Introduction to Maple

Maple is a computer algebra system (CAS), indicating that it can carry out most mathematical computations ***symbolically*** and ***exactly*** (i.e. not using floating-point arithmetic). The following information is intended to familiarize the user with the basics of Maple: Performing Basic Calculations; Expressions and Parameters; Matrix (and Vector) Calculations; Functions; Plotting; and Procedures.

1 Key Features

Before starting, it is important to point out a few *key* features of Maple:

- **Maple is not necessarily visually sequential.**
  What does this mean? Essentially, each Maple command is entered at a Maple prompt, which looks like `[ >`. Upon executing a command on one prompt, a new prompt is generated, awaiting the next command. Seems fairly simple ... but there's a catch. The user can go back and edit and execute (by hitting the return key) a previous command, thus possibly overwriting the value of a variable or parameter. This is not obvious, reading the Maple commands from top to bottom of the worksheet.

  Hence, order of execution is not forced to follow the order of appearance of commands in the worksheet. For this reason, it is often best to execute each new command with a new line in the worksheet, otherwise it's hard to trace errors in work.

- **All commands in Maple are ended with a semi-colon (;) or colon (:)**
  - Ending a command in Maple with a semi-colon (;) allows the execution of the statement (once Enter or Return is hit), and the results are displayed immediately after the group of Maple statements.
  - Ending a command in Maple with a colon (:) allows the execution of the statement (once Enter or Return is hit), but the results are not displayed to the user.

  NOTE: If you do not end a line in Maple with a semi-colon or colon you will get a message "Warning, premature end of input", or something similar. Just go back and end the statement with a semi-colon or colon, whichever is appropriate.

- **% Represents the previous result**
  This is similar to the `Ans` key on the TI calculators.

- **Multiplication must be indicated explicitly**
  Although you know that $2x + 4\cos(x)$ means multiply the x value by 2, and the $\cos(x)$ value by 4, then add the results together, you must tell Maple about the "understood" multiplication between the 2 and x and the 4 and the $\cos(x)$, by typing $2 * x + 4 * \cos(x)$.

- **The `with()` statement**
  Often times there are specialized functions built into the Maple libraries which aren't accessible by default. Then you must include these libraries by including a `with(library-name):` at the start of your Maple worksheet, before making a call to a special function. Note, it's typically preferable to use the colon to suppress the list of all of the functions included in this library.

2 Performing Basic Calculations in Maple

Maple can be used as a simple calculator. Simply type in the expression that you want to evaluate after the Maple prompt, making sure to end this expression with a semi-colon, and hit enter. Note, that there

some special function calls you may need: $|-2|$ is expressed by `abs(2)`; e^3 is expressed by `exp(3)`; $\arcsin(4) = \sin^{-1}(4)$ is expressed by `arcsin(4)`; $\sqrt{5}$ is expressed by `sqrt(5)`; and π is expressed by `Pi` (note the capitalization).

Example: Verifying $\sin^{-1}\left(\sin\left(\frac{5\pi}{4}\right)\right) = -\frac{\pi}{4}$:

```
> arcsin(sin(5*Pi/4));
```

```
                              - 1/4 Pi
```

3 Expressions and Parameters

Often times it is helpful to save expressions, or to make use of variables. In order to **define** an expression in Maple, you must use the colon equals (:=) notation. For example, to define the parameter, *tol*, to be 10^{-8}, you enter the following at the Maple prompt: `tol := 10^(-8);`. Thus, every time you refer to *tol* in the future, Maple will replace it's value with 10^{-8}.

Example: Setting the tolerance, *tol*, to be 10^{-8}:

```
> tol := 10^(-8):
> tol;
```

```
                           1/100000000
```

4 Matrix and Vector Calculations

In order to use the defined matrix and vector structures, and predefined Maple functions that operate on matrices and vectors, you need to include the library **LinearAlgebra** in Maple 9 which supersedes the library **linalg** on older versions of Maple. Thus, before using matrix or vectors, be sure to include the line `with(LinearAlgebra):`. You'll want to use the colon if you would like to supress the list of functions included in the `LinearAlgebra` library.

