

What is Mathematica?

Mathematica is a software system and computer language for use in mathematical applications. The three classes of **Mathematica** computations are: numerical, symbolic, and graphical. **Mathematica** can be used as a calculator with a much higher degree of precision than traditional calculators. **Mathematica** can perform operations on functions, manipulate algebraic formulas, and do calculus. **Mathematica** is also able to produce both two- and three-dimensional graphs. **Mathematica** supports its own high-level programming language.

Mathematica is more than a calculations package. It is a symbolic, mathematical tool used by thousands of scientists around the world. **Mathematica** does not just do arithmetic, any program can do that.

Mathematica can do math. Algebra, calculus, differential equations, and many more features round out this remarkable software.

NOTE: Many of the examples in this tutorial were loosely based on examples found in the books *Mathematica*, by Stephen Wolfram, and *The Beginner's Guide to Mathematica Version 2*, by Theodore Gray and Jerry Glynn, both published by Addison-Wesley Publishing Company.

Mathematica: The Basics

Work in Mathematica takes place in the *kernel* or *notebook* (we will use these two words interchangeably throughout this tutorial). It is in the kernel where computations are actually performed. The kernel is much like a worksheet.

Once in Mathematica, there is a **Help** heading on the menu bar. Selecting this menu gives a list of help areas. You can also get help and information about specific commands (objects) by typing `?Name` to get information on Name, `??Name` to get extra information on Name, or `?Aaaa*` to get information on all objects beginning with Aaaa.

In this tutorial, examples will most often duplicate the notebook display in this form:

```
In[1]:=
      3 + 2
```

```
Out[1]=
      5
```

Be warned that the "`In[1]:=`" prompt will not actually appear until *after* you have entered your input, in this case `3 + 2`. You will see an unformatted screen until you enter **[SHIFT-RETURN]** to tell Mathematica to evaluate the input and return a result. Then you will see the *In* and *Out* prompts.

Mathematica can handle very large numbers. If a number is so large that it cannot be displayed on one line, Mathematica places a `\` (backslash) at the end of the line to show that the number continues on the next line.

If you've made a mistake, or a calculation is taking too long, you can interrupt the calculation. **[CONTROL-C]** will interrupt the calculation.

Numerical Calculations

Arithmetic

Arithmetic can be performed in Mathematica just as on a calculator.

```
In[1]:=
      2 + 6
Out[1]:=
      8
```

In this example, you would have typed `2 + 6` and then pressed **[SHIFT-RETURN]**.

The arithmetic operations in Mathematica are:

Notation	Operation
x^y	exponential
$-x$	subtraction
x/y	division
$x y z$ $x*y*z$ $(x)(y)(z)$	multiplication (note that an asterisk, parenthesis, or a space between the numbers or variables denotes multiplication)
$x+y+z$	addition

Mathematica follows the standard mathematical sequence of operations. This order can, naturally, be altered by the use of parenthesis, `()`.

Precision

Mathematica does not have a limited set on precision, like a calculator does. It gives you an exact result, often in symbolic or mathematical terms. If you prefer, Mathematica can give you an approximate numerical result given in scientific notation. You can even tell Mathematica to which degree of precision you would like the result.

To tell Mathematica that you want an exact result, simply type in your input. For example:

```
In[1]:=
      2 ^ 100
Out[1]:=
      1267650600228229401496703205376
```

To tell Mathematica you want an approximate numerical result, type `//N` following your input. For example:

```
In[2]:=
      2 ^ 100 //N
Out[2]:=
      1.26765 1030
```

If you would like your result to be calculated to a specific precision, you may tell Mathematica how many significant figures you want. Typing `N[your expression, n]` will give you the number *n* digits in the result.

Mathematica will keep expressions in symbolic format, just as you normally would on paper:

```
In[2]:=
      2*Pi^3
Out[2]=
      2 Pi3
```

To get a numerical value of this expression...

```
In[3]:=
      2*Pi^3 //N
Out[3]=
      62.0126
```

Keep in mind that Mathematica, when given an exact rational number as a fraction, returns that number reduced to its lowest terms. For example:

```
In[3]:=
      430/16
Out[3]:=
      215/8
```

Mathematica does NOT return 26.875 as the result. However, if you give a number containing an explicit decimal point, Mathematica will return an approximate numerical result. For example:

```
In[4]:=
      430./16
Out[4]:=
      26.875
```

Reminder: Mathematica will return an exact result if there are no explicit decimal points in the input. If there is an explicit decimal point in the input, Mathematica will give you an approximate numerical result.

Why? Mathematica assumes that an integer with no decimal point is exact, so the output returned to you is exact. If there is an explicit decimal point, Mathematica assumes that the number is not exact but only accurate to that number of decimal places, *i.e.*, the value has significant digits.

Mathematical Functions

Mathematica can compute square roots, handle complex numbers, evaluate mathematical functions, and calculate numerical integrals plus a host of other functions.

Some of the most common mathematical functions include: `Sqrt[x]`, the square root; `Log[x]`, the natural logarithm; `Sin[x]`, `Cos[x]`, `Tan[x]`, the trigonometric functions (arguments in radians); etc.

