

```
N[%%,20]
```

```
1.9999999999999946709
```

When grouping terms in an entry, never use square brackets, [ ] , or set brackets, {}. Square brackets are reserved in *Mathematica* to enter arguments in functions and set brackets are used to make lists of things like

{ milk, eggs, sugar, butter }

Likewise, all of the familiar functions that you know from your calculus days should be entered into *Mathematica* using square brackets and they must be capitalized. *Mathematica* is case sensitive.

```
x=Sin[Pi/3] - Log[2] + Exp[4]
```

$$\frac{\sqrt{3}}{2} + e^{4} - \text{Log}[2]$$

Observe that *Mathematica* simplified Sin[Pi/3] because it has a nice exact form. The following entry puts *x* into floating point form (5 digits).

```
N[x,5]
```

```
54.771
```

Log[2] is the *Mathematica*'s notation for the base *e* or *natural* logarithm. If a base b logarithm is needed, b positive and not 1, use square brackets to enter

Log[b,x]

as illustrated below.

```
Log[10,100]
Log[2,256]
```

```
2

8
```

## Lists in *Mathematica*

Several expressions, entered as a sequence separated by commas, and enclosed in set brackets, is called a *list*. Lists can be made easily using the Table function. These ideas are illustrated below.

The next input cell enters the list of the first four factorials.

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

```
{1, 2, 6, 24}
```

Note that *Mathematica* calculates the values and outputs the values in the same list brackets. Any list can be processed like this. If floating point form is desired apply N.

```
{Sin[1], Sin[2], Sin[3] }
N[%,2]
```

```
{Sin[1], Sin[2], Sin[3]}
```

```
{0.84, 0.91, 0.14}
```

**Important Observation**: *Mathematica* assumes that the arguments in a trigonometric function are in radians, never degrees.

## The Table function

The two lists above are of the form { f(1), f(2), f(3), ..., f(n) }. Such a list can be obtained quickly and easily using the Table function in the form

**Table[f[k], {k,n}]**

```
Table[Sin[k],{k,3}]
```

```
{Sin[1], Sin[2], Sin[3]}
```

```
Table[n!, {n,4}]
```

```
{1, 2, 6, 24}
```

The nth term in a list L can be obtained with the entry L[[n]]. The next entry defines L to be a list and the following entry picks out the fourth entry in the list.

```
L=Table[3^k, {k,5}]
```

```
{3, 9, 27, 81, 243}
```

```
L[[4]]
```

```
81
```

Suppose that the elements in a list are themselves lists, like the following example.

```
L=Table[ {x,x/(1+x),Sin[x]}, {x,2,12,3}]
```

```
    2              5              8                11
{{2, -, Sin[2]}, {5, -, Sin[5]}, {8, -, Sin[8]}, {11, --, Sin[11]}}
    3              6              9                12
```

Then L[[m,n]] refers to the nth member of the mth list. See below

```
{L[[3,2]], L[[1,1]], L[[2,3]]}
```

$$\left\{\frac{8}{9},\ 2,\ \mathtt{Sin[5]}\right\}$$

### Using the Table function

The Table function is very handy. The following entries will give you a better idea of how it works.

1. Make the list of odd integers from 11 to 37.

```
Table[k, {k,11,37,2}]
```

```
{11, 13, 15, 17, 19, 21, 23, 25, 27, 29, 31, 33, 35, 37}
```

2. Make the sequence of numbers from 1/3 to 37/3 in increments of 4/3. Convert it to floating point form (5 digits).

```
Table[k, {k,1/3,37/3,4/3}]
N[%,5]
```

$$\left\{\frac{1}{3},\ \frac{5}{3},\ 3,\ \frac{13}{3},\ \frac{17}{3},\ 7,\ \frac{25}{3},\ \frac{29}{3},\ 11,\ \frac{37}{3}\right\}$$

```
{0.33333, 1.6667, 3.0000, 4.3333,
 5.6667, 7.0000, 8.3333, 9.6667, 11.000, 12.333}
```

3. Make a sequence of 20 prime integers starting at 7.

```
Table[Prime[k], {k,4,23}]
```

```
{7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47, 53, 59, 61, 67, 71, 73, 79, 83}
```

That was neat, how did it work?

*Mathematica* has a function called Prime, whose value at n, Prime[n], is the nth prime. For example, Prime[1] is 2, which is the first prime; 7 is the fourth prime, so Prime[4] = 7. *Mathematica* has other integer functions like this. For example, the PrimeQ[n] function returns true when n is prime and false otherwise.

```
PrimeQ[10001237]
```

```
True
```

```
PrimeQ[100001237]
```

```
False
```

4. Make a list of the binomial coefficients in the 15th row of Pascal's triangle.

```
Table[Binomial[15,n], {n,0,15}]
```

```
{1, 15, 105, 455, 1365, 3003, 5005, 6435, 6435, 5005, 3003, 1365, 455, 105, 15, 1}
```

Add them up.

```
Sum[%[[k]],{k,16}]
```

```
32768
```

Check that this sum is the same as $2^{15}$.

```
2^15
```

```
32768
```

Wow, what just happened?