4.1 Entering Matrices and Vectors

In Maple, vectors and matrices are defined by using < and > pairs. Each < > pair designates entries in a single *column*, and columns are separated by |'s. Thus, matrices are defined as a collection of columns. For example to define

$$A = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix}, \quad \mathbf{x} = \begin{bmatrix} 2 \\ 4 \end{bmatrix}, \quad \text{and } \mathbf{y} = [5, 4, 3, 2, 1]$$

with the `LinearAlgebra` library type in

```
with(LinearAlgebra):
A := <<1,4>|<2,5>|<3,6>>;
x := <2,4>;
y := <5|4|3|2|1>;
```

or with the `linalg` package

```
with(linalg):
A := matrix(2,3,[1,2,3,4,5,6]); x := Vector([2,4]);
y := Vector[row]([5,4,3,2,1]);
```

There are several commonly used matrices and vectors, so they have special "shortcuts". A few of particular interest in the `LinearAlgebra` library are given below.

`A := ZeroVector(3)`	creates a 3×1 vector of ones
`B := ZeroMatrix(5,4)`	creates a 5×4 matrix of zeros
`C := UnitVector(3,5)`	creates a 5×1 vector with a 1 in the third entry (0's elsewhere)
`D := DiagonalMatrix(<3,2,1>)`	creates a 3×3 matrix whose main diagonal is $< 3, 2, 1 >$

Example: Creating the matrix, D, with diagonal elements $-3, -1, 1, 3$:

```
> with(LinearAlgebra):
> d := <-3,-1,1,3>;  #  Or with the linalg package use d := Vector([-3,-1,1,3]);
```

$$d := \begin{bmatrix} -3 \\ \\ -1 \\ \\ 1 \\ \\ 3 \end{bmatrix}$$

```
> DM := DiagonalMatrix(d);  # with linalg use DM := Matrix(1..4,1..4,d,shape=diagonal);
```

$$DM := \begin{bmatrix} -3 & 0 & 0 & 0 \\ & & & \\ 0 & -1 & 0 & 0 \\ & & & \\ 0 & 0 & 1 & 0 \\ & & & \\ 0 & 0 & 0 & 3 \end{bmatrix}$$

4.2 Referencing an Entry in a Matrix or Vector

Matrices in Maple are indexed by integers starting with 1. To see or use the certain entries of the matrix A or vector $\mathbf{x}$, do the following:

`A[i,j]`	references the ij^{th} entry of A
`x[k]`	references the k^{th} element of the vector $\mathbf{x}$

4.3 Matrix and Vector Arithmetic

Special functions are required for matrix and vector arithematic:

Function	Result
`Add(A,B);`	returns $A + B$ where A and B vectors or matrices of the same size
`Multiply(A,B);`	returns $A * B$ where A and B are vectors, matrices, or scalars of appropriate size (can be dot product)
`Transpose(A);`	returns the transpose of the matrix A
`MatrixInverse(A);`	returns the inverse of the nonsingular matrix A
`Determinant(A);`	returns the determinant of the matrix of A
`CrossProduct(x,y);`	returns the cross product of the vectors $\mathbf{x}$ and $\mathbf{y}$

Example: Matrix, Vector, and Scalar Multiplication:

```
> A := <<1,4>|<2,5>|<3,6>>;
```

$$A := \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix}$$

```
> x := <-1,0,1>;
```

$$x := \begin{bmatrix} -1 \\ 0 \\ 1 \end{bmatrix}$$

```
> Multiply(A,x);
```

$$\begin{bmatrix} 2 \\ 2 \end{bmatrix}$$

```
> B := <<3,3>|<3,3>>;
```

$$B := \begin{bmatrix} 3 & 3 \\ 3 & 3 \end{bmatrix}$$

```
> Multiply(B,A);
```

$$\begin{bmatrix} 15 & 21 & 27 \\ 15 & 21 & 27 \end{bmatrix}$$

```
> Multiply(5,A);
```

$$\begin{bmatrix} 5 & 10 & 15 \\ 20 & 25 & 30 \end{bmatrix}$$

5 Functions

There are several ways to define functions in Maple.