Something important to note about Mathematica functions: The arguments of the functions are in square brackets, `[]`, *not* parenthesis, `()`. Also, the names of functions begin with **Capital Letters**.

There are several built-in constants in Mathematica. Examples of these are `Pi`, `I`, `Infinity`, `E`, etc.

Using Previous Results

If you need to use the result you obtained in the previous calculation in your present calculation, you can simply type `%` in place of the entire previous result. For example:

```
In[1]:=
      7 * 10
Out[1]=
      70
```

```
In[2]:=
      % + 25
Out[2]=
      95
```

In fact, you can use a result an *n*th number of times previous. For example, `%%` is the next-to-last result. `%%%%` is the 8th previous result, etc.

Multiple Operations

You can do several operations on one line. By separating these operations with a semicolon (`;`) after all but the last, you are telling Mathematica to do each of the operations but only print the output of the last one. For example:

```
In[1]:=
      q = 10 ; r = 2 ; q - r
Out[1]=
      8
```

If at any time you would like to have Mathematica perform an operation but not print the output, place a semicolon after the expression. For example:

```
In[1]:=
      z = 9 * 5 ;
```

Now you can use the variable z in further calculations.

Variable Definition

In order to assign a specific value to a variable, simply tell Mathematica what you would like to name the variable and what value it should be assigned.

```
In[1]:=
      x = 2
Out[1]=
      2
```

This action defines the variable x to have the value 2. This value assignment is permanent until you remove it, or start a new Mathematica session.

Having assigned a value to x , you can now use x in further equations without having to ever type x 's value. For example:

```
In[1]:=
      x ^ 3
Out[1]=
      8
```

While variable definition is not particularly useful in this example, it becomes indispensable if you are dealing with numbers with 20, 30, or even more digits.

In naming variables, remember that Mathematica is case-sensitive. In other words, to Mathematica, there is a difference between the variable $aabb$ and the variable $aABb$. It is *unwise* to use variables beginning with a capital letter, because Mathematica's functions all begin with a capital letter, and you do not want your variable to be confused with a function name.

When using variables, the symbol for multiplication is either an asterisk, $*$, parenthesis $()$, or a space. For example, $x * y$, $x(y)$, and $x y$ are the same operation. If you type xy , with *no space*, Mathematica will treat the expression as a variable named xy . We would advise using the asterisk, $*$, for all multiplication operations, just to avoid confusion.

Summations and Products

Mathematica will handle summations and products easily. The notation is intuitive. For a simple product,

$\sum_{i=j \min}^{i \max} f$, the notation would be `Sum[f, {i, imin, imax}]`. For example, if we had this summation: $\sum_{i=1}^5 \frac{x}{\pi^i}$

```
In[2]:=
Sum[x/Pi^i, {i, 1, 5}]
```

```
Out[2]=
      x      x      x      x      x
  ---- + ---- + ---- + ---- + ----
    Pi5  Pi4  Pi3  Pi2  Pi
```

Nested summations, $\sum_{i=i \text{ min}}^{i \text{ max}} \sum_{j=j \text{ min}}^{j \text{ max}} f$ take on this format: `Sum[f, {i, imin, imax}, {j, jmin, jmax}]`.

A working example can be found in this summation:

$$\sum_{i=1}^5 \sum_{j=1}^i 2x^i y^{2j}$$

```
In[3]:= Sum[2*x^i * y^(2*j), {i,1,5},{j,1,i}]
```

```
Out[3]= 2 x y2 + 2 x2 y2 + 2 x3 y2 + 2 x4 y2 + 2 x5 y2 + 2 x2 y4 + 2 x3 y4 +
> 2 x4 y4 + 2 x5 y4 + 2 x3 y6 + 2 x4 y6 + 2 x5 y6 + 2 x4 y8 + 2 x5 y8 +
> 2 x5 y10
```

Products follow the same formats as summations, $\prod_{i=i \text{ min}}^{i \text{ max}} f$ would be represented as `Product[f, {i, imin, imax}]`.

For example

```
In[5]:=
Product[x+i*y, {i, 1, 4}]
```

```
Out[5]=
(x + y) (x + 2 y) (x + 3 y) (x + 4 y)
```

Boolean Values

The Boolean values are "true" and "false." In Mathematica, you can test if an expression is true or false. For example, `==` is an equality test. Similarly, `!=` tests for an inequality. `<`, `<=`, `>`, `>=` are also tests. For example:

```
In[1]:=
x = 9
Out[1]=
9
```

```
In[2]:=
x == 9
Out[2]=
True
```

```
In[3]:=
x == 2
Out[3]=
False
```

```
In[4]:=
x != 9
Out[4]=
False
```

```
In[5]:=
x >= 7
Out[5]=
True
```

Notice that there are spaces before and after the operators. This is not necessary, although it is important for clarity. If you typed `x!=9`, it is confusing whether you mean to set `x` factorial equal to 9, or if you mean to test if `x` is not equal to 9.