The Sum function will add the first n terms in a list L via the syntax

**Sum[ L[[k]], {k,1,n} ]**

That's nice, but what I meant was why is the sum equal to $2^{15}$ ?

Oh that.Well, the 15th row of Pascal's triangle contains the coefficients in the expansion of the expression

$$(a+b)^{15}$$

Replacing the letters *a* and *b* with the number 1 makes the number $2^{15}$ and, at the same time, has the effect of adding the coefficients in the expansion of $(1+1)^{15}$.

## The Table function is very versatile

Examine the following examples to see the versatility of the Table function.

Table can make a list of lists as follows.

```
Table[a^m+b^n, {m,2,4}, {n,4,6} ]
```

$$\{\{a^2+b^4,\ a^2+b^5,\ a^2+b^6\},\ \{a^3+b^4,\ a^3+b^5,\ a^3+b^6\},\ \{a^4+b^4,\ a^4+b^5,\ a^4+b^6\}\}$$

Table can work with text like this.

```
L = {me,you,him,us,you,them};
Table[ (L[[k]]+TheKing)/L[[k]], {k,6}]
```

$$\left\{\frac{\text{me}+\text{TheKing}}{\text{me}},\ \frac{\text{TheKing}+\text{you}}{\text{you}},\ \frac{\text{him}+\text{TheKing}}{\text{him}},\ \frac{\text{TheKing}+\text{us}}{\text{us}},\ \frac{\text{TheKing}+\text{you}}{\text{you}},\ \frac{\text{TheKing}+\text{them}}{\text{them}}\right\}$$

Several lists can be joined together with the Join function. The next input cell first creates three lists (output suppressed) then joins them into one long list.

```
L1=Table[11,{3}]; L2=Table[4,{12}]; L3=Table[12,{13}];
S=Join[L1,L2,L3]
```

```
{11, 11, 11, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4,
 4, 12, 12, 12, 12, 12, 12, 12, 12, 12, 12, 12, 12, 12}
```

The Product function multiplies the numbers in the list named S.

```
Product[S[[k]],{k,28}]
```

```
2389209043962657308147712
```

FactorInteger generates the prime factorization.

```
FactorInteger[%]
{{2, 50}, {3, 13}, {11, 3}}
```

Interpret the output as $2^{50}\ 3^{13}\ 11^{3}$.

I get it, Product is like Sum, only it multiplies, right?

Right. See the **Help** page.

The **Help** page?

Yes. For example, to learn about the FactorInteger function you can enter

```
?FactorInteger

 FactorInteger[n] gives a list of the prime factors
    of the integer n, together with their exponents. More…
```

and click on More... for more information.

## Moving towards symbolic calculations: Factoring in *Mathematica*.

The last example shows that the FactorInteger function factors integers. *Mathematica*'s Factor function will factor polynomials. We will define a polynomial in the *x* variable named "Poly" below. However, before we define it we must free up the x variable since at the present time (in this Notebook) *x* is already in use.

```
x
```

$$\frac{\sqrt{3}}{2} + e^{4} - \text{Log}[2]$$

Free x with the entry

Clear[x]

or the entry

x = .

```
x = .
```

Here is the definition of Poly.

```
Poly = Sum[ 2^k*x^k, {k,4}]
```

$$2\,x + 4\,x^{2} + 8\,x^{3} + 16\,x^{4}$$

The Factor function will factor Poly over the integers.

```
Factor[Poly]
```

$$2\,x\,(1 + 2\,x)\,(1 + 4\,x^{2})$$

Add the optional expression

**Extension -> {I}**

to factor Poly over the complex numbers of the form *a* + *i b* where *a* and *b* are integers. These are called Gaussian integers. (The symbol I represents the complex number *i* , i.e. $\sqrt{-1}$ .)

```
Factor[Poly,Extension->{I}]
```

```
2 x (-ⅈ + 2 x) (ⅈ + 2 x) (1 + 2 x)
```

Note that in the output *Mathematica* uses the symbol ⅈ to represent *i* .

To see the zeros of Poly use the Solve function as follows.

```
Solve[Poly==0,x]
```

$$\left\{\left\{x \to -\frac{1}{2}\right\}, \{x \to 0\}, \left\{x \to -\frac{ⅈ}{2}\right\}, \left\{x \to \frac{ⅈ}{2}\right\}\right\}$$

Note two things.

1. The syntax for Solve is **Solve[ equation, variable]**. The equation to be solved must be in the form

lhs == rhs

to distinguish it from an equation defining the value of a variable.

2. The output is list of lists. Each list contains a solution in the form x -> a.

We will have more to say about Solve in the next section.