5.1 Using Arrow Notation

This method emphasizes the mapping nature of a function

5.1.1 Defining the Function

$function\_name := variables- > expression$

Example: Arrow Notation to define $f(w,v) = \sin(w)\cos(v)$:

```
> f := (w,v) -> sin(w)*cos(v);
```

```
                    f := (w, v) -> sin(w) cos(v)
```

5.1.2 Evaluating the Function

$function\_name(input\_value)$

Example: Evaluating $f(w,v) = \sin(w)\cos(v)$, defined with arrow notation:

```
> f(Pi/4,Pi/6);
```

```
                          1/4 sqrt(2) sqrt(3)
```

5.2 As a Formula

This method emphasizes the expression

5.2.1 Defining the Function

$function\_name := expression$

Example: Formula notation to define $f(w,v) = \sin(w)\cos(v)$:

```
> f := sin(w)*cos(v);
```

```
                          f := sin(w) cos(v)
```

5.2.2 Evaluating the Function

`subs(`$variable\_name = input\_value$`,` $function\_name$`)`

Example: Evaluating $f(w,v) = \sin(w)\cos(v)$, defined with formula notation:

```
> subs(w=Pi/4, v=Pi/6, f);
```

```
                        sin(1/4 Pi) cos(1/6 Pi)

> evalf(subs(w=Pi/4,v=Pi/6,f));

                                .6123724357
```

5.3 Using a Procedure

This method emphasizes the procedural aspect of a function

5.3.1 Defining the Function

function_name := `proc (`*variables*`)` *expression* `end;`

Example: Using a procedure to define $f(w, v) = \sin(w)\cos(v)$:

```
> f := proc(w,v) sin(w)*cos(v) end;

              f := proc(w, v) sin(w)*cos(v) end proc
```

5.3.2 Evaluating the Function

function_name(*input_value*)

```
> f(Pi/4,Pi/6);

                          1/4 sqrt(2) sqrt(3)
```

6 Plotting

Maple is capable of plotting explicit, implicit, and parameterized functions in 2D and in 3D, as well as plotting approximate solutions to differential equations. There are also a plethora of options for the plotting commands. The best way to learn about the plotting abilities of Maple is to look at their `Help` files. To access Maple's help files, simply go to the `Help` menu option on the top bar and select `Topic Search...`, or, if you know the function name you want help with, type `?`*FunctionName* at the prompt. Below are a few examples of plotting. Note that Maple uses the **double dot** (..) in defining the range of input values!

6.1 Plotting a single-variable, explicitly defined function:

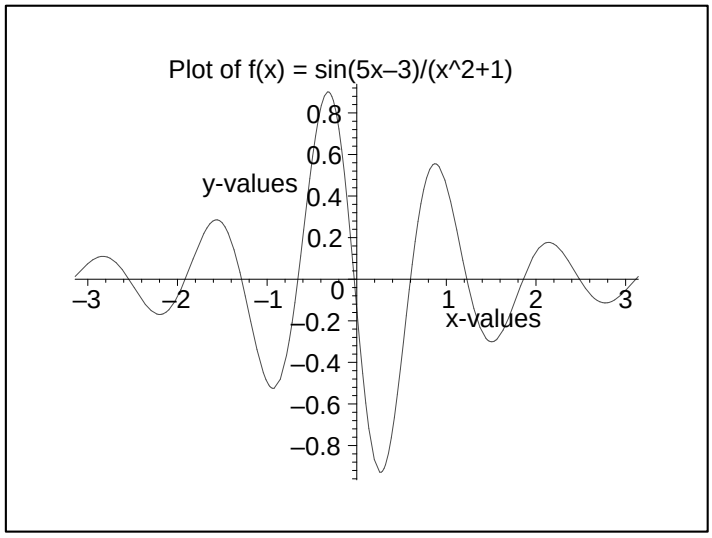
If you've defined $f(x) = \sin(3x - 5)$ using the arrow notation or the procedure, you can plot the graph of $f(x)$ for $-\pi \leq x \leq \pi$ with the command:
`plot(f(x),x=-Pi..Pi);`
(If you've used the formula notation, you do `plot(f,x=-Pi..Pi)`, i.e. drop the (x) from the $f(x)$).