Graphics

Mathematica is a graphics lover's dream program. This section just touches on the plethora of plotting options available. We highly recommend *Mathematica*, by Stephen Wolfram, and *The Beginner's Guide to Mathematica Version 2*, by Theodore Gray and Jerry Glynn, both published by Addison-Wesley Publishing Company.

These plots were captured from an X-Windows session. You will receive no plots via the ASCII sessions. If you need help look at our X-Window application page.

Two-Dimensional Plotting

Mathematica uses the `Plot` command to produce 2-D plots. You basically specify the equation to be plotted followed by a list that contains the variable and min and max values for a range. The statements usually look like this, `Plot[f[x], {x, xmin, xmax}]`. For example, here is a simple parabola:

```
In[1]:=
Plot[(3*x^2 - 4), {x, -10, 10}];
```

You can plot multiple curves on one graph in a similar way by enclosing the functions as a list using brackets, `{}`:

```
In[2]:=
Plot[{Sin[x], x - x^3/6 + x^5/120}, {x, -4, 4}];
```

You can specify dashed lines, different colors, etc. with the `PlotStyle` command tacked onto the end of the `Plot` command. Let's look at the same plot as above, only using `PlotStyle -> Hue` to add some color:

```
In[3]:=
Plot[{Sin[x], x - x^3/6 + x^5/120}, {x, -4, 4}],
PlotStyle -> {Hue[2/3], Hue[1/3]}];
```

Many plots have extreme ranges. You can specify the axis ranges but often Mathematica is smart enough to figure it out on its own:

```
In[4]:=
Plot[Tan[x], {x, -2 Pi, 2 Pi}, PlotStyle -> Hue[2/3] ];
```

You can define functions (or answers) and then print the out directly:

```
In[5]:=
```

```
f[x_] := x^3
```

```
Out[5]=
```

```
f[x_] == x^3
```

```
In[6]:=
```

```
Plot[f[x], {x, -5, 5}];
```

You need to tell Mathematica when you wish to graph polar plots. You do this by loading the graphics library:

```
In[7]:=
```

```
Needs["Graphics`Graphics`"]:
```

NOTE: The second ``Graphics`` is encased by back-ticks, often found below the tilde. This is important to remember. You can now plot your graph:

```
In[8]:=
```

```
PolarPlot[ {t, t^1.1}, {t, 0, 2Pi}];
```

Implicit plots, or 2-D plots that depend on two variables (for example, circles and ellipses) also need to have a special library loaded in the same way as polar plots:

```
In[9]:= Needs["Graphics`ImplicitPlot`"];
```

Again, note the use of ``back ticks.`` Now lets plot:

```
In[9]:=
```

```
ImplicitPlot[ (x^2 + y^2)^2 == (x^2 - y^2), {x, -2, 2}];
```

Three-Dimensional Plotting

It is quite easy (and fun) to plot in three dimensions. The format is similar as above. One just uses the `Plot3D` command:

```
In[10]:=
```

```
Plot3D[x^3 + y^3, {x, -3, 3}, {y, -3, 3}];
```

You can increase the resolution of the plots with the `PlotPoints` option. For example, let's look at a surface:

```
In[11]:=
```

```
Plot3D[ (x^3) * Sin[a*x^2], {a,0,5}, {x,0,3}];
```

Looks a little rough. But let's smooth things out a bit with the `PlotPoints` option:

```
In[11]:= Plot3D[ (x^3) * Sin[a*x^2], {a,0,5},{x,0,3}, PlotPoints -> 50];
```

The orientation of the box is defined by default to be:

This can be changed in a pull-down menu.

Animations

Mathematica can also produce animation of plots via a simple `Do` loop. This way you can see how a plot can vary over different conditions. Below is an animation of the Cosine function

```
In[11]:= Do[Plot[Cos[n*x], {x, 0, 2Pi}],  
  {n, 1, 2, 0.2}];
```



Other Plots

There are many more types of plots that one can make with Mathematica.

Algebra

Defining Functions

Functions are defined in Mathematica using the `:=` operator. For example:

```
In[1]:=
      f[y_] := 3 y + 5
```

Now you can use this function with the variable `y` being any number of expressions.

```
In[2]:=
      f[1]
Out[2]=
      8
```

You can use the function on both numerical and symbolic expressions.

```
In[1]:=
      f[3^2]
Out[1]=
      32
```

```
In[2]:=
      f[z*q]
Out[2]=
      5 + 3 q z
```

It is very important to include the `_` (underscore) after each variable, and remember to use square brackets, `[]`. If you've forgotten the definition of a function `f`, use the command `?f` to show its definition. If you've finished using `f`, use the command `Clear[f]` to clear the definition of `f`.

Functions give you a lot of flexibility. For example, you could define a function, `area`, like this:

```
In[1]:=
      area[length_, width_] := length*width
```

And then solve the function for two variables:

```
In[2]:=
      area[2, 10]
Out[2]=
      20
```

You can even include units in your arguments (words are symbolic values!).

```
In[3]:=
    area[10 meters, 5 meters]
Out[3]=
    50 meters2
```

You, naturally have the option of plotting any functions that you have defined:

```
In[2]:=
    f[x_] := Abs[x]
In[3]:=
    Plot[ {g[x],f[x]}, {x, -10, 10} ]
```

Polynomial Manipulation

There are several built-in operators in Mathematica that allow the user to quickly and easily manipulate even large polynomials. The two simplest, and commonly used, ones are [Factor](#) and [Expand](#).

```
In[1]:=
    Factor[a b c + 2 a c + b c + 2 c + 3 a b + 6 a + 3 b + 6]
Out[1]=
    (1 + a) (2 + b) (3 + c)
```

```
In[2]:=
    Expand[(1 + a) (2 + b) (3 + c)]
Out[2]=
    6 + 6 a + 3 b + 3 a b + 2 c + 2 a c + b c + a b c
```

```
In[3]:=
    Factor[E*Tan[x]^2 + 3*E*Tan[x] + 2*E - Tan[x]^2 - 3*Tan[x] - 2]
Out[3]=
    (-1 + E) (1 + Tan[x]) (2 + Tan[x])
```

You can find common denominators by using [Together](#). Then you can split the fractions apart with [Apart](#):

```
In[2]:=
    Together[( (3*x^2)/y ) + ( (4*x)/(2*z) )]
Out[2]=
    2 x y + 3 x2 z
```

$$\frac{\quad}{y z}$$

```
In[3]:=
Apart[(2*x*y + 3*x^2*z) / (y*z)]
```

```
Out[3]=
      3 x^2      2 x
      ---- + ----
       y         z
```

The command `Simplify` tries to put an expression in its simplest form. According to Mathematica, the simplest form is that which has the least number of elements. Here are some examples:

```
In[4]:=
Simplify[(x^2 + 2*x*y + y^2) / (x+y)]
```

```
Out[4]=
x + y
```

```
In[7]:=
Simplify[(3*a^3*b^2*c^2)/(a*b*c)]
```

```
Out[7]=
3 a^2 b c
```

Solving Equations

The command `Solve` takes two arguments. One is the equation to be solved. The other is the variable to solve for.

```
In[1]:=
Solve[x^2 - x == 6, x]
```

```
Out[1]=
{{x -> -2}, {x -> 3}}
```

Note: We used the double equals signs, `==`. This stands for equality (*do not confuse == with =*. The single `=` stands for assignment).

By using lists, you can solve for systems of equations with several variables. In these cases, the first argument is a list of the equations to be solved, and the second argument is a list of the variables to solve for. Lists are designated with curly brackets, `{}`. Mathematica will also tell you if there is more than one solution for your system of equations.

```
In[8]:=
  Solve[{x^2 + y^2 == 16, x^2 - 4 == y}, {x,y}]
```

```
Out[8]=
  {{y -> -4, x -> 0}, {y -> -4, x -> 0}, {y -> 3, x -> -Sqrt[7]},
  > {y -> 3, x -> Sqrt[7]}}
```

You can see that there are four solutions (three unique) to these equations. To better observe the output, we can format the answer by using the `//TableForm` option at the end of our input stream:

```
In[9]:=
  Solve[{x^2 + y^2 == 16, x^2 - 4 == y}, {x,y}]/TableForm
```

```
Out[9]//TableForm=
  y -> -4    x -> 0
  y -> -4    x -> 0
  y -> 3     x -> -Sqrt[7]
  y -> 3     x -> Sqrt[7]
```

Sometimes there are no solutions to a set of equations. Mathematica will return a list with zero elements in these cases:

```
In[10]:=
  Solve[{x==5, x^2==5}, x]
```

```
Out[10]=
  {}
```

You can also solve expressions via replacement. For example, if we have the expression $3x^2 + 2x - 5$ and we want to solve it for $x=6$ we would do this:

```
In[11]:=
  (3*x^2 + 2*x - 5) /. x->6
```

```
Out[11]=
  115
```

Here we used two new symbols. The `./` designates **replace** and the `->` means **with**. So we are saying in the cited expression, *replace x with 6*. Naturally, we can perform multiple replacements:

```
In[12]:=
  (5*x^2*y - 4*x*y^2 + 7*x*y) /. {x->5, y->6}
```

Out[12]=
240

Calculus

Differentiation

Differentiation with Mathematica utilizes the `D` command. You need to specify an expression and the element you wish to differentiate with respect to. Some examples:

```
In[4]:=
D[(5*x^3 - 4*x^2 + 9*x - 8), x]
```

```
Out[4]=
9 - 8 x + 15 x^2
```

```
In[5]:=
D[(Sin[x] * (Cos[y]/Cos[x])), x]
```

```
Out[5]=
Cos[y] Sec[x]^2
```

```
In[6]:=
D[u[x] v[x] w[x], x]
```

```
Out[6]=
v[x] w[x] u'[x] + u[x] w[x] v'[x] + u[x] v[x] w'[x]
```