Example: Plotting $f(x) = \dfrac{\sin(5x - 3)}{x^2 + 1}$ over $-\pi \leq x \leq \pi$

```
> f:= x -> sin(5*x-3)/(x^2+1);
```

```
                          sin(5 x - 3)
                f := x -> ------------
                                2
                               x  + 1
```

```
> plot(f(x),x=-Pi..Pi,title="Plot of f(x) = sin(5x-3)/(x^2+1)",
  labels=["x-values","y-values"]);
```



6.2 Plotting a single-variable explicit function:

If you haven't previously defined a function, you can still plot it by substituting in the expression for the $f(x)$ in the call above:

Example: Plotting $f(x) = \dfrac{\sin(5x-3)}{x^2+1}$ over $-\pi \leq x \leq \pi$

```
> plot(sin(5*x-3)/(x^2+1),x=-Pi..Pi,title="Plot of f(x) = sin(5x-3)/(x^2+1)",
  labels=["x-values","y-values"]);
```

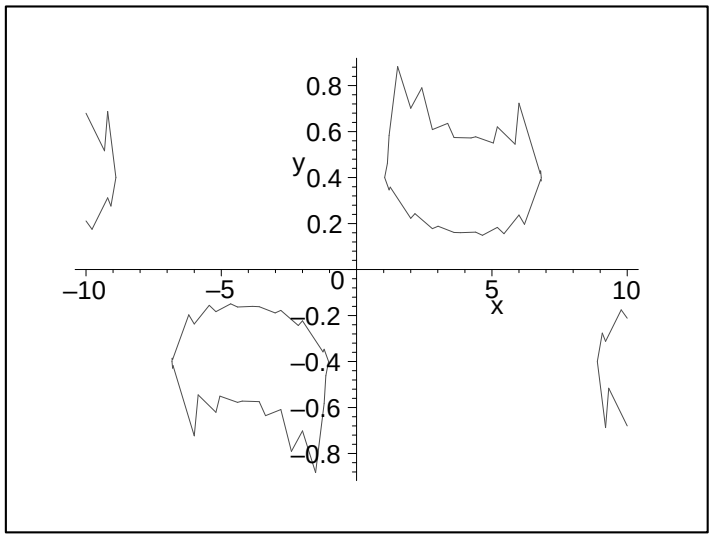
6.3 Plotting a single-variable implicit function:

You need to include the `plots` library to use the `implicitplot` function, so start by including the `with(plots):` command.

Example: Plotting $\sin(xy) = |y|$ over $-10 \leq x, y \leq 10$

```
> with(plots):
```

```
> implicitplot(sin(x*y) = abs(y), x=-10..10,y=-10..10);
```

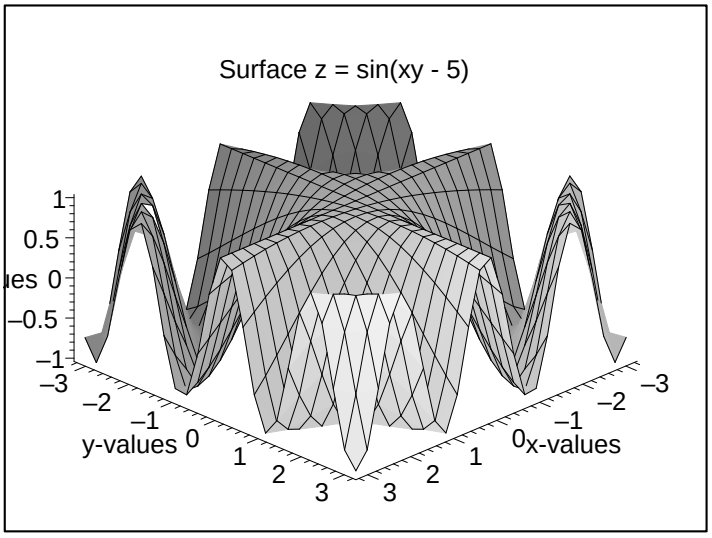


6.4 Plotting a two-variable function:

You can use `plot3d()` to plot functions of two variables that have been previously defined as well as functions you have not defined (just as you could with single-variable functions).