In this last example we differentiated three functions with respect to x . Mathematica, knows that u , v , and w are functions, but not knowing what those functions are used the standard product rule for differentiation. First derivatives of these functions are designated with a prime, i.e., u' , v' , and w' . Note that you can use these as input:

```
In[7]:=
D[u'[x], x]
```

```
Out[7]=
u''[x]
```

Likewise, you can specify to Mathematica that you want an n th derivative calculated. Here we will calculate the 2nd derivative of the given expression:

```
In[11]:=
D[(5*x^3 - 4*x^2 + 9*x - 8), {x, 2}]
```

```
Out[11]=
-8 + 30 x
```

Technically speaking, the `D` command calculates the *partial derivative*, $\frac{\partial}{\partial x} f$. If we have an expression and calculate its derivative like this:

```
In[8]:=
D[x^2 + y^2, x]
```

```
Out[8]=
2 x
```

we are assuming that `y` is independent of `x`. *What if it isn't?* Then we need to calculate the *total derivative*, $\frac{d}{dx} f$. We use the `Dt`, or Total Derivative command in the same way as the partial derivative command:

```
In[9]:=
Dt[x^2 + y^2, x]
```

```
Out[9]=
2 x + 2 y Dt[y, x]
```

Naturally, we can calculate the total derivatives of both variables with respect to each other:

```
In[10]:=
Dt[x^2 + y^2, {x,y}]
```

```
Out[10]=
Dt[x^2 , {x, y}] + Dt[y^2 , {x, y}]
```

Integration

Integration in Mathematica is performed using the `Integrate` command. For a simple integration of the type you would use this format, `Integrate[f, x]`, where `f` is a function of `x`. For example, suppose you wanted to solve this integral:

```
In[5]:=
Integrate[(15*x^2 - 8*x +9),x]
```

```
Out[5]=
      9 x - 4 x2 + 5 x3
```

Of course, there are times when we want to solve a definite integral of type

where we know the limits. We need to add a small list to the Mathematica inquiry, `Integrate[f, {x, xmin, xmax}]`. We see that this is consistent with other similar commands such as `Sum` and `Product`. Let's go ahead and integrate the above function from the limits of $x=-10$ to 10 :

```
In[6]:=
Integrate[(15*x^2 - 8*x +9), {x,-10,10}]
```

```
Out[6]=
10180
```

Naturally, Mathematica will perform integrals involving various operators:

```
In[1]:=
Integrate[Sin[x] Cos[x], x]
```

```
Out[1]=
      2
    -Cos[x]
-----
      2
```

Let's look at another integral,

```
In[1]:=
Integrate[(1/(x+Exp[x])), {x,0,2}]
```

```
General::intinit: Loading integration packages -- please wait.
```

```
Out[1]= Integrate[-----, {x, 0, 2}]
          x
          E  + x
```

What happened here? Well, Mathematica decided that the answer is an approximation so it left the integral in symbolic terms. If we want a numerical approximation we need to precede the Mathematica statement with the `N` or numerical command:

```
In[2]:=
N[Integrate[(1/(x+Exp[x])), {x,0,2}]]
```

```
Out[2]=
```

0.690176

Mathematica can also do nested integrals of the type

The format lies is step with others that we have seen, `Integrate[f, {x, xmin, xmax}, {y, ymin, ymax}]`. An example would be

```
In[7]:= Integrate[(x+2*y^2), {x,0,1},{y,0,x}]
```

```
Out[7]= 1
        -
        2
```

Limits

The command to find the limit of a function in Mathematica is `Limit`. It takes two arguments. The first is the function. The second is of the form $x \rightarrow 0$, where x is the variable and 0 is the value which the variable approaches. For example,

```
In[1]:= Limit[x^2, x -> 0]
```

```
Out[1]= 0
```

For certain functions, such as `Tan[x]`, the limit may differ depending on the direction from which you are approaching. Mathematica has an option called `Direction` to allow you to tell Mathematica from which direction you are approaching. A positive value results in a limit from the left, and a negative value results in a limit from the right.

```
In[1]:= Limit[Tan[x], x -> Pi/2, Direction -> 1]
```

```
Out[1]= Infinity
```

```
In[2]:= Limit[Tan[x], x -> Pi/2, Direction -> -1]
```

```
Out[2]= -Infinity
```

Solving Differential Equations

We looked at the `Solve` command when we discussed Algebraic Calculations. Mathematica can also solve differential equations with the `DSolve` command. The formatting for this command is `DSolve[eqns, y[x], x]` which means solve a differential equation for $y[x]$, taking x as the independent variable. For example:

```
In[9]:=
DSolve[ y'[x] == x^2 + 3*x, y[x], x]
```

```
Out[9]= {{y[x] ->  $\frac{3x^2}{2} + \frac{x^3}{3} + C[1]}$ }
```

As is often the case with differential equations, a constant, `C[1]`, is part of the solution. This can only be determined if boundary conditions are set as in this next example. Here we set up the two equations as lists using braces, `{}`:

```
In[11]:=
DSolve[ {y'[x] == x^2 + 3*x, y[0]==0}, y[x], x]
```

```
Out[11]= {{y[x] ->  $\frac{3x^2}{2} + \frac{x^3}{3}$ }
```

As with `Solve`, `DSolve` can solve a system of equations if you give a list of functions as the second argument.

```
In[14]:=
DSolve[ {x'[t] == y[t], y'[t] == x[t]}, {x[t], y[t]}, t]
```

```
Out[14]= {{x[t] ->  $\left(\frac{1}{2} \frac{E^t}{E} + \frac{t}{2}\right) C[1] + \left(\frac{-1}{2} \frac{E^{-t}}{E} + \frac{t}{2}\right) C[2]}$ ,
> y[t] ->  $\left(\frac{-1}{2} \frac{E^{-t}}{E} + \frac{t}{2}\right) C[1] + \left(\frac{1}{2} \frac{E^t}{E} + \frac{t}{2}\right) C[2]}$ }
```

2 E

2 E

You may have noticed that the solutions to all of the above equations are given in terms of $y[x]$. Let's look at an example:

```
In[15]:=
DSolve[ y'[x] == x^3, y[x], x]
```

```
Out[15]= {{y[x] ->  $\frac{x^4}{4} + C[1]}$ }
```

Sometimes, especially when you wish to substitute the answer into another equation, you may wish to have the answer in terms of y :

```
In[16]:=
DSolve[ y'[x] == x^3, y, x]
```

```
Out[16]= {{y -> Function[x,  $\frac{x^4}{4} + C[1]$ ]}}
```

We can now substitute this answer directly into another expression:

```
In[17]:=
y''[x] + y[x] /. %
```

```
Out[17]= {3 x2 +  $\frac{x^4}{4} + C[1]$ }
```

Vector and Matrix Operations

In Mathematica, vectors are lists, `{}`, and matrices are lists of lists.

```
In[11]:=
  SampleVector = {x,y}
```

```
Out[11]=
  {x, y}
```

```
In[12]:=
  SampleMatrix = {{a, b}, {c, d}}
```

```
Out[12]=
  {{a, b}, {c, d}}
```

To better see the matrix in a manner you are used to, try using the `//MatrixForm` option:

```
In[13]:=
  SampleMatrix//MatrixForm
```

```
Out[13]//MatrixForm=
      a    b
      c    d
```

We can do all of the traditional vector and matrix functions. For example, to calculate a dot product use the `.` command:

```
In[15]:=
  SampleMatrix . SampleVector // MatrixForm
```

```
Out[15]//MatrixForm=
      a x + b y
      c x + d y
```

```
In[17]:=
  SampleVector . SampleMatrix // MatrixForm
```

```
Out[17]//MatrixForm=
      a x + c y
```

$$b x + d y$$

Below are some of Mathematica's vector and matrix features:

```
In[18]:=
  Det[ SampleMatrix ]
```

```
Out[18]=
  -(b c) + a d
```

```
In[19]:=
  Transpose[ SampleMatrix ] // MatrixForm
```

```
Out[19]//MatrixForm=
      a   c
      b   d
```

```
In[20]:=
  Inverse[ SampleMatrix ] // MatrixForm
```

```
Out[20]//MatrixForm=
      d           b
-----          -(-----)
-(b c) + a d      -(b c) + a d

      c           a
-(-----)      -----
-(b c) + a d      -(b c) + a d
```

```
In[21]:=
  Dimensions [SampleVector]
```

```
Out[21]=
  {2}
```

```
In[22]:=
  Dimensions[ SampleMatrix ]
```

```
Out[22]=
  {2, 2}
```

```
In[24]:=
  Eigenvalues[ SampleMatrix ] // MatrixForm
```

```
Out[24]//MatrixForm=
  a + d - Sqrt[(-a - d)^2 - 4 (-(b c) + a d)]
```

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

```
In[25]:= Eigenvectors[ SampleMatrix ] // MatrixForm
```

```
Out[25]//MatrixForm=
```

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

```
> 1
```

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

```
> 1
```

Note that if you wish to see *both* the eigenvalues and the eigenvectors of a matrix you can use a single command, [Eigensystem](#).

```
In[27]:= Eigensystem[ SampleMatrix ] // MatrixForm
```

```
Out[27]//MatrixForm=
```

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

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

```
> 2
```

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

$$\begin{array}{c}
 2 c \\
 \\
 \left\{ \frac{-(-a + d - \text{Sqrt}[a^2 + 4 b c - 2 a d + d^2])}{2 c}, 1 \right\} \\
 >
 \end{array}$$

There are several other commands available to help you establish new matrices quickly.

```
In[28]:= DiagonalMatrix[ {x,y,z} ] // MatrixForm
```

```
Out[28]//MatrixForm=
      x    0    0
      0    y    0
      0    0    z
```

```
In[29]:= Array[ x, {5,3} ] // MatrixForm
```

```
Out[29]//MatrixForm=
      x[1, 1]  x[1, 2]  x[1, 3]
      x[2, 1]  x[2, 2]  x[2, 3]
      x[3, 1]  x[3, 2]  x[3, 3]
      x[4, 1]  x[4, 2]  x[4, 3]
      x[5, 1]  x[5, 2]  x[5, 3]
```

```
In[30]:= IdentityMatrix[4] // MatrixForm
```

```
Out[30]//MatrixForm=
      1    0    0    0
      0    1    0    0
      0    0    1    0
      0    0    0    1
```

Statistics

Mathematica can do a variety of statistical functions. **The Good News** is that Mathematica is a powerful mathematical tool, allowing you to perform a plethora of other functions on your data. **The Bad News** is that all the data must either be read in from a file (see below) or entered in as a list. There is no spreadsheet-like feature for Mathematica.