Example: Plotting the surface $z = sin(xy - 5)$ over $-\pi \leq x, y \leq pi$

```
> plot3d(sin(x*y -5),x=-Pi..Pi,y=-Pi..Pi,axes=frame,title="Surface z = sin(xy - 5)",
  labels=["x-values","y-values","z-values"]);
```



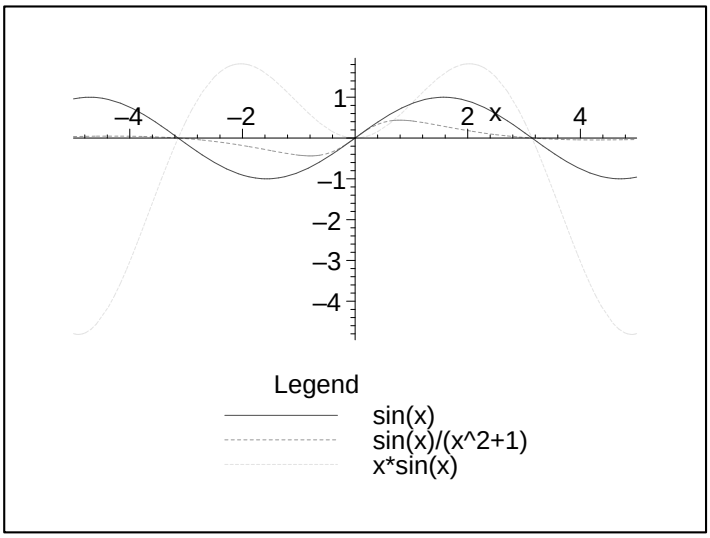
6.5 Plotting multiple functions:

There are a couple of ways to plot functions:

1. You can give Maple a set of functions in one plotting command by listing the functions in braces: $\{f(x), g(x), h(x)\}$.

   Example: Plotting graphs of $y = \sin(x)$, $y = \frac{\sin(x)}{x^2+1}$ and $y = x\sin(x)$ over $-5 \leq x \leq 5$

   ```
   > plot([sin(x), sin(x)/(x^2+1), x*sin(x)], x=-5..5, linestyle=[1,2,3],
     legend=["sin(x)","sin(x)/(x^2+1)","x*sin(x)"]);
   ```



2. You can name and save each plot, and display them together using the `display()` command (this is particularly handy when the plots aren't generated by the same type of plot command). Note that the `display()` command is included in the `plots` library, so you must have included this library before trying to use the display command.

   Example: Plotting graphs of $y = \sin(x)$, $y = \frac{\sin(x)}{x^2+1}$ and $y = x\sin(x)$ over $-5 \leq x \leq 5$

```
> with(plots):
> p1 := plot(sin(x), x=-5..5, linestyle = 1):
> p2 := plot(sin(x)/(x^2+1),x=-5..5, linestyle = 2):
> p3 := plot(x*sin(x), x=-5..5, linestyle = 3):
> display(p1,p2,p3);
```

   NOTE: Use the colon after naming a plot, rather than a semi-colon.

7 Procedures

You can create your own more complicated functions or programs in Maple using procedures. You've seen procedures before as one way to define a function. In order to get started writing your own procedures in Maple, there are a few things you should be familiar with:

7.1 Local Variables:

Maple supports local variables that are used only during execution of a procedure and module then discarded. Simply declare the variables to be local inside the procedure statement.

Example: The "Hello World" Procedure

```
> Hello := proc(n)
> local i;
> for i from 1 to n do
> printf(" Hello World! ");
> end do;
> end proc;

  Hello := proc(n)
```

```
local i;
    for i to n do printf(" Hello ") end do
end proc

> Hello(5);
 Hello World!  Hello World!  Hello World!  Hello World!  Hello World!
```

7.2 Looping:

The example above illustrates one looping structure in Maple. There are several key things to notice:

- Indexed loops go "from (start) to (end)", you can also indicate the step size with a "from (start) by (step-size) to (end)"
- All loops have their own "end" statement.
- There are two primary types of loops:
  1. for / while / do loops:
     Example: `for i from 1 by 2 while i < 100 do ...  end do;`
  2. if / then / else loops:
     Example `if (a > b) then ...  else ...  end if;`

7.3 Output:

Procedures will return only one value, and by default, it's the value that was calculated last. If you want to return a different value, use the command `return` *desired_value*. Once Maple hits this line in the procedure, it will exit the procedure returning the given value.