Before you start, you need to tell Mathematica that you will need the statistics package loaded (Note the use of backquotes, ```, in the statement):

```
In[2]:= Needs["Statistics`Master`"]
```

Now you are ready to enter in your data. Let's look at a list of numbers:

```
In[3]:= myData = {2.354, 5.653, 7.567, 9.324, 0.324,
                6.599, 4.334, 1.212, 3.546, 8.566,
                9.005, 0.336, 5.778, 3.096, 1.256,
                6.554, 7.265, 4.443, 2.045, 0.452};
```

Okay, we have our data. Now let's play:

```
In[4]:= Mean [myData]
```

```
Out[4]= 4.48545
```

```
In[5]:= Median [myData]
```

```
Out[5]= 4.3885
```

```
In[6]:= GeometricMean [myData]
```

```
Out[6]= 3.03012
```

```
In[7]:= HarmonicMean [myData]
```

```
Out[7]=
```

```
1.52325
```

```
In[8]:=
RootMeanSquare [myData]
```

```
Out[8]=
5.37197
```

```
In[9]:=
StandardDeviation [myData]
```

```
Out[9]=
3.03295
```

```
In[10]:=
Variance [myData]
```

```
Out[10]=
9.19876
```

```
In[11]:=
Skewness [myData]
```

```
Out[11]=
0.0729398
```

```
In[12]:=
Kurtosis [myData]
```

```
Out[12]=
1.52487
```

This can get a bit tedious. Luckily, Mathematica allows you to generate several "*statistical reports*" with one command:

```
In[13]:=
LocationReport [myData]
```

```
Out[13]=
{Mean -> 4.48545, HarmonicMean -> 1.52325, Median -> 4.3885}
```

```
In[14]:=
DispersionReport [myData]
```

```
Out[14]=
{Variance -> 9.19876, StandardDeviation -> 3.03295,
> SampleRange -> 9., MeanDeviation -> 2.59419, MedianDeviation -> 2.61,
```

```
> QuartileDeviation -> 2.64075}
```

```
In[15]:=
  ShapeReport [myData]
```

```
Out[15]=
  {Skewness -> 0.0729398, QuartileSkewness -> -0.0368267,
```

```
> KurtosisExcess -> -1.47513}
```

Reading Data From A File

As we are aware, it is often that we have our data in some form of datafile, usually a table that was created by another program. Here, we have a file named `datafile` stored on the Cluster.

```
ibm-08> cat datafile
 2.354 5.653 7.567 9.324 0.324
 6.599 4.334 1.212 3.546 8.566
 9.005 0.336 5.778 3.096 1.256
 6.554 7.265 4.443 2.045 0.452
```

We can use the command `ReadList["file", Number]` to read in this data and store it as a list. Here we are reading in the data and storing it to list named `theData`:

```
In[25]:=
  theData = ReadList["datafile", Number]
```

```
Out[25]=
  {2.354, 5.653, 7.567, 9.324, 0.324, 6.599, 4.334, 1.212, 3.546,
 > 8.566, 9.005, 0.336, 5.778, 3.096, 1.256, 6.554, 7.265, 4.443, 2.045,
 > 0.452}
```

We can then perform all of the usual statistics on this dataset.

```
In[27]:=
  LocationReport [theData]
```

```
Out[27]=
  {Mean -> 4.48545, HarmonicMean -> 1.52325, Median -> 4.3885}
```

Programming with Mathematica

Converting Mathematica Expressions to C and Fortran

Mathematica is a tool. As is often the case, this tool is used in conjunction with other languages, mainly C and Fortran. Some expressions can be difficult to translate, so Mathematica does it for you. The two statements

used are `CForm` and `FortranForm`. For example, $\sum_{i=1}^5 \sum_{j=1}^i 2x^i y^{2j}$ is expressed this way in

Mathematica:

```
In[10]:=
Sum[2*x^i * y^(2*j), {i,1,5},{j,1,i}]
```

We can express this for C and Fortran.

```
In[11]:=
CForm[%]
```

```
Out[11]//CForm=
2*x*Power(y,2) + 2*Power(x,2)*Power(y,2) + 2*Power(x,3)*Power(y,2) +
2*Power(x,4)*Power(y,2) + 2*Power(x,5)*Power(y,2) +
2*Power(x,2)*Power(y,4) + 2*Power(x,3)*Power(y,4) +
2*Power(x,4)*Power(y,4) + 2*Power(x,5)*Power(y,4) +
2*Power(x,3)*Power(y,6) + 2*Power(x,4)*Power(y,6) +
2*Power(x,5)*Power(y,6) + 2*Power(x,4)*Power(y,8) +
2*Power(x,5)*Power(y,8) + 2*Power(x,5)*Power(y,10)
```

```
In[12]:=
FortranForm[%]
```

```
Out[12]//FortranForm=
- 2*x*y**2 + 2*x**2*y**2 + 2*x**3*y**2 + 2*x**4*y**2 + 2*x**5*y**2 +
- 2*x**2*y**4 + 2*x**3*y**4 + 2*x**4*y**4 + 2*x**5*y**4 +
- 2*x**3*y**6 + 2*x**4*y**6 + 2*x**5*y**6 + 2*x**4*y**8 +
- 2*x**5*y**8 + 2*x**5*y**10
```

Note that this does not evaluate the expression, it just generates code that you can import into your existing programs.

Converting for TeX Output

Often, you may want to take portions of your notebooks and format them for publication. Many people use a program called TeX. And, as you can expect from the previous example, the command that one can use to convert Mathematica output to TeX is [TeXForm](#).

```
In[13]:=
      TeXForm[%]
```

```
Out[13]//TeXForm=
```

```
2\,x\,{y^2} + 2\,{x^2}\,{y^2} + 2\,{x^3}\,{y^2} + 2\,{x^4}\,{y^2} +
2\,{x^5}\,{y^2} + 2\,{x^2}\,{y^4} + 2\,{x^3}\,{y^4} + 2\,{x^4}\,{y^4} +
2\,{x^5}\,{y^4} + 2\,{x^3}\,{y^6} + 2\,{x^4}\,{y^6} + 2\,{x^5}\,{y^6} +
2\,{x^4}\,{y^8} + 2\,{x^5}\,{y^8} + 2\,{x^5}\,{y^{10}}
```

After an addition of a carriage return to help the equation fit the page, the final TeX results look like this:

$$2x y^2 + 2x^2 y^2 + 2x^3 y^2 + 2x^4 y^2 + 2x^5 y^2 + 2x^2 y^4 + 2x^3 y^4 + 2x^4 y^4 + 2x^5 y^4 + 2x^3 y^6 + 2x^4 y^6 + 2x^5 y^6 + 2x^4 y^8 + 2x^5 y^8 + 2x^5 y^{10}$$

File Input And Output

Often you may be performing complex calculations on data that is stored in a separate file. Mathematica allows for file I/O. We have already seen an example on the [Statistics Page](#). Here we will dig a little deeper.

There are four basic commands.

- << Read in a text file
- >> Write to a text file
- >>> Append to a text file
- !! Display a text file

Let's see what we can do.

```
In[16]:=
      Expand[ (3*x2-5*y)^2 ] >> testfile
```

Now we can see if the file really exists on the machine ibm-03.

```
ibm-03> ls -asl testfile
 4 -rw-rw----  1 dale      NCSTeam      26 Nov 19 20:23 testfile
```

```
ibm-03> cat testfile
9*x2^2 - 30*x2*y + 25*y^2
```

Yes, it's there. Of course we could have used Mathematica to look at the file.

```
In[17]:=
  !!testfile

  9*x2^2 - 30*x2*y + 25*y^2
```

We can read and append to the file:

```
In[17]:=
  MyExpression := <<testfile
```

```
In[18]:=
  MyExpression
```

```
Out[18]=
  9 x22 - 30 x2 y + 25 y2
```

```
In[19]:=
  Expand[ (x+2*y)^4 ] >>> testfile
```

```
In[20]:=
  !!testfile

  9*x2^2 - 30*x2*y + 25*y^2
  x^4 + 8*x^3*y + 24*x^2*y^2 + 32*x*y^3 + 16*y^4
```

What about a list of numbers? Suppose we have a table of numbers on ibm-03 called datafile?

```
In[20]:=
  !!datafile

  2.354 5.653 7.567 9.324 0.324
  6.599 4.334 1.212 3.546 8.566
  9.005 0.336 5.778 3.096 1.256
  6.554 7.265 4.443 2.045 0.452
```

Let's try to read them into a list called `myList`.

```
In[20]:=
  myList= {<<datafile}
```

```
Out[20]=
  {195.547}
```

Not quite what we wanted. We need to tell Mathematica that the numbers are indeed a list. To do this we can use the command `ReadList["file", Number]` to read in this data and store it as a list. Here we will read in the data and store it to list named `myList`:

```
In[25]:=
myList = ReadList["datafile", Number]
```

```
Out[25]=
{2.354, 5.653, 7.567, 9.324, 0.324, 6.599, 4.334, 1.212, 3.546,
> 8.566, 9.005, 0.336, 5.778, 3.096, 1.256, 6.554, 7.265, 4.443, 2.045,
> 0.452}
```

We can then perform all of the usual statistics on this dataset.

```
In[27]:=
LocationReport [myList]
```

```
Out[27]=
{Mean -> 4.48545, HarmonicMean -> 1.52325, Median -> 4.3885}